University of Zurich[UZH]

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Analysis and Implementation of Arbitrage Bots in Centralized and Decentralized Finance

Claudio Gebbia
Zurich, Switzerland
Student ID: 19-712-173

Supervisor: Christian Killer, Dr. Thomas Bocek
Date of Submission: July 17, 2022

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

Decentralized Finance (DeFi) has gained a lot of interest in the past few years, shown by an estimated increase in deposited assets from $700 million in December 2019 to over $200 billion in 2022 [1]. It brings use cases from traditional finance to blockchain, where the need for financial intermediaries normally required in centralized finance (CeFi), such as brokers or banks, can be eliminated. However, due to the novelty and lack of regulations in this financial space, prices are very volatile, which can lead to deviation across exchanges. This opens up potential possibilities for arbitrage, the concept of taking advantage of price differences of the same asset across exchanges. In this thesis, arbitrage opportunities across centralized and decentralized exchanges for cryptocurrencies are analyzed by developing and evaluating arbitrage bot prototypes that perform arbitrage fully automatically. Different strategies to perform arbitrage, like cyclic or spatial arbitrage, are explored and compared. The state of arbitrage availability is evaluated, based on the data collected by the arbitrage bots.

Das dezentrale Finanzwesen (DeFi) hat in den letzen Jahren ein grosses Interesse geweckt, was der geschätzte Anstieg der investierten Vermögenswerte von $700 Millionen im Dezember 2019 auf über $200 Milliarden im Jahr 2022 zeigt [1]. DeFi bringt Anwendungen der traditionellen Finanzwirtschaft in die Blockchain-Welt, wo der Bedarf an Finanzintermediären, die normalerweise im zentralisierten Finanzwesen (CeFi) benötigt werden, wie Makler oder Banker, beseitigt werden kann. Aufgrund der Neuheit und fehlenden Regulierungen in diesem Finanzraum sind Preise jedoch sehr volatil, was zu Abweichungen zwischen den Börsen führen kann. Dies eröffnet potenzielle Möglichkeiten für Arbitrage, das Konzept der Ausutzung von Perisunterschieden des gleichen Vermögenswertes zwischen Börsen. In dieser Arbeit werden Arbitrage Möglichkeiten zwischen zentralisierten und dezentralisierten Börsen für Kryptowährungen analysiert durch die Entwicklung von Arbitrage-Bot-Prototypen, die Arbitrage vollautomatisch durchführen. Verschiedene Strategien zur Durchführung von Arbitrage, wie zyklische oder räumliche Arbitrage, werden erforscht und verglichen. Mit den von den Arbitrage-Bots gesammelten Daten wird die Verfügbarkeit von Arbitrage auf diesen neuen Märkten beurteilt.

Acknowledgments

I would like to thank my supervisors, Christian Killer and Prof. Dr. Thomas Bocek, for their support and assistance during the development of this thesis.

Furthermore, I want to thank Severin Wullschleger for his continuous engagement during the practical part of the project.

Finally, I would like to thank Prof. Dr. Burkhart Stiller, head of the Communication System Research Group (CSG) at the University of Zurich for giving me the opportunity to write my bachelor's thesis under his group.

Contents

Chapter 1

Introduction

The rise of cryptocurrencies has brought many use-cases from the traditional financial world to the blockchain world. While centralized exchanges (CEXes) that trade cryptocurrencies are very similar to the brokers familiar to most readers, the decentralized characteristics of this space gave rise to a new way of trading assets. By using a blockchain, financial intermediaries such as brokers and banks can be omitted. This is what Decentralized Finance (DeFi) tries to achieve. Trading, lending, borrowing, or even earning interest on staked assets can all be done in DeFi solely based on smart contracts [2]. This is a very new space, still far from being flawless, but it brings many promises traditional finance can never make due to its centralized and institution-driven nature [3]. On decentralized exchanges (DEXes), users are able to create markets for the exchange of any cryptocurrency pair.

The infancy and, therefore, imperfect aspect of DeFi has sparked a lot of criticism due to the very high stakes. The 24-hour trading volume on Uniswap V3, the most famous decentralized exchange (DEX), is no lower than 1.4 trillion USD, and the one of all DEXes combined at the time of writing, as high as 3.2 trillion USD [4]. This raises many questions about price fairness between exchanges or the overall legitimacy of actions happening on DEXes due to the lack of regulations in this space combined with the lack of centralized authority which would make sure markets are fair.

A concept that is not new to finance but even more important in DeFi is arbitrage. In the case of cross-exchange price differences on a certain token pair, arbitrage possibilities are created, allowing traders (arbitrageurs) to take advantage of these inconsistencies throughout the markets and make profits [5]. This is not only profitable for the arbitrageurs but also a necessity for DEXes to stay synchronized with the rest of the market. While arbitrage on CEXes is often done by the institutions themselves or by people with high performance computing resources, it is available to every user to perform arbitrage on decentralized markets. It can also be added that this is the only way decentralized markets can stay synchronized. The number of exchanges and new cryptocurrencies created every day, combined with the freedom of creating new pairs on any exchange, make the task of synchronizing and arbitraging every inconsistency complicated and leaves room for anyone to get a chance of yielding a reward by finding inconsistencies.

Arbitrage is considered by many as a risk-free way to make profits. Since in traditional markets it is mostly left to a very limited number of players to perform these kinds of trades, it makes it an even more attractive opportunity for many to try out in DeFi where there are no entry barriers and where it is completely transparent and open to everyone. This thesis implements different kinds of arbitrage bots in order to determine what the situation is in DeFi as well as CeFi concerning arbitrage opportunities across the markets. The questions of the efficiency and possibility of executing arbitrage trades in DeFi and potential advantages or disadvantages over centralized finance are answered.

1.1 Description of Work

In an initial step, existing arbitrage bots are categorized and evaluated. These findings act as a guideline for what bots should be implemented later in the thesis. A first arbitrage bot prototype is implemented to interact with CEXes. The price differences, as well as the arbitrage opportunities, are collected and assessed. This should provide an overview of the inconsistencies present in these more traditional market structures, applied to cryptocurrency trading. Next, two different types of arbitrage bot prototypes are implemented for DEXes on various blockchains, appraising the opportunities present using two different arbitrage strategies. This begins with building detection tools to monitor prices on different exchanges continuously. To be able to execute trades in case an arbitrage opportunity is found, the implementation of an execution tool in the form of a smart contract is performed.

The thesis goes through a detailed explanation of how different bots were implemented and the strategies used by them. Finally, the results collected by the bots are sorted and presented. These results lay the basis for an analysis performed in later chapters, where the different findings are compared. This creates the possibility to analyze and discuss the results, yielding an assessment of existing inconsistencies in the market on DEXes as well as CEXes and the difficulty of performing arbitrage. A closer look is taken at the DeFi market, analyzing the arbitrage trades available on different chains as well as coins available on those chains. The thesis also touches on the subject of a possible impact of miner extractable value (MEV) performed by miners on arbitrage opportunities in general.

1.2 Thesis Outline

In Chapter 2, the basic terminologies and technologies that cover the basis for this thesis are explained. Following, existing work and research done on arbitrage in centralized as well as decentralized exchanges is presented in Chapter 3. Chapter 4 contains the approach taken to implement different types of arbitrage bots and specifies the strategies, while in Chapter 5, the technologies used and parts of the code are discussed. Chapter 6 then presents the results and findings of the different arbitrage bots. Finally, the results are discussed in Chapter 7 before the summary and conclusion in Chapter 8.

Chapter 2

Background

The following chapter provides an overview of the most important aspects of cryptocurrencies and centralized as well as decentralized markets. It covers the necessary knowledge about subjects this thesis touches on and gives a brief introduction to the topic. The discussion begins with blockchain and its connection to cryptocurrencies, followed by an introduction to centralized and decentralized finance, covering smart contracts and automated market makers.

2.1 Blockchain and Cryptocurrencies

In 2008, S. Nakamoto introduced a model for a completely decentralized payment system based on cryptography called Bitcoin [6]. The idea was to eliminate the need for financial intermediaries when transferring money and to avoid fees, long transaction times, and dependency on financial institutions. At the model's core is an electronic coin transferable between two parties solely based on cryptographic proof. A key element introduced was the concept of a time-stamping blockchain in order to determine precisely when and in what order a transaction was executed, which solved the double-spending problem.

The time-stamping is crucial to the functionality of Bitcoin since it allows for an undoubted assertion of what account holds how many coins at each point in time. Each user holds the coin in a wallet protected by a unique "private key" that serves as an access code. Only the holder of the private key can access the funds stored in a wallet. Every single transaction as well as every wallet with its available funds are stored on the blockchain, which serves as a shared public ledger [7].

This permissionless nature of Bitcoin allows users to freely join and leave the system without fear of having their assets frozen by controlling parties.

The blockchain is run by miners across the world that validate each transaction. They check for the needed electronic signature and ensure the blockchain is still uninterrupted. For this effort, miners are rewarded with new tokens by the system. This amount changes depending on the gas price and gas amount needed by the processed transactions. Users

can set a gas price they are willing to pay for a transaction before submitting it. In addition, a maximum gas amount can be specified. The gas price multiplied by the amount of gas burned by the transaction results in the transaction cost that the user pays to the miner. Before creating a block, miners can decide upon the transactions they want to include in the block. While mostly choosing according to the highest gas price offered by the user, it turns out that this choice also gives the miner power to manipulate the order in which trades are executed [2], [8].

Since the introduction of Bitcoin, cryptocurrencies have gained in popularity, and many new tokens have emerged. They all are based on the same idea as Bitcoin [6], but try to solve different problems or improve the initial implementation. New ideas like the Ethereum blockchain proposed by V. Buterin [9] in 2013 which created the possibility of running decentralized applications of any kind on the blockchain, led to an increasing number of people engaging with cryptocurrencies. This resulted in considerable yields in the cryptocurrency markets over the past few years. The high profits present in these markets appealed to companies and individuals to invest resources and money. The underlying technologies also created new possibilities and tools like atomic trades or non-collateralized loans, which are not available in traditional finance to this day. With the rise of cryptocurrencies, exchanges seized the opportunity and started to support them as tradable assets.

2.2 Centralized Finance

The idea of centralized finance (CeFi) in cryptocurrency markets is to mitigate the risk that comes with decentralized assets, by introducing a financial institution that takes care of the technical side of cryptocurrencies and offers the users some kind of security [2]. These institutions are known as centralized exchanges (CEX). Some examples of CEXes are *Binance*, *Coinbase*, and *Kraken*. When using a centralized exchange, the institution gains full control over the user's assets as all coins are held in wallets under the CEX's custody. Centralized exchanges take care of buying, sending, or holding cryptocurrencies for users and provide customer support in case there is a problem with any of the services. This security and comfort come at the cost of a small transaction fee paid to the CEX for every trade.

Another particularity of centralized exchanges is that each user has to go through a "Know Your Customer" (KYC) process when using the CEX for the first time. During KYC, the users have to identify themselves with valid contact information and ID or passport. Regulations require that financial institutions know the identity and financial background of their customers before allowing them to engage in any kind of trading. Depending on the user's location and intent, this process can include several legal documents [10]. This removes the anonymity and permissionless nature of cryptocurrencies in exchange for a more familiar and comfortable interaction when buying or selling assets.

2.2.1 Market Makers

Most common centralized markets utilize the two-sided market principle, where buyers and sellers place their orders on the order book. Buyers state the amount and price they are willing to pay as a bid order (*cf.* B, Figure 2.1), while sellers state how much and for what price they are willing to sell their assets as an asks order (*cf.* A, Figure 2.1). These bid and ask orders are collected and visible to all traders. When a buyer places a bid order that matches a seller's ask order or vice versa, these two orders are matched, and the trade is executed between the two parties. Separating the lowest ask and the highest bid price is a gap called the bid-ask spread or spread of the market [11].

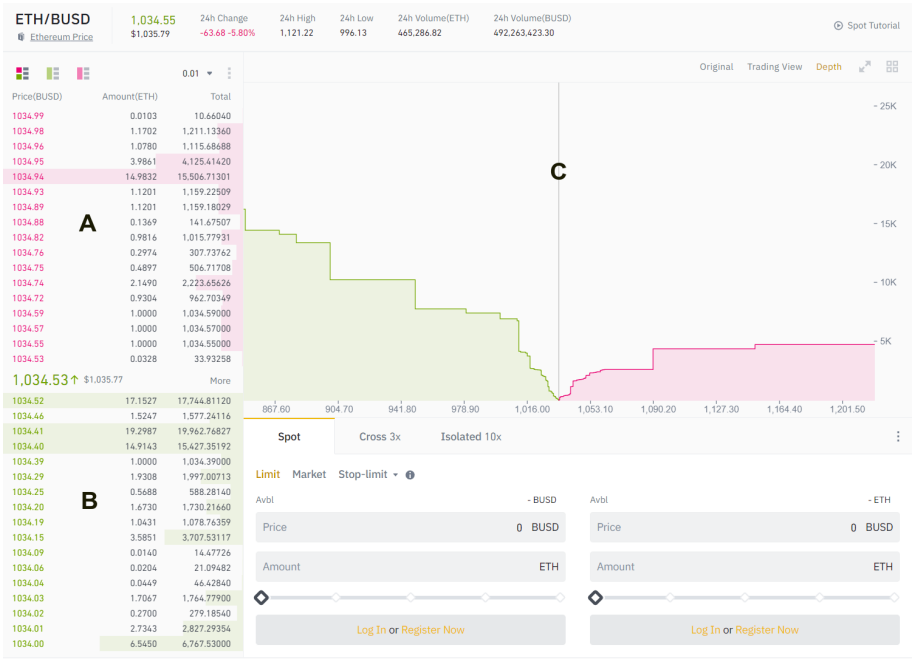

Figure 2.1: Orderbook ETH/EUR, taken from binance.com

If a trader wants to, there is the possibility to buy at the market price, which is a price in the market spread at which the CEX is able to fill an order as fast as possible [12]. Market makers ensure that when a user wants to buy or sell for a price in the market spread, there is always a counterparty who fills the trade directly. By placing bids and asks along the spread, they provide liquidity to the market and facilitate trading for all users [11]. The CEX itself often employs market makers since they provide big benefits to the exchange by making the market more attractive. In Figure 2.1, there is another diagram (*cf.* C) that shows a visual representation of the order book. The x-axis shows the price, and the y-axis shows the accumulated sum of all orders present in the order book that would execute at the corresponding price.

2.3 Decentralized Finance

With a focus on advancing financial technologies and services based on smart contracts, Decentralized Finance is a new sub-field of blockchain technology [2]. It builds on the decentralized and permissionless nature of blockchains and has the goal of creating trustless platforms for traders. Instead of relying on central authorities in DeFi, there are decentralized exchanges (DEX) that operate as smart contracts deployed on blockchains. When trading assets on decentralized exchanges, users interact with smart contracts directly with no centralized authority involved. Users that want to trade on a DEX need to connect their own wallet and can make trades without ever losing control over their assets [2]. Some examples of DEXes are *Uniswap*, *Sushiswap*, and *Pancakeswap*.

2.3.1 Automated Market Makers

The exchange rates of each asset on a DEX are calculated algorithmically through predefined formulas by using the liquidity reserves. This way of defining exchange rates is called an automated market maker (AMM). AMMs allow users to trade digital assets automatically instead of having a traditional order book with buyers and sellers listed. AMMs use liquidity pools (LP) as counterparties for transactions; therefore, the trader does not rely on other users' orders to trade [13]. The idea proposed by Ethereum founder V. Buterin [14] was that liquidity pools are pots of shared tokens that users can interact with to swap tokens. The supply of these liquidity pools comes from liquidity providers, who get exchange fees from users as an incentive. A swap works in the following way; if someone puts an amount of token A in the liquidity pool, the same person can take out a certain amount of token B while paying a small transaction fee to the liquidity providers. Therefore the total value of tokens in a pool never changes. For each token pair (A/B), a trader has to interact with a separate pool. If there is no pool for the desired token pair, a user may have to take a route through another token pool to perform a swap. The DEXes have implemented algorithms that automatically calculate the best route to take in this case, which makes the process easier [15].

The price of the tokens in a LP is determined by a mathematical formula. As mentioned before the total value of the pool should always stay the same this is achieved by having a constant product between the two assets. This 2.1 is the most commonly used formula, and AMM using this formula are called constant product AMMs:

$$balanceA * balanceB = k \tag{2.1}$$

This formula enforces that there is always a constant balance of assets in the liquidity pool. Suppose a trader decides to swap one token for the other. In that case, the amount that the user is getting can simply be computed by solving the constant product formula 2.1 for the desired token after subtracting the amount taken out.

In the liquidity pool x/y given in Figure 2.2, in which fees are omitted for simplicity, with the constant product formula:

$$x * y = k \tag{2.2}$$

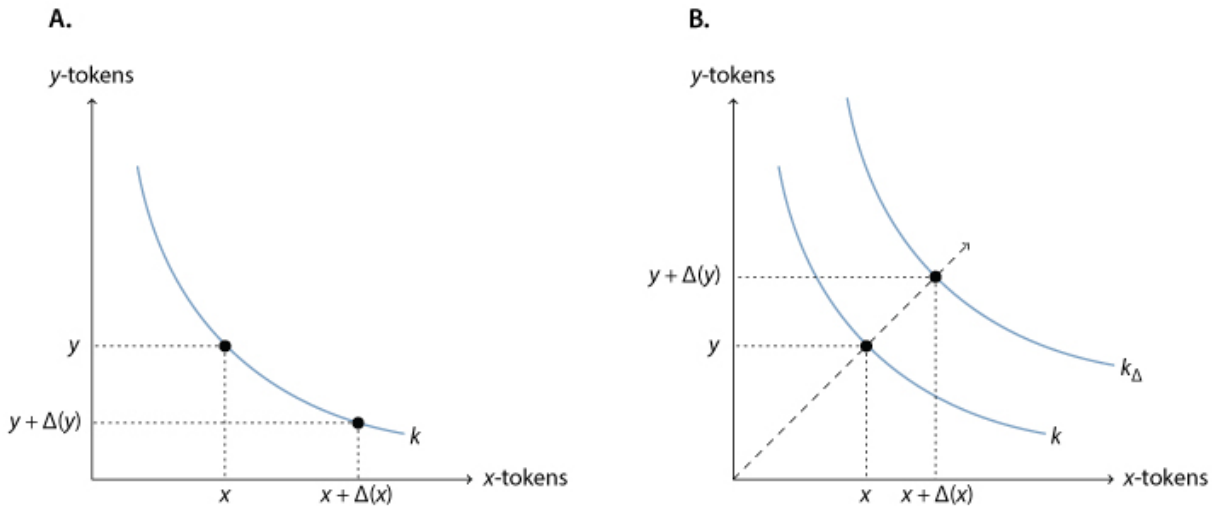

Figure 2.2: Change of price in pool through a swap in A and liquidity providing in B, taken from [16]

We assume in A that someone wants to swap x to y. Δx is the amount that wants to be swapped to y. To calculate Δy, the amount of token y the user can take out of the pool, it has to be ensured that formula 2.2 still holds after the swap. Therefore the equation, which is the constant product formula with Δx and Δy included in 2.3, can be solved for Δy, which results in 2.4.

$$(x + \Delta x)(y + \Delta y) = k \tag{2.3}$$

$$\Delta y = \frac{k}{x + \Delta x} - y \tag{2.4}$$

With this formula every trade from x to y can be calculated precisely. If someone wants to trade in the opposite direction, the equation has to be solved for Δx [17].

Example

Assuming there is a liquidity pool ETH/USDC with the supply of 100 ETH and 300'000 USDC, the constant product is 30'000'000. By applying 2.4, the amount returned when swapping 1 ETH to USDC can be calculated.

- $k = x * y = 30'000'000$
- $x = 100$
- $y = 300'000$
- $\Delta x = 1$

The result when inserting these numbers into 2.4 is -2970, which is the amount of USDC that can be taken out of the pool when trading 1 ETH. This is 30 USDC less than one would expect for the price of 3000 USDC/ETH initially present in the pool. The reason for this discrepancy is that at the moment ETH is added to the liquidity pool and USDC is removed, the value of ETH decreases while the value of USDC increases because of the lower supply. This particularity is known as the *slippage* of a trade which represents the difference between the spot price (the price present in the pool at trading time) and the realized price (the price actually traded for) [13]. Using this model, a liquidity pool cannot be depleted, as tokens will become more expensive as reserves decrease [16][17].

2.3.2 Smart Contracts

All of the logic and algorithms necessary for decentralized exchanges need to be deployed and run on the blockchain to ensure decentralization and undoubted reliability. This is what smart contracts (SCs) achieve, as they are a set of programs that can be deployed and run on a blockchain [5]. A majority of them are written in Solidity, a high-level, object-oriented programming language, but other languages such as Vyper or Yul are supported as well. As with all blockchain content, smart contracts can be viewed by anyone, ensuring complete transparency of every interaction. However, in order to guarantee immutability, they cannot be modified once deployed. Furthermore, smart contracts can also execute atomic transactions, which is critical when performing several swaps that rely on each other's successful execution. It is a crucial mechanism for all of DeFi [18].

DEXes use a number of smart contracts, which work together to execute different functionalities such as trading, staking, or retrieving information on the exchange. Uniswap V2 is one of the most copied DEX, and its most significant contracts are split into a core and a periphery. The core includes the factory contract, which is responsible for creating contracts for liquidity pools and can therefore be used to create new trading pairs on the exchange. An additional part of the system's core are the pair contracts that implement the pools, act as the AMM, and keep monitoring token balances in the pools. The second category of smart contracts on Uniswap V2 is the periphery, which contains the router contract. The latter offers interactions with the basic functionality of the exchange. It supports all transactions to interact with Uniswap V2, such as trading and providing or removing liquidity [19].

In order to interact with the smart contracts of a decentralized exchange, the address as well as the application binary interface (ABI) of the contracts, are needed. Smart contracts are stored in bytecode on the blockchain, which requires ABIs to be accessed by another program. They define the functions that are available on the contract and make sure that the returned data has the expected format [20].

2.3.3 Different Blockchains

All blockchains operate independently, which makes atomic transactions more difficult and stops the user from using the full potential of DeFi. This creates a problem since complete

decentralization cannot be achieved until blockchains are somehow interconnected. Several protocols like Polkadot and Cosmos try to mitigate this problem by supplying a platform for developers to connect different blockchains [21].

To this day, the most common way to work with assets between blockchains are wrapped tokens. They act as representatives of certain assets on different blockchains. By locking the real asset in a vault and replacing it with a representative of itself on a different blockchain or in another token standard, spending BTC on the ETH blockchain, for example, becomes possible. Since exchanges mostly operate with one token standard only, (e.g exchanges on the Ethereum blockchain only support ERC20 tokens), tokens like Bitcoin or even Ethereum itself (which is not an ERC20 token) have to be wrapped for trading on exchanges like Uniswap or Sushiswap (the wrapped versions of ETH and BTC are called WETH and WBTC) [22].

Apart from different token standards, blockchains also have different consensus mechanisms and therefore have different transaction fee costs or execution times. The consensus mechanism is used to agree on the state of every block on the blockchain. Its process includes validation and authentication of every transaction [23]. Ethereum, for example, is known to have one of the highest transaction fees in the space, also because it is one of the most used blockchains for smart contracts. Binance Smart Chain, on the other hand, being a more centralized blockchain, has transaction fees that equal a small fraction of the Ethereum ones. These differences in transaction fees come from the diverse functionalities of the blockchains. Nonetheless, many blockchains are so-called Ethereum Virtual Machine (EVM) chains and work the exact same way as the Ethereum blockchain; they can differ in their consensus mechanism. The two most famous consensus mechanisms are Proof of Work (PoW) and Proof of Stake (PoS). PoW has been criticized several times for its inefficiency and extremely high transaction fees. In PoW, every miner wastes computing power to compete for the possibility of mining the next block. PoS aims to improve that by making the decision about who will be validating the next block, based on the amount of deposited tokens a node has proportionally to the total deposited funds. This happens before miners have to use any computing power, resulting in lower transaction prices and better scalability [24].

2.3.4 Terra UST Depegging

Events such as the Terra UST Depegging that happened on May 7, 2022 can severely impact the whole cryptocurrency market. The algorithmic stablecoin UST, one of the largest stable coins on the cryptocurrency market, launched by Terraform Labs, started to deviate from its peg to the USD. Stablecoins are cryptocurrencies designed to maintain a fixed value over time by being pegged to another asset, most often FIAT currencies such as the USD. Exchanges often use stable coins as functional assets to offer trades between BTC or ETH, for example, and more ”secure” and stable assets without having to convert cryptocurrencies to FIAT currencies that are not on the blockchain and not as accessible [25].

However, there are different kinds of stablecoins, while USDT and USDC, for example, have at least 100% of the circulated amount backed by other assets and therefore securing

the worth of the currency itself, UST is an undercollateralized algorithmic stablecoin that solely relies on arbitrageurs to keep the price steady. This made its peg vulnerable to multiple large transactions that caused the price of UST to drop momentarily and triggered a wave of people scare-selling UST [26]. After only 3 days the price of UST was at 0.7$, and within a week, UST was only worth around 0.12$. The drop in value of UST, which at the time of writing, has a market price of 0.03$, is visible in Figure 2.3. This event greatly impacted the whole cryptocurrency market, with the total market cap falling by 30%. In the course of the thesis, this event is often mentioned to refer to a period of extreme price action in the cryptocurrency market.

Figure 2.3: Price chart UST/USDC from May to June 2022, taken from [27]

2.4 CeFi vs DeFi

In comparison to traditional centralized finance, decentralized finance offers several important advantages that all derive from its main ability to be decentralized and supposed to give users more control over their own assets. In the following, some of the most important advantages of DeFi over CeFi are listed and explained.

Transparency By using DeFi, users can view the specific rules on which the financial products and assets operate. Every asset has a specific range of operations it can fulfill. These operations are set at creation time and can be viewed by anyone. This is impossible in CeFi since most of the implemented functionalities belong to the institutions and are hidden from the user. Furthermore, every transaction and operation happening on the blockchain is visible to anyone [2].

Control In DeFi, every user has full control over their assets. Only the owner and holder of the wallet containing the assets can perform any transactions. It is impossible for any third party to move, destroy or alter anything saved on a wallet that is not controlled by themselves. This ensures the decentralized aspect of DeFi. Having a financial institution hold all assets, like in CeFi, comes with the known risks of centralization. The CEX can freeze, lose or take away the assets at any given time. The user has to trust the institution that is being used instead of having full control over their assets [2].

Accessibility Any computer with an internet connection can access or create DeFi products. After the initial deployment on the Blockchain, the distributed network of

miners will make sure the application works as planned [2]. Every actor that wants to, can participate in the market, which aims to make trading as accessible and fair as possible.

Atomicity Blockchain transactions support the atomic execution of several sequential financial operations. This means that programs can be implemented to ensure that either the entire transaction executes or the entire transaction fails if one step fails. In CeFi, this would only be possible with legal agreements that are slow and costly to set up [2].

Even though DeFi brings many interesting and innovative ideas, there are still many unresolved problems. Some problems like Risks and Performance are presented in various research [28], [2], [29] and are included in the following list.

Performance Blockchains like Ethereum have limited performance and transactions that can be executed per second. This prevents the current implementation from being scalable in the long term. Several new solutions are being implemented, with the most important one being Ethereum 2.0 with several performance improvements and a long-awaited change to proof of stake, close to being released [29].

Fees Due to the limited performance, traders on proof of work blockchains increase the amount of gas price they are ready to pay for a better chance of a miner including their trade in the next block, which is necessary for the trade to go through. This results in increasing transaction fees that have to be paid to the miner. These are often way higher than centralized finance fees [29].

Exploits Front running is the concept of using the power to manipulate the order in which trades are executed in favor of making profits [8], [28]. This poses a threat to the fairness of DeFi and is discussed further in section 7.

Risks DeFi still has many vulnerabilities or logic errors in the system caused by humans, these vulnerabilities can be exploited and create risks like smart contract attacks, flash-loan attacks, or market manipulation [2].

2.5 Arbitrage

Arbitrage refers to a series of actions on a specific asset that leads to an increase in its value. This can be done in different ways. Spatial arbitrage involves exploiting the discrepancy of prices in different markets by concurrently buying assets on one exchange and selling them on another exchange for a higher price, like in Figure 2.4 [30]. Cyclic or Intra-exchange arbitrage is an arbitrage method that searches for different routes through currency pairs on the same exchange to make a profit when finally trading back to the starting asset. The concept is not new and has been used in traditional finance to even out the price differences in markets around the world. When using these differences to execute profitable arbitrage trades, the arbitrageur reduces the price gap automatically as a byproduct of the trade [30].

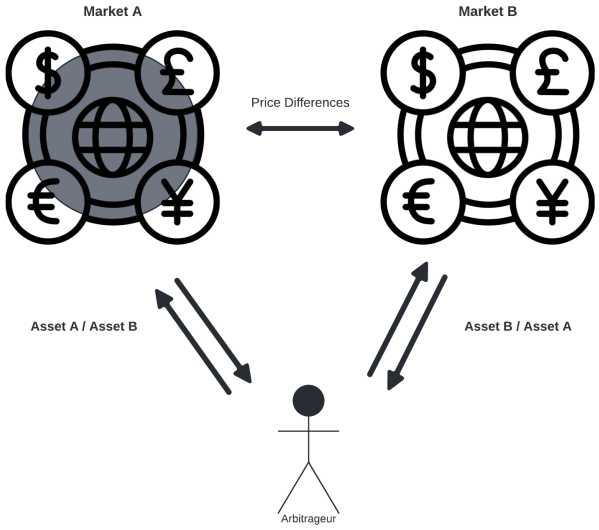

Figure 2.4: Arbitrage between two Markets

As new exchanges emerge in DeFi, they all need to be synchronized with the rest of the market. Arbitrage in CeFi is mostly about who has the fastest connections to every market and the fastest hardware to execute the necessary trades. Mostly it is performed by exchanges themselves or larger companies that have the resources as well as the capital to perform big arbitrage trades. There is a possibility that in the new markets of DeFi, the opportunities are more substantial and long term since it has not gained the attention of these large institutions yet. In addition, there are many more markets and exchanges to explore, which increase the probability of price differences and demand for arbitrageurs.

Chapter 3

Related Work

In this section, several papers covering arbitrage in cryptocurrency markets (DeFi and CeFi) are discussed and compared. First, arbitrage detection systems or trading bots implemented on centralized finance are examined, followed by research done in DeFi covering cyclic and spatial arbitrage.

3.1 Arbitrage on Centralized Exchanges

The uniqueness of cryptocurrency markets concerning the lack of regulations motivated I. Makarov and A. Schoar to analyze arbitrage opportunities present in centralized markets in [31]. They state that due to the lack of provisions ensuring investor protection combined with the wide range of independent exchanges, the importance of arbitrageurs is increased in cryptocurrency markets. Arbitrageurs make sure that prices across regions and different exchanges remain synchronized.

Their main approach was to analyze historical data from 34 CEXes across 19 countries to find out how high the arbitrage opportunities present were between December 2017 and February 2018. In this period, an estimate of around $2 Billion dollars of arbitrage profits were detected. Further analysis of the data revealed that the price differences between regions are considerably higher than within one country. The highest differences were measured between South Korean and US markets, where it reached more than 15% for several days. They also show that arbitrage opportunities are positively correlated with the strictness of capital controls. Capital controls slow down the capital flow in or out of the country which complicates the procedure for arbitrageurs of quickly evening out differences. This also results in higher arbitrage opportunities between crypto-assets and fiat currencies than between to two crypto-assets [31].

In [32] S. Bistarelli and colleagues experimented with arbitrage on CEXes with Bitcoin and implemented an arbitrage bot for CEXes. They used the *Sharp Ratio*, a metric used in finance to compare performance between different investments, as a main calculation metric for arbitrage opportunities between two exchanges. Whenever the Sharp Ratio between two identical portfolios on two exchanges varies, it can be considered an indication

that one of the two exchanges has to correct itself towards the other. As an initial step, the presented strategy was tested on historical data of two exchanges, and it yielded 30% returns after the first test run. Later on, the strategy was tested on real markets, by checking bid and ask prices with a fixed frequency on *bitbay*, *bitfinex*, *bitstamp*, and many more exchanges. Their bot was able to perform a 250% return on their initial 1000$ investment in two years. Several constraints like a daily spending limit of 100 USD and only executing transactions with a gain of minimum 2% made sure that the bot only engaged in the most lucrative trades. The paper of Bistarelli *et al.* provides fundamental evidence that arbitrage in centralized cryptocurrency markets, specifically with bitcoin, is indeed possible and can be lucrative.

Nowadays, a variety of websites and tools is available, which offers arbitrage detection with an easy to use dashboard. These systems are analyzed and compared in [30]. After conducting system analysis of some of the most famous arbitrage tools, for instance, Gekko, Blackbird, Arby.Trade and Zenbot, Levus and colleagues attempted to create an optimal information system for arbitrage detection as well as execution. This tool was then connected to a telegram bot that gives trading advice for arbitrage situations, with the most important information revealed to users in the form of a message.

Intra exchange arbitrage or cyclic arbitrage on Binance has been looked at in [33]. To discover paths through pairs on Binance, the Bellman-Ford algorithm has been used, which computes single-source shortest paths in a graph. By using the prices between the pairs as weights for the edges, this algorithm is able to find the lowest cost route to trade through a list of pairs and end up at the goal asset. The starting and goal asset used was Bitcoin, which resulted in an easy way to assess profits after each iteration, but also proved to have limitations regarding usable pairs. Therefore in a second attempt, every coin was allowed as starting and goal asset. By applying the proposed strategy, Han managed to increase his initial investment of 5000 USD by several hundred USD [33].

3.2 Arbitrage in Decentralized Exchanges

Trades and behavior of arbitrageurs on DEXes have been analyzed by Wang and colleagues between 4. May 2020 and 15. April 2021 in [5], with a focus on cyclic arbitrage. Due to the complete transparency of DEXes, they were able to follow and assess the trades performed by arbitrageurs on Uniswap V2. After examining exploitable arbitrage opportunities on Uniswap V2, they observed a total of 10 ETH (40'000 USD at that time) worth of arbitrage trades over a period of four months. Regarding the topic of arbitrageurs' behavior and strategies, they noticed that 85% of the cyclic transactions had a cycle length of 3. Further on they detected that during the measured time period, 295'518 arbitrage trades were executed atomically, or in other words, through a smart contract that executes all swaps in one transaction, while only 88 trades were executed sequentially with single calls for each trade. Traders using smart contracts can ensure that their arbitrage transactions are executed completely and also be certain a profit has been made by the end of the trades and otherwise revert the transactions. This justifies the findings that out of all 295'518 atomically executed arbitrage trades, only 0.3% had a negative revenue, compared to the 52.3% negative revenue arbitrage trades on the sequential arbitrage executions [5].

In [34] a ”market inefficient searching tool” has been built to discover a maximum profit route on several DEXes. In the initial phase, the tool searches for profit routes through Uniswap V2, trying to find cyclic arbitrage opportunities. However, this turned out to not be profitable since the three presented routes all exhibited negative profits. They tried the same routes on three other DEXes and found only one (*1inch*) to be profitable, although only multiplying their token count by 1.007. Furthermore, [34] discussed several factors that play an important role when trying to find arbitrage opportunities.

Trade Size Increasing the order size is shown to not simply help with profits directly, whereas very low trade sizes result in bad revenue. The optimal trade size is shown to be at around 1 ETH and the worse at 100 ETH, with a loss of 5% on the investment [34].

Transaction Fee While transaction fees have a big impact on lower trading sizes, Boonpeam *et al.* indicate in [34] that they are not directly correlated with losses at medium to high trading sizes. Their analysis displays the most profits on *1inch* even though it has one of the highest fees.

Liquidity Finally, the liquidity of the pool interacted with is considered to be crucial for profits. A pool with low liquidity is connected with very high price slippage. It is, therefore, important to find a ”sweet spot” by having a trade size big enough to not be significantly impacted by fees but small enough to not cause excessive price slippage while trading [34].

3.3 Comparison and Discussion

Despite arbitrage being a well-researched topic in finance, there remains a paucity of research on decentralized finance in particular. While many of the previous works, such as [31] and [30] investigated arbitrage opportunities, and [33] and [32], even implemented working arbitrage bots on CEXes, research on the topic of arbitrage in decentralized finance is still scarce. Out of the few researches found on DEX arbitrage, none executes arbitrage trades with a working implementation. Furthermore, no comparison between arbitrage opportunities on different blockchains has been made by any known research at the time of writing. Since it is still a very new area of finance, it leaves room for further investigation, part of which is contained in this work. The execution on DEXes and CEXes are both looked at in this thesis, with an emphasis on DEXes and different blockchains.

Paper	EX	Strategy	Currencies	Data	Execution
I. Makarov [31]	CEX	Spatial	XRP, BTC, ETH, USD, EUR	Historical December 2017 - February 2018	No
S. Bistardelli [32]	CEX	Spatial	BTC	Historical to train bot, real time to trade	Yes
R. Levus [30]	CEX	Cyclic & Spatial	Many Cryptocurrencies	Real time	No
E. Han [33]	CEX	Cyclic	ETH, DAI, USDT, MKR, BAND	Real time	Yes
Y. Wang [5]	DEX	Cyclic	Many Cryptocurrencies	Historical and real time	No
N. Boonpeam [34]	DEX	Cyclic & Spatial	Many Cryptocurrencies	Real time	No
This Work	DEX / CEX	Cyclic & Spatial	Many Cryptocurrencies	Real time	Yes

Table 3.1: Comparison Related Work

Chapter 4

Approach

This chapter presents the different arbitrage bots implemented. Details about the approach and strategy are discussed. Section 4.1 focuses on a spatial arbitrage bot working on centralized exchanges, while section 4.2 discusses spatial arbitrage bots implemented for decentralized exchanges. Finally, section 4.3 explains the approach taken to implement a cyclic arbitrage bot on the Pancakeswap DEX.

4.1 Spatial Centralized Exchange Bot

The arbitrage bot for centralized exchanges is designed in three main components, pair and exchange setup, arbitrage opportunity scanning, and trade execution. The approach is to start and define what pairs should be analyzed on which exchanges, then to scan through the specified pairs, and finally to execute a trade when a profitable opportunity is found. This design allows for an easy way to change the strategy if desired.

4.1.1 Strategy

A variety of different suitable exchanges for the arbitrage bot had to be evaluated in a first step. The focus was on having exchanges that are spread across the globe and at least having one from Asia, and Europe included since price differences are greater between european and asian markets [31]. Five exchanges were chosen to cover these requirements. *Binance*, *Huobi*, *Kucoin*, *Bitstamp*, and *Lykke*, which can be seen in Table 4.1.

To be able to perform arbitrage trades between these exchanges, the same cryptocurrency pairs have to be available to trade on at least two of the chosen exchanges. Most of the pairs available on the chosen exchanges include USDT as the base asset. Therefore USDT has been chosen as the starting token for all arbitrage trades. This means that every trade has to start and end with USDT. This is against the suggested better performance in arbitrage when starting from FIAT currencies (EUR, USD) by [31], but was necessary

Exchange	Founded	Country of HQ	Ranking	Fees
Binance	2017	Cayman Islands / Seychelles	1	0.1%
Huobi	2013	China	2	0.1%
Kucoin	2014	Seychelles	6	0.2%
Bitstamp	2013	UK	33	0.5%
Lykke	2015	UK	331	0.0%

Table 4.1: Centralized exchanges used for the CEX bot [4]

for a fair comparison between CEXes and DEXes later on. One example trade is presented in 4.1.

$$USDT =>^{Binance} ETH =>^{Huobi} USDT \tag{4.1}$$

The sequence of actions which are illustrated in Figure 4.1, presents the execution flow performed by the arbitrage bot. It begins with the creation of currency pairs by matching any of the available currencies with USDT to get a list of pairs that can be used at a later point by the bot (*cf.* 2, Figure 4.1). Then all the exchanges defined are initialized in order for the bot to be able to fetch information from them (*cf.* 3, Figure 4.1). Pairs are consecutively checked (*cf.* 4, Figure 4.1) on every defined exchange (*cf.* 5, Figure 4.1) by sequentially fetching the order book of two exchanges (*cf.* 6, Figure 4.1) and verifying whether there is an arbitrage opportunity between them.

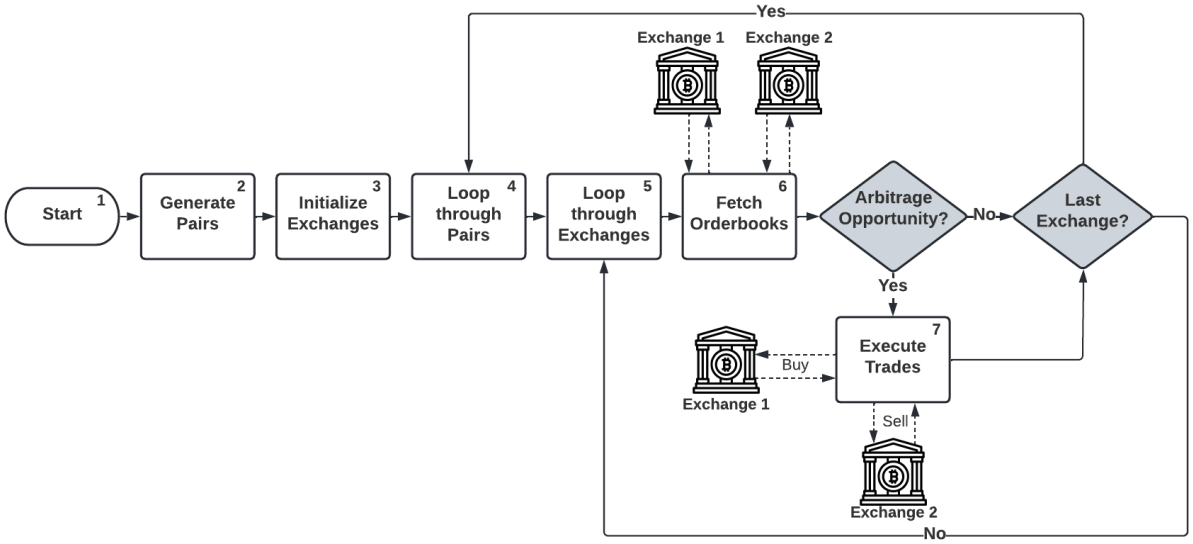

Figure 4.1: Sequence of actions performed by CEX arbitrage bot

In order to reliably determine prices of potential trades, the highest sell and lowest buy price available on the order book has to be found by identifying the amount n of necessary orders to cover the total trade. Consider the following example ask side of an order book for the pair BUSD/BNB on *Binance* presented in Figure 4.2 that is read bottom-up. At the lowest ask price of 217.4 BUSD/BNB it is only possible to buy an amount of 268.827

BNB. If the bot wants to buy 270 BNB, the higher second price from the bottom 217.5 BUSD/BNB has to be paid. The reverse procedure applies when selling BNB, where the bot seeks for the highest priced order that covers the complete trade. Therefore the amount n of orders necessary is used to determine the sell or buy price.

Price(BUSD)	Amount(BNB)	Total
219.0	120.045	26,289.85
218.9	181.618	39,756.18
218.8	960.421	210,140.11
218.7	269.212	58,876.66
218.6	597.118	130,529.99
218.5	35.115	7,672.63
218.4	13.939	3,044.28
218.3	18.871	4,119.54
218.2	55.371	12,081.95
218.1	26.257	5,726.65
218.0	158.490	34,550.82
217.9	295.599	64,411.02
217.8	33.378	7,269.73
217.7	139.034	30,267.70
217.6	154.229	33,560.23
217.5	166.338	36,178.51
217.4	268.827	58,442.99

Figure 4.2: Ask side of orderbook BNB/BUSD, taken from binance.com

To calculate n, each order is checked for its size, and as soon as enough orders are found that can be used to cover the total trade size, the price of the last order necessary is saved. The depth of the market plays an essential role in this step. If the market is deep, i.e. many orders are placed at prices close to the highest bid or lowest ask, the bot does not have to commit to a low bid or high ask price in order to find a price at which the order can be filled immediately. This strategy of determining the sell or buy price, results in the most reliable prices and the smallest deviations from the final execution price. It also increases the chance for the order to be filled immediately and the expected arbitrage to be successful.

After both the ask and bid price are determined, fees are subtracted from the result to calculate the actual trading price. If the bot wants to execute a trade, the fees of each exchange have to be covered. Every CEX has a unique trading fee which ranges substantially between CEXes, which are presented in Table 4.1.

If an opportunity above a defined threshold is found, the following trades need to be executed (*cf.* 7, Figure 4.1). At an initial stage, the amount B_U USDT has to be traded to B_A token A on exchange 1. Then B_A token A has to be traded to S_U USDT on exchange 2. This yields a profit if the bid price P_{b1} on exchange 1 is smaller than the ask price P_{a2} on exchange 2, visible in rule 4.2.

$$P_{b1} < P_{a2} <=> B_U < S_U \tag{4.2}$$

This trade increases the total portfolio of token A over both exchanges by ΔI calculated in 4.3.

$$\Delta I = S_U - B_U \tag{4.3}$$

4.1.2 Challenges

The main challenge with centralized exchanges is that the funds are not held in a wallet controlled by the user, but spread across wallets defined and controlled by the CEX. These wallets cannot be used on several exchanges like in DeFi, but rather only present the users with available funds that can be withdrawn or spent. All swaps and trades happen exchange internally, and it is not necessary for the exchange to actually move funds from one wallet to another. This makes it impossible to use the same wallet on different exchanges, which leaves two possibilities when trying to execute inter-exchange arbitrage trades.

1. Buy the desired asset for a cheap price on the exchange with the lower price, transfer it from one exchange to the other, and finally sell the asset on the more expensive exchange.
2. Hold the currencies of the pair on both exchanges, sell it on the more expensive exchange and buy it on the cheaper one.

Neither of these options is optimal, however the second option offers two important advantages and therefore is more suitable for the implementation of the bot.

Fees When making many transactions like in the first option, it is certain that a large amount of fees has to be paid every time assets have to be transferred.

Time The larger problem is the transaction time of each currency. While some can be sent from one exchange to another in a matter of seconds, there are still many currencies that take several minutes or hours to be sent, and by the time the money arrives on the second exchange, the opportunity is gone.

Thus, holding all assets considered for arbitrage on every observed exchange remains the only option. The problem with this approach is that one has a number of currencies and exchanges, and distributing the investment evenly and efficiently can be problematic. The variety of assets that can be considered is also drastically reduced when one does not have a large budget available. Nevertheless, the second option was adopted for the bot since it allows for the execution of arbitrage trades.

Another challenge was the correct interpretation of the data returned by the CEX. Prices, especially on smaller exchanges with a large spread, can be very arbitrary when looking at the last traded price. On small exchanges, there is the possibility that the last trade was filled a long time ago, and therefore the current price deviates from the actual market price. In case an order is placed with this price, it would not be filled and is therefore useless for arbitrage. The best way to resolve this problem, as explained in section 4.1.1,

was to calculate the highest/lowest price of the last bids/asks until the order size was big enough to fill the arbitrage trade. Using this price, the trade was almost certain to be filled immediately.

Furthermore, arbitrage on CEXes does not allow for atomicity when executing the two trades necessary for arbitrage. This increases the risk of losses since one trade can be executed while the other stays unfilled in the order book. In this case, the already executed trade has to be manually reverted since it does not provide any benefit.

4.2 Spatial Decentralized Exchange Bot

The spatial arbitrage bot on decentralized exchanges tries to find price discrepancies between two different decentralized exchanges. Interactions with decentralized exchanges differ greatly from those with centralized exchanges. Improvements can be made over the CEX bot by using tools only available in decentralized finance, like for example, the atomic execution of trades using smart contracts. In the following the precise approach taken to implement the spatial DEX bot is presented and explained.

4.2.1 Strategy

Two important decisions had to be taken before implementing the bot on decentralized exchanges. Firstly, it had to be decided on which blockchains the bot would be operating. Secondly, whether the bot should work between blockchains or only on one had to be set. In both spatial and cyclic arbitrage, the decision taken was that the bots implemented should operate on one blockchain at a time only. The reason being that when trying to implement inter-blockchain arbitrage bots, bridging between the different chains is often slow and problematic to automate. The spatial arbitrage bot was built for the Ethereum blockchain (ETH) as well as the Binance Smart Chain (BSC). A third blockchain, Avalanche (AVA), was also used to deploy the bot, but it is not covered in the thesis text due to the strong similarity to BSC. Having three implementations available later on would be important to evaluate differences in performance along with general functionality on the different chains.

On both blockchains (ETH and BSC), two exchanges were chosen for price comparison. The compared exchanges were *Uniswap* and *Sushiswap* on ETH as well as *Pancakeswap* and *Biswap* on BSC, which is visible in Figure 4.3. All four of these exchanges are based on very similar smart contracts in terms of swaps and information retrieval about the current states of liquidity pools. Due to this fact, a general way of interacting with the DEXes could be implemented and used throughout the different exchanges. Since the strategy proposed works identically on both versions of the arbitrage bot, the ETH version will be used as an example to elaborate on the strategy implemented and the way this bot scans the market for arbitrage opportunities and executes found trades.

The arbitrage bot is split into two main components, namely the arbitrage detection tool, running constantly, and the smart contract deployed on the blockchain that is responsible

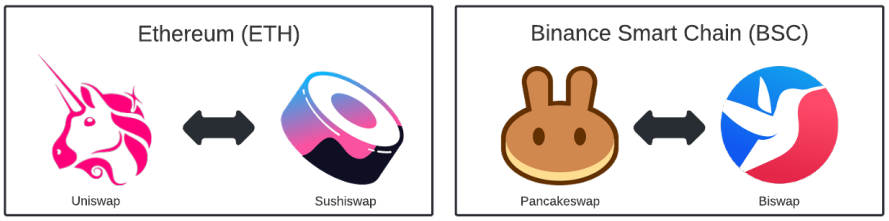

Figure 4.3: Exchanges compared on ETH and BSC

for an atomic execution of the trades. The arbitrage detection tool continually tries to estimate the outcome of potential arbitrage trades. As soon as a profitable trade is found, it calls the smart contract that executes the trades and checks whether a profit was made.

Arbitrage Detection

The sequence of actions performed by the bot is presented in Figure 4.4. To start the process, a predefined list of token pairs is analyzed, and the factory contract of both exchanges is called to find the address of the token pairs' liquidity pools for Uniswap and Sushiswap (*cf.* 2, Figure 4.4). Next, the current gas price is fetched from the blockchain explorer visible in step 3 in Figure 4.4. For each pair that has to be checked, the following steps are performed, presented as a loop starting at step 4 in Figure 4.4.

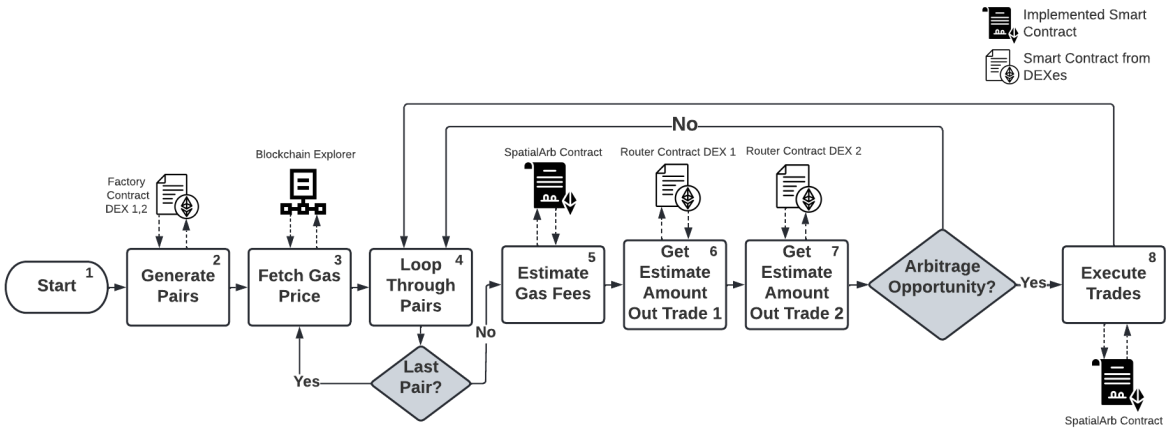

Figure 4.4: Sequence of actions performed by spatial DEX bot

The gas fee for a potential trade is predicted by estimating the cost of a function call in the `SpatialArb` contract that is deployed on the blockchain. The result of this estimation multiplied by the current gas price results in the total transaction cost calculated in step 5 in Figure 4.4. After calculating the transaction cost, a potential trade is analyzed to get the estimated amount returned from the trade, using the following steps:

1. Estimate the amount returned when trading amount I_A token A on Uniswap (*cf.* 6, Figure 4.4).

2. Take the output amount O_B of token B and estimate the amount returned when trading O_B token B back to O_A token A on Sushiswap (*cf.* 7, Figure 4.4).

$$O_A - I_A - transactionCost > 0 \tag{4.4}$$

If the inequality 4.4 holds, profit can be made, and the smart contract is called to execute the trades (*cf.* 8, Figure 4.4).

Arbitrage Execution

The arbitrage execution function is implemented on a smart contract that is deployed on the blockchain before execution. The function responsible for trading, executes the following steps atomically as proposed by [5]. This means that if one of the steps fails, all other steps are reverted. Therefore making it a safe way to try out arbitrage opportunities without losing any money in case of a failed transaction or a mistake made by the detection tool. The steps executed, as can be seen in Figure 4.5 are:

1. Assess the current amount of token A on the contract (*cf.* 2, Figure 4.5)
2. Swap an amount of token A on Uniswap for token B (*cf.* 3, Figure 4.5)
3. Check the current amount of token B on the contract
4. Swap amount returned for token B on Sushiswap (*cf.* 4, Figure 4.5)
5. Check if the amount of token A after the last swap is higher than the initial amount of token A (*cf.* 5, Figure 4.5), if that is not the case, revert all steps.

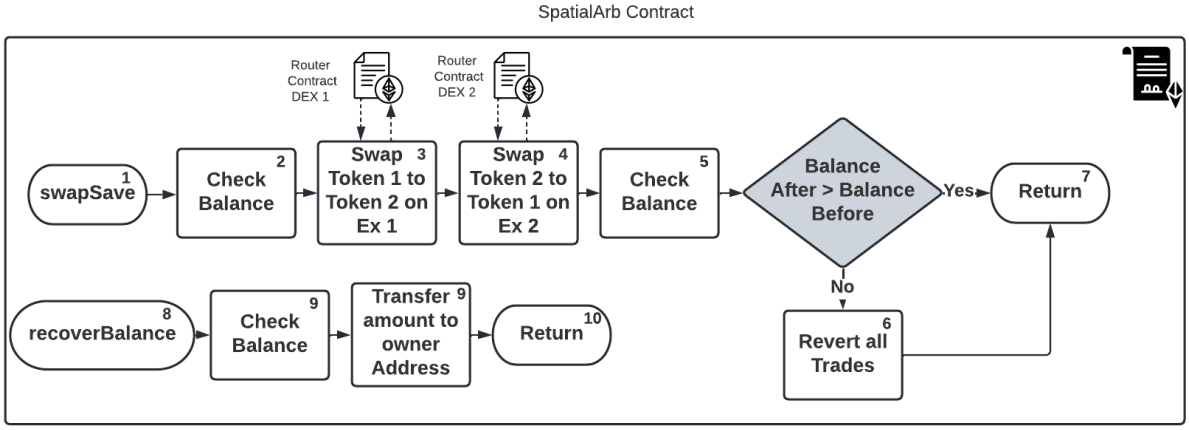

Figure 4.5: Sequence of actions performed by Smart Contract for spatial arbitrage on DEXes

Step 5. is to make sure that in case no profit was made due to price changes during the execution or wrong arbitrage prediction, the trades are reverted (*cf.* 6, Figure 4.5) and no money is lost (except for the gas fees).

Since the assets are traded from the smart contract there needs to be a way to retrieve the assets. A function that is implemented on the smart contract makes sure that when called by the contract owner, it transfers all of the funds available to the owner's address (*cf.* 9, Figure 4.5).

4.2.2 Challenges

When trading on decentralized exchanges, the user alters the state of the blockchain. This results in gas fees that have to be paid for each transaction. The price differences used for arbitrage trades are very small and can definitely be reduced or even completely nullified by gas fees. Therefore, a crucial step when estimating the arbitrage opportunity is to calculate the near to exact transaction cost. When overestimating the transaction cost it is highly probable, that potential arbitrage opportunities will never be executed. Underestimating the cost, could trigger many arbitrage trades that end up reverting due to no profit, and cost the user a considerable amount in gas fees.

Therefore these two points had to be optimized. Firstly, the gas fee in general had to be kept as low as possible, this included keeping the actions performed on the blockchain as small as possible by reducing the number of steps in the smart contract to a minimum. This resulted in the idea of having a server for arbitrage detection and a contract only for the execution in case of a near certain opportunity. Secondly the fee estimation had to be implemented to be as close as possible to the actual fee. Ideas of using fixed or mean fee costs had to be discarded early on in the process because they turned out to be very inaccurate.

Finally, an optimal gas estimation was achieved by using the estimation function made available by the node provider. It simulates the call of a smart contract and returns the amount of gas that was used by the simulation, which is an approximation of the real gas needed for the smart contract call. The estimation function, can only simulate successfully, when the smart contract call it is estimating is successful when executed. This made it impossible to use on the actual trading function in the smart contract created in section 4.2.1 since it would never be able to estimate the gas amount when no profit is made. Therefore another function is implemented on the SC that does not include the safety `require` statement to ensure that a profit was made. This function could be simulated with the estimation function, since it always succeeds even if it would perform a non-profitable trade. This second "unsafe" swap function is only used to estimate the gas amount. Therefore there is no danger of losing any money from it. Moreover, the difference between the two functions should be small enough to still get an accurate gas estimation outcome.

4.3 Cyclic Decentralized Exchange Bot

Cyclic arbitrage, as mentioned in section 2.5, focuses on discovering different paths through pairs on an exchange to find price disparities. The exchange chosen for this type of arbitrage bot was Pancakeswap on the BSC. With BSC's low gas fees and fast execution times, it presents a higher chance of finding exploitable cyclic arbitrage trades than ETH.

4.3.1 Strategy

Similar to the spatial arbitrage bot in section 4.2, the main functionality was split into two parts; the arbitrage detection tool and the arbitrage execution. The main difference between the two bots lays in the strategy used in the arbitrage detection server, as well as the change of execution flow in the smart contract.

Arbitrage Detection

The sequence of actions performed by the cyclic arbitrage bot is depicted in Figure 4.6. It starts with detecting arbitrage opportunities by iterating through a list of tokens available on the BSC. Through this process, pairs are created and later checked for the availability of liquidity pools on Pancakeswap (*cf.* 2, Figure 4.6). To find cycles, the approach was to create a graph where each token represents a node, and each LP found between two tokens is saved as an edge between two nodes. After creating a graph, it can be searched for closed walks, in other words routes through the graph that start and end at the same point (*cf.* 3, Figure 4.6). This results in a list of every combination of LPs the bot can utilize to perform cyclic arbitrage trades.

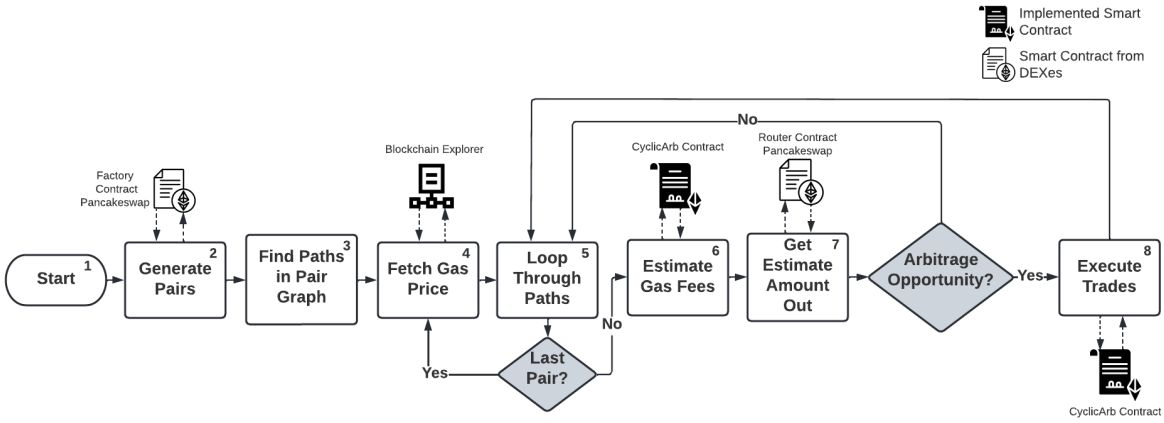

Figure 4.6: Sequence of actions performed by Cyclic DEX bot

The only walks considered are the ones that start and end with WBNB, the wrapped version of the BSC's native BNB since it is the most used token on Pancakeswap. Another advantage of considering only one token as the start and endpoint of cyclic trades is that only this token has to be held on the smart contract executing the trades, as is discussed

in section 4.3.2. Furthermore, any preparation swaps to the starting token of the cycle are unnecessary since they all start with WBNB. Figure 4.7 shows a potential setup of tokens on Pancakeswap with one possible cyclic arbitrage trade highlighted.

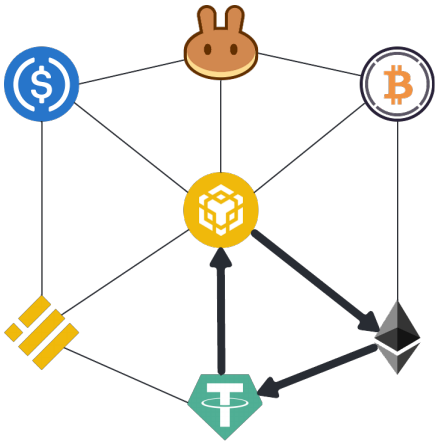

Figure 4.7: Example graph showing pools available between tokens, with one possible cyclic trade

Following the exploration of the graph and finding all available cycles, the bot fetches the current gas price P_G from the blockchain explorer in step 4 in Figure 4.6 and starts to iterate through all found paths in step 5, executing the following operations for each of them:

1. Estimate gas fees (*cf.* 6, Figure 4.6)
2. Calculate transaction cost C_T
3. Estimate the amount O_B of WBNB returned when trading a certain amount I_B through the route (*cf.* 7, Figure 4.6)
4. Calculate arbitrage opportunity

$$O_B - I_B - C_T > 0 \tag{4.5}$$

If inequality 4.5 holds, a profitable cyclic arbitrage trade has been found, and the server can call the smart contract to execute it (*cf.* 8, Figure 4.6), otherwise the next path is checked.

Arbitrage Execution

Similar to the spatial DEX bot approach in section 4.2, the execution of the trades is implemented on a smart contract deployed on the blockchain. Apart from small changes, the idea and execution of trades are the same.

The steps executed are:

1. Assess the current amount of WBNB on the contract (*cf.* 2, Figure 4.8).
2. Perform all trades in the route on Pancakeswap (*cf.* 3, Figure 4.8).
3. Asses the final amount of WBNB on the contract (*cf.* 4, Figure 4.8).
4. Check if the current amount of WBNB is larger than the amount at the start, if that is not the case, revert all previous steps.

Step 4 is to ensure that in case no profit was made due to price changes during the execution or wrong arbitrage prediction, the trades are reverted, and no money is lost (except for the gas fees).

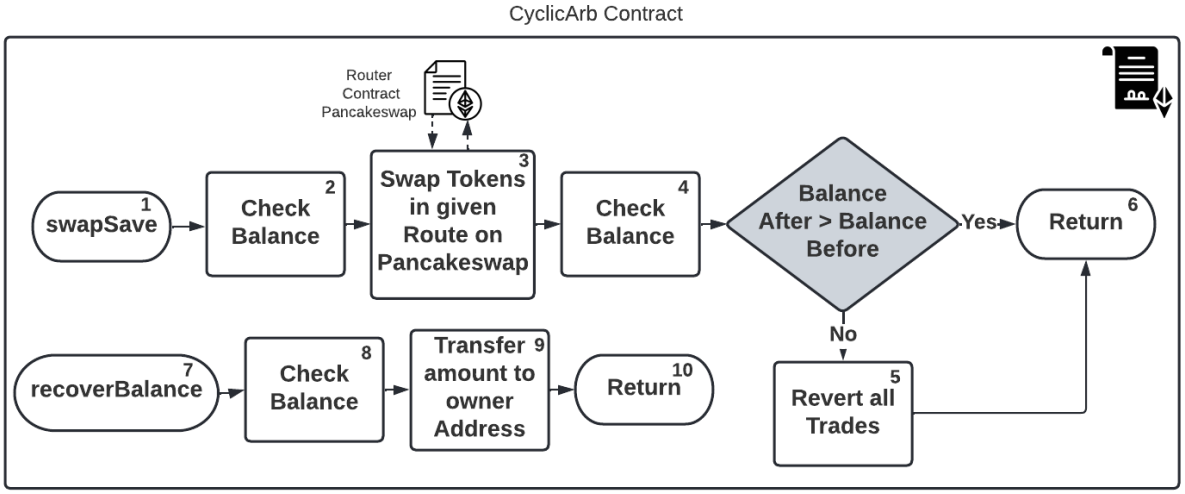

Figure 4.8: Sequence of actions performed by the smart contract for cyclic arbitrage execution on DEXes

4.3.2 Challenges

Since the general structure and environment are the same between the cyclic and spatial arbitrage DEX bot 4.2 the challenges faced were similar, apart from small differences in the setup phase. Challenges concerning gas fees were already discussed in section 4.2.2. The main difference in the setup between spatial and cyclic arbitrage was finding the paths or cycles. Since technically a nearly infinite amount of paths can be explored in the graph considered in section 4.3.1, the path length had to be fixed to save resources. With a path length of three, about 28 cycles could be determined, which was considered a large enough number of cycles to check and was also the route length mostly seen on successful arbitrage trades in [5].

Chapter 5

Implementation

The following chapter covers the implementation process of all arbitrage bots developed for this thesis. First, the technologies used and decisions taken in the centralized exchange bots are discussed in section 5.1. Followed by a deeper dive into the implementation of the decentralized exchanges bots in section 5.2.

5.1 Centralized Exchange Bot

The arbitrage bot for centralized exchanges was implemented in Python using CCXT, a "Cryptocurrency exchange Trading Library", that provides an abstraction layer on top of existing Application Programming Interfaces (APIs) with the support for many centralized exchanges around the world [35]. A visualization of the abstraction the CCXT library provides can be seen in Figure 5.1.

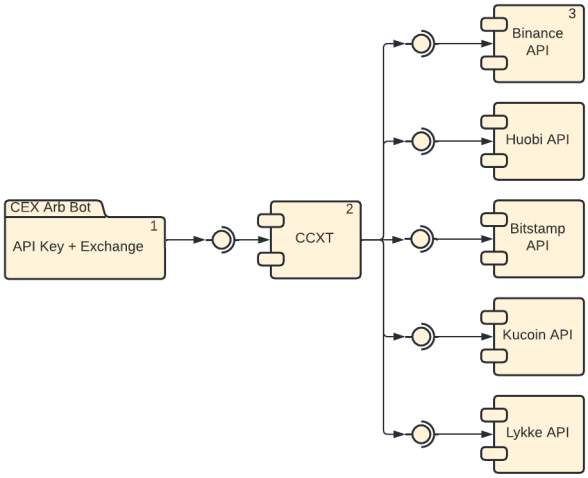

Figure 5.1: CCXT API abstraction

The library is useful when using exchanges with different API structures since it generalizes the API calls and encapsulates them. With support for more than 120 exchanges

and the possibility of writing functions for any additional exchange needed, there were no restrictions on the choice of exchanges. As discussed in section 2.4, on centralized exchanges it is necessary to complete a KYC process, after which an account can be opened and private API keys can be generated.

In a first step multiple pairs are generated between a base currency and a list of quote currencies that are decided upon before the start of the execution. Following, the arbitrage detection is prepared by connecting CCXT to the APIs of the different exchanges, using the private API keys provided by each exchange. This is done in CCXT by creating a new instance of an exchange object and passing API Key and Secret Key to the constructor. These exchange objects can be initialized by calling `load_markets`. When the exchange object is created, any operation available on CCXT, like `fetchOrderBook` or `create_limit_sell_order` can be called in the same way for every exchange. An example of the initialization can be seen in Listing 1. The private key and the secret are both stored as environment variables and can be accessed using the `os` library.

```
binance_access_key = os.environ.get('BINANCE_KEY_HOME')
binance_secret_key = os.environ.get('BINANCE_SECRET_HOME')
binance = ccxt.binance({
        'apiKey' : binance_access_key,
        'secret' : binance_secret_key,
        'enableRateLimit':True,
    })
binance.precisionMode = ccxt.DECIMAL_PLACES
binance.load_markets()
```

Listing 1: Initialization of exchange object

After importing all exchanges objects, the main `while` loop responsible for constantly fetching market conditions starts its execution. It checks every token pair for arbitrage opportunities across every exchange combination. Figure 5.2 shows the execution flow for one trading pair and one exchange pair of the `checkMarkets` function. In order to compare prices between the exchanges, the `fetchOrderbook` function is used to request the current order books from the two exchanges in step 1 in Figure 5.2. For the exchange where the asset will be bought, the ask orders have to be considered, while on the exchange where the asset will be sold, the bid orders are relevant (*cf.* 2, Figure 5.2). The function needs to loop through the orders until the amount that wants to be traded is covered by the accumulated size of orders found (*cf.* 3, Figure 5.2). Upon fulfillment of this requirement, the last necessary bid and ask prices to cover the trading size are the price at which the asset can instantly be bought or sold on the respective exchanges (*cf.* 4, Figure 5.2). If the calculated output, which derives from multiplying the desired trading amount by the buy or sell price and subtracting the fees of the used exchange, results in a positive arbitrage opportunity, the orders for the two calculated exchange rates are executed, otherwise `False` is returned and no order will be executed (cf 7, Figure 5.2).

The code for the `executeOrder` function is presented in Listing 2. The creation of a limit order with CCXT requires no more than one function call with three arguments. It is also

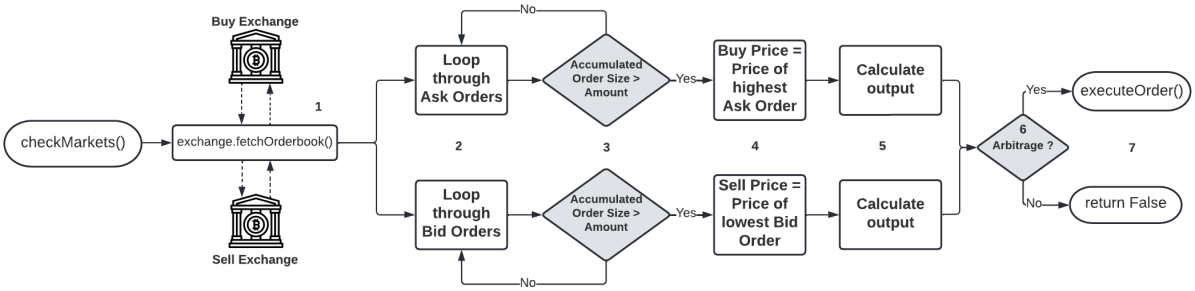

Figure 5.2: Execution flow of `checkMarkets` function

visible that the abstraction is equal, for whichever exchange is used. On line 3 a buy order is created that buys a specified amount of an asset at the price discovered in the process explained above. Line 4 creates a sell order on the second exchange that sells the same asset for the discovered price. The sell amount is determined by the amount returned by the first trade.

```
1 def executeOrder(exchange1,exchange2,buyPrice,sellPrice,sellAmount,symbol):
2     exchange1.create_limit_buy_order(symbol,TRADE_AMOUNT,buyPrice)
3     exchange2.create_limit_sell_order(symbol,sellAmount,sellPrice)
```

Listing 2: Order execution function

5.2 Decentralized Exchange Bot

Due to the similarity of technologies and implementation used in the two decentralized exchange bots (cyclic and spatial), the following explanation focuses on the spatial arbitrage bot and cover most of the technologies. In section 5.2.2, the few differences of the cyclic arbitrage bot will be covered.

Since both arbitrage bots need to interact with blockchains, a web3 library is needed. These are libraries that enable the interaction between programs and blockchains. For this purpose the ether.js library was used, which provides the user with any functionality needed to read and write to the blockchain. It also covers critical functions like keeping private keys safe and signing transactions when necessary [36]. When using these functions to communicate with a blockchain, a provider is needed. These are abstractions of a connection to the blockchain that provide an interface to node functionalities, through which the status of the blockchain can be fetched [36]. Additionally, these providers make sure the transactions called by the ethers library can be sent to alter the state of the blockchain. For each blockchain, a different provider is necessary that offers a node to interact with.

The functionality of the spatial arbitrage bot on DEXes is explained on the example of the Ethereum network. On BSC the bot works equivalently and the implementation

is identical. This is due to the fact that BSC and ETH are both EVM blockchains as discussed in section 2.3.3. Furthermore it should be noted that the smart contracts (cyclicSwap.sol and spatialSwap.sol) built for these bots, have to be deployed on the blockchain in order for this bot to work. In the following explanation the assumption is made that these two contracts are already deployed. All other smart contracts mentioned are already deployed by the DEXes.

5.2.1 Spatial Arbitrage Bot

The spatial arbitrage bot consists of two main parts; arbitrage detection and arbitrage execution. The arbitrage detection is implemented as a Node.js server and written in Typescript. It consists of a core file SpatialArb.ts and a file Fetch.ts to keep all blockchain interaction and requests separate from the main logic of the bot. The spatial DEX bot execution starts with a call to an initialization function that is responsible for preparing all addresses needed later in the code execution by calling the `fillPairContracts` function on every pair and exchange that has to be checked for arbitrage opportunities.

```
export const fillPairContracts = async (
  exRouter: string,
  exFactory: string,
  route: Token[],
  provider: any
) => {
  const factoryContract = new ethers.Contract(
    exFactory,
    [
      "function getPair(address tokenA, address tokenB)
      external view returns (address pair)",
    ],
    provider
  );

  const key: string = `${exRouter + route[0].address + route[1].address}`;

  paircontracts[key] = await factoryContract.getPair(
    route[0].address,
    route[1].address
  );
};
```

Listing 3: Function to fetch addresses of a pair contract between two tokens

Listing 3 shows a simplified version of the `fillPairContracts` function's implementation found in the Fetch.ts file of the projects GitHub repository [37]. It is responsible for filling

a dictionary with pair contract addresses. The pair contract is the contract of a liquidity pool between the two tokens on a certain exchange and is needed to fetch prices between two tokens.

In order to save a considerable amount of resources and computing power, the dictionary is filled at the beginning of execution which means that no pair contract addresses have to be fetched for the rest of the execution. The router contract and factory contract address of the DEX have to be passed to this function together with the route, which contains the two tokens that need to be traded, and the provider which is required for every blockchain interaction. Between lines 7 and 14 in Listing 3, an ethers contract object is created, through which a contract function on the blockchain can be called. To create a contract object, the address, the application binary interface (ABI), and the provider have to be passed to the contract object's constructor. The factory contract of the DEX can then be called through the contract object in line 18 in Listing 3 with the two tokens as arguments. What should be returned is the address of the LP with the two provided tokens inside. This address is saved in a dictionary that can always be accessed during execution.

```
1  const checkMarket = async (...) => {
2    const gasCost = await estimateSpatialSCGas(...);
3
4    const trade0 =
5      await fetchTrade(ex1.Router, ex1.Factory, route, amount, provider);
6
7    const minAmountOut0 = trade0[0][1];
8    const trade0token0amount = trade0[1];
9    const trade0token1amount = trade0[2];
10
11   const revRoute = [...route].reverse();
12
13   const trade1 =
14     await fetchTrade(ex2.Router, ex2.Factory, revRoute, minAmountOut0, provider);
15   const minAmountOut1 = trade1[0][1];
16   const trade1token0amount = trade1[1];
17   const trade1token1amount = trade1[2];
18
19   checkOp(swap);
20 };
```

Listing 4: Function that checks for arbitrage opportunities

After fetching the current gas price on the blockchain, the arbitrage detection starts with the `checkPairs` function, that calls `checkMarket` for every pair and exchange provided. First, the momentary gas price is fetched from the provider. In the `checkMarket` function for which a simplified version of the real implementation available on the GitHub repository [37] is depicted in Listing 4, the gas cost is calculated, by calling `estimateS-`

`patialSCGas`. This function takes all necessary arguments for a swap and calls the `estimateGas` method on the contract's function `swapNormal` in line 23 in Listing 4, in an attempt to estimate the gas needed for the execution of the trades. This returns an approximation of the amount of gas that will be used in case of trade execution.

When multiplied by the gas price, a total transaction cost estimation can be computed. Next the estimated amount returned from a first swap on exchange 1 is fetched by calling the `fetchTrade` function. A sequence diagram of the `fetchTrade` function is presented in Figure 5.3. It can be observed that this function calls the `getAmountsOut` function on the router contract of the exchange provided. The provider is only necessary as a connection between the Node.js server and the blockchain. The router contract of the exchange then calls `getReserves` on the pair contract of the LP between the two tokens that will be traded. The pair contract needs the reserves of the LP to calculate the price of each token as explained in section 2.3.1 and estimate the amount of tokens the user will get out of the trade.

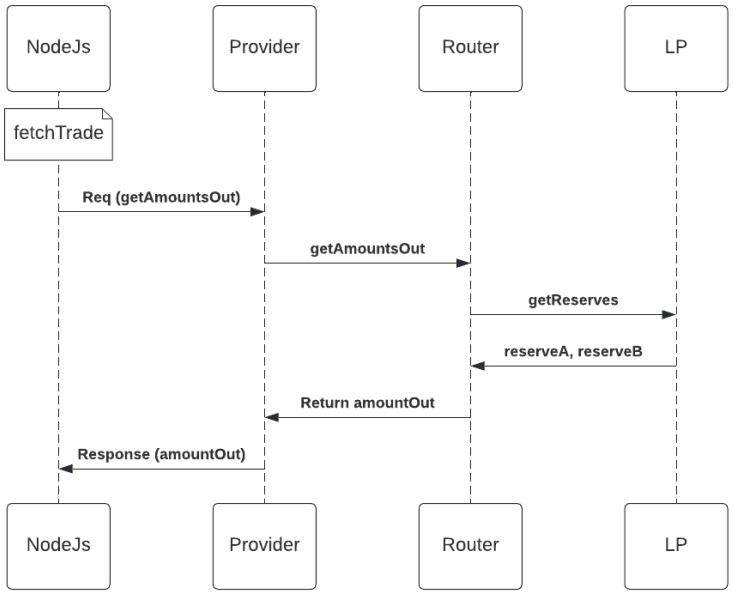

Figure 5.3: Sequence diagram of `fetchTrade` function to fetch amount returned from trade

The router contract of exchange 1 also subtracts LP fees before returning the final amount that the user would get back in case of the execution of this trade. This is passed to a second `fetchTrade` call for the inverse swap direction on Exchange 2.

All of the data gotten from these function calls are passed on to the `checkOp` function in the form of a `swap` object, which uses all of the data and calculates if the combination of trades is a potentially profitable arbitrage. The calculation to get the arbitrage opportunity in % is the following:

$$opportunity = \frac{amountOut - amountIn - transactionCost}{amountIn} * 100 \tag{5.1}$$

If the opportunity is larger than the fixed threshold defined at runtime, the current arbitrage trade is attempted to be executed. The `executeSpatialSwap` function is responsible for the execution of profitable arbitrage trades. In Figure 5.4 a sequence diagram, explaining the different calls necessary to execute an arbitrage trade, is presented. Since all the calculations have already been done at this point, that `executeSpatialSwap` simply calls the `swapSave` function on the smart contract through the provider. The arguments needed by the smart contract to execute the trade are: address of token1 and token2, address of the router contract of exchange1 and exchange2.

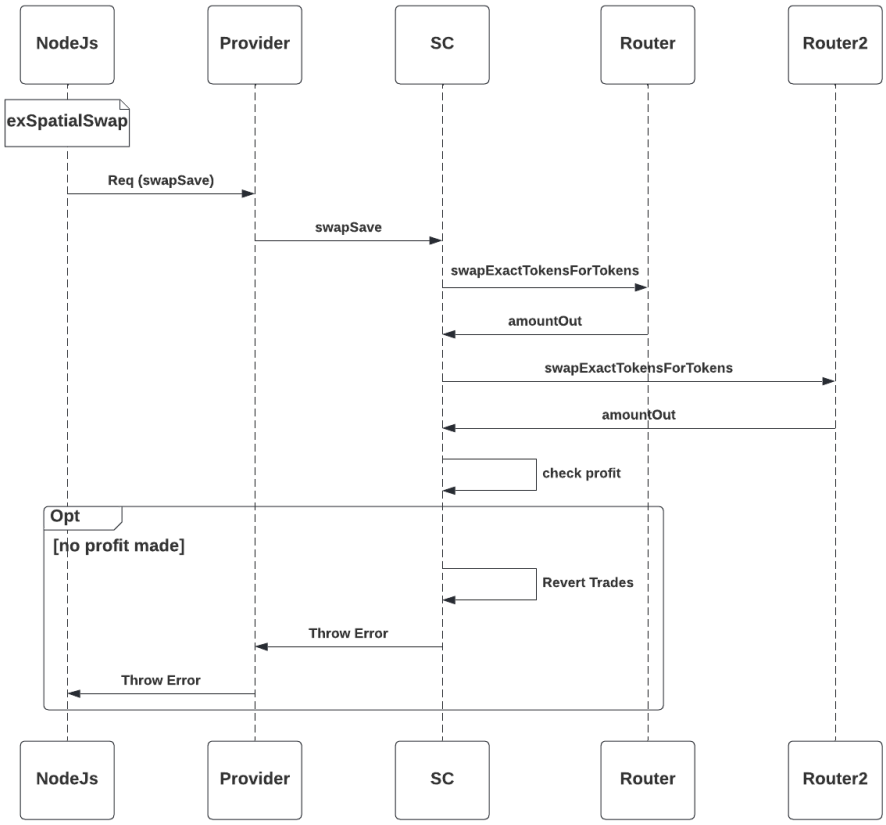

Figure 5.4: Sequence diagram of `executeSpatialSwap` function to execute arbitrage trades

The `swapSave` function visible in Listing 5 consists of two `swap` calls that are responsible for trading tokenIn to tokenOut on Exchange 1 and tokenOut to tokenIn on Exchange 2. The `swap` function is implemented as a private function that trades one token to another on a specified exchange. To know how much can be swapped back after the first trade, the `balanceOf` function of the ERC20 interface is called in line 11, which returns the balance of the specified token after the first trade. The returned amount is used for the second swap that has to be performed. After both trades are executed, the `balanceOf` function is called again to be used in the final `require` statement in line 17, that compares the start balance with the end balance. In case the statement in the `require` is false, all executed actions and trades in this function are reverted and the initial state of the blockchain is restored.

The spatialSwap contract is an ownable contract as can be seen in line 1 of the code

example in Listing 5. This annotation allows for functions that can only be executed by the owner of the contract (the wallet address that deployed the contract). One example for such a function is `swapSave` in line 3. If this was not an "owner only" function, anyone with the smart contract address could execute trades on the contract.

```
1  contract spatialSwap is Ownable{
2
3      function swapSave(address _tokenIn, address _tokenOut, address ex1, address ex2,
4      uint256 _amountIn) external onlyOwner {
5
6          uint256 balance1BeforeTransfer = IERC20(_tokenIn).balanceOf(address(this));
7          uint256 balance2BeforeTransfer = IERC20(_tokenOut).balanceOf(address(this));
8
9          swap(ex1,_tokenIn,_tokenOut,_amountIn);
10
11         uint256 tradableAmount = IERC20(_tokenOut).balanceOf(address(this)) -
12         balance2BeforeTransfer;
13
14         swap(ex2,_tokenOut,_tokenIn,tradableAmount);
15         uint endBalance = IERC20(_tokenIn).balanceOf(address(this));
16
17         require(endBalance>balance1BeforeTransfer,"No Profit Made, Reverting Trade");
18     }
```

Listing 5: Smart contract function to execute arbitrage swaps

5.2.2 Cyclic Arbitrage Bot

The implementation of the cyclic arbitrage bot is at its core very similar to the one of the spatial arbitrage bot. The technologies used and the flow of execution are the same. Following, the three main differences are presented in order of code execution.

Route finder

Before the arbitrage detection begins in the CyclicArb.ts file, as in the spatial arbitrage, an initialization function is called. The difference is that the function does not start with a predefined token pair list, but generates routes by calling the `findRoutes` function. It starts with a list of tokens, `findRoutes` iterates through every token, and checks if there is a liquidity pool between it and any of the other tokens on Pancakeswap. This is represented as step A in Figure 5.5, and it is checked with the condition presented in Listing 6. If the condition is met, a pool exists and an edge can be added to the graph between the two tokens. This is done for every token pair creating a graph, where nodes are tokens and edges represent liquidity pools. Then an algorithm explores the graph

and finds all closed walks of size 3 available (step B, Figure 5.5) and finally returns the found routes. It was decided that this resource-intensive task is only executed once in the beginning of the arbitrage detection. By storing the results in a dictionary, time could be saved when accessing routes during the bot's execution.

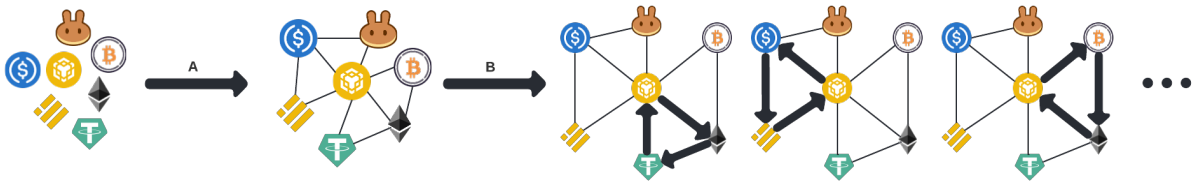

Figure 5.5: Visual representation of `findRoutes` function

```
if ((await fetchPairContracts([token1, token2], provider)) != NULL_ADDRESS){
    graph[token1.symbol].push(token2);
}
```

Listing 6: Condition that checks if there is a pool between two tokens on the exchange

Fetching the Price

Instead of fetching the price on several exchanges, in cyclic arbitrage the `getAmountsOut` can be called one single time for the complete cycle. The reason being, that `getAmountsOut` accepts a list as an argument for a trading route and returns the result of all swapping through every token in the list.

Smart Contract

Similar to the `getAmountsOut` function that can take a list of tokens as an argument to trade through a cycle, the `swapExactTokensForTokens` function used in `swap` can also take a list of tokens to trade through as an argument. This is why the smart contract looks slightly different. Furthermore, instead of having to pass two exchanges, it is sufficient to pass one exchange, since the trade stays in the same DEX. The rest of the contract is identical to the SpatialArb smart contract in section 5.2.

```
function swapSave(address[] memory path, address router, uint256 _amountIn) external
onlyOwner{
    address _tokenIn = path[0];
    uint balanceBeforeTrades = IERC20(_tokenIn).balanceOf(address(this));
    swap(router,path,_amountIn);
    uint balanceAfterTrades = IERC20(_tokenIn).balanceOf(address(this));
    require(balanceAfterTrades>balanceBeforeTrades,
    "No profit made, reverting..");
}
```

Listing 7: `swapSave` function, excerpt from smart contract cyclicSwap.sol

Chapter 6

Evaluation

This chapter presents the results collected by the implemented arbitrage bots and evaluates them according to the questions raised in section 1. First, the results of the centralized exchange arbitrage bot are presented in section 6.1, then in section 6.2, the results of both strategies of the DEX arbitrage bots on different blockchains are shown.

During the execution of each arbitrage bot, logs were collected. Most of the figures in this chapter use a metric (arbitrage opportunity), which is calculated through the equation presented in 6.1. The *transactionCost* consists of different factors depending on where the trade is executed. On CEXes *transactionCost* is the trading fees specific to the exchange, multiplied by the amount traded. On DEXes *transactionCost* consists of gas fees multiplied by the gas price plus the liquidity provider fees specific to each DEX.

$$opportunity = \frac{amountOut - amountIn - transactionCost}{amountIn} * 100 \tag{6.1}$$

6.1 Centralized Exchanges

The centralized arbitrage bot presents very stable results over most pairs and exchanges, with a few exceptions on single pairs. The different Figures 6.1, 6.2, and 6.3 show the arbitrage opportunities present on the 4th of June 2022. The different exchanges that have been compared are *Binance*, *Huobi*, *Lykke*, *Bitstamp*, and *Kucoin*. The labels on the figures are in the following order: trading pair, buy exchange, sell exchange.

As can be seen, there are no arbitrage opportunities found by the bot in the measured time period. All of the pairs demonstrate very stable arbitrage opportunities lower than 0 which represent little to no price differences between the exchanges. In these examples, pairs with Bitcoin, Ethereum, and XRP are depicted. The arbitrage opportunity is negative due to the cost of executing the trade, combined with higher buy than sell price between the exchanges.

In contrast to the mostly stable results, Figure 6.4 clearly shows a trading pair, namely USDT/XRP, where arbitrage opportunities were present on several occasions. These are

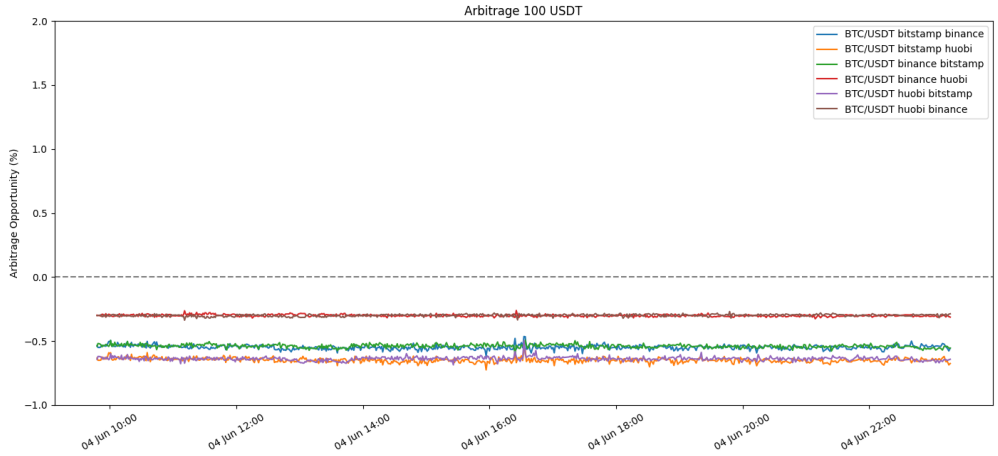

Figure 6.1: Arbitrage opportunities for pair USDT/BTC

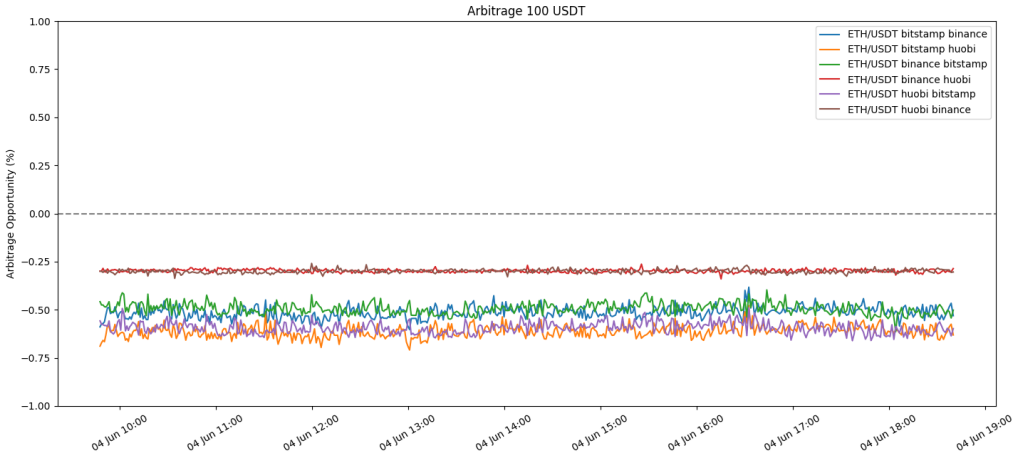

Figure 6.2: Arbitrage opportunities for pair USDT/ETH

very likely triggered by a high buy order on Bitstamp which results in a short-term price difference between several exchanges and Bitstamp. As can be seen in Figure 6.4, Lykke, Binance and Huobi, all show arbitrage opportunities compared with Bitstamp at the same time. While between other exchanges in Figure 6.3, it is clearly observable that no such arbitrage opportunities were present.

This can be due to the fact that the trading volume of the USDT/XRP pair is rather low on Bitstamp. If someone places a bid order with a price that is too high, the market reacts very quickly, and that order will be filled instantly. Since these fills are too fast for the implemented bot, as soon as a limit order had been placed by the bot, these orders were not visible anymore. Pairs that are traded a lot, have a very stable line on the arbitrage opportunity chart. An example is Bitcoin or Ethereum, which can be seen in Figures 6.1 and 6.2. The price difference on many different exchanges was constantly small. This led to a clearly visible result, and no arbitrage opportunity was found during that period.

These results show that in this case for a simple arbitrage bot, like the one built in this

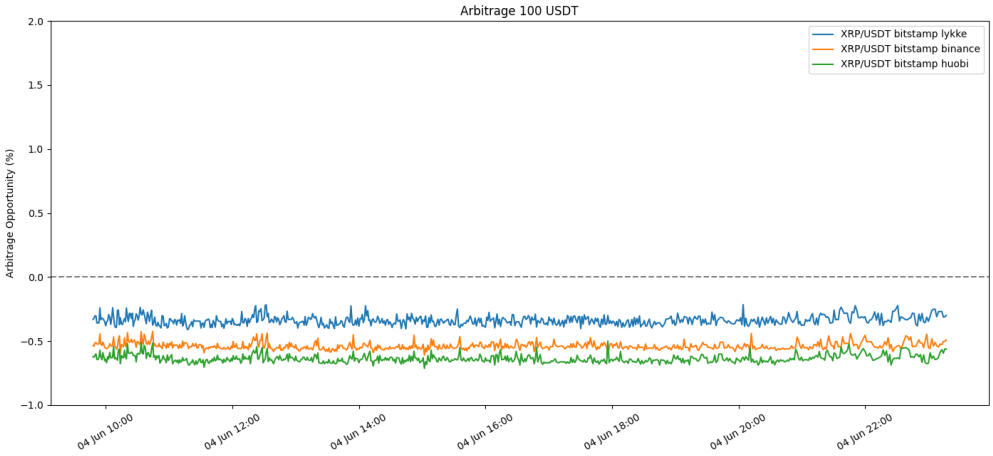

Figure 6.3: Arbitrage opportunities for pair USDT/XRP

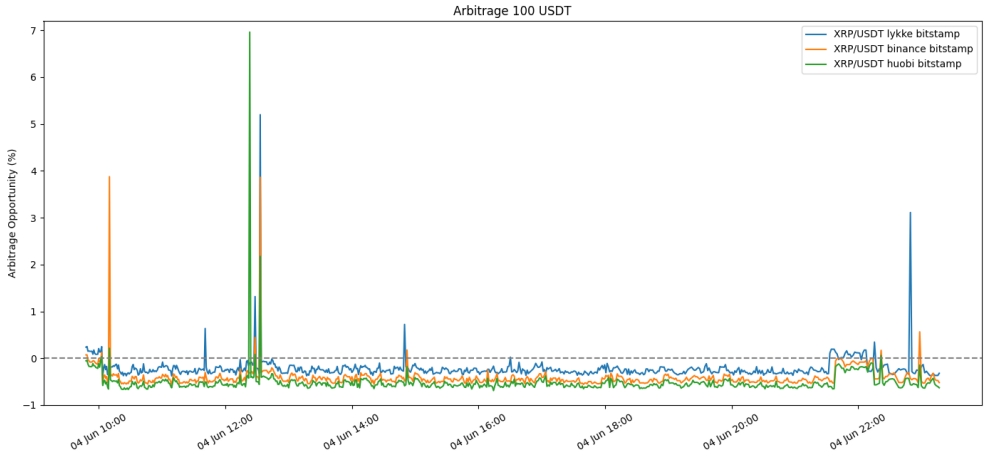

Figure 6.4: Arbitrage opportunities for pair USDT/XRP, selling on Bitstamp

experiment, taking advantage of price differences was not possible. Centralized exchanges ensure that prices stay synchronized throughout their markets and use highly efficient and zero latency equipment to immediately even out any possible inconsistencies. Even though in several works, successful arbitrage was performed on CEXes [31], [32], [33], it is difficult to take these experiments as a benchmark since they were all conducted years before this thesis when other market conditions were present. Furthermore, [31] did not perform any arbitrage and only analyzed price inconsistencies in the market. With the increased popularity of cryptocurrencies, the centralized exchanges also improved the stability of prices between each other. All the pairs compared on the different exchanges were:

- ETH/USDT
- BTC/USDT
- XRP/USDT
- XLM/USDT
- LTC/USDT
- SOL/USDT
- ADA/USDT

6.2 Decentralized Exchanges

The decentralized exchange bot often presented less stable arbitrage graphs. The arbitrage opportunities fluctuated from highly negative numbers to close to zero or positive values. It could be argued that this less stable arbitrage graph follows from two characteristics of DEXes namely; prices on DEXes only being synchronized through arbitrage bots and trades having a more direct impact on LPs, and therefore, the price of a pair on an exchange. The labels on the Figures presented in this section are in the following order: exchange 1, exchange 2, asset 1, asset 2. On exchange 1, asset 1 is traded to asset 2, and on exchange 2, asset 2 is traded back to asset 1.

6.2.1 Ethereum Blockchain

When looking at the arbitrage results from the bot on the Ethereum blockchain, it is well visible in Figure 6.5, which shows the arbitrage opportunites on the pair WETH/SUSHI without including gas fees, and Figure 6.6, that shows the same graph with gas fees included, that the current gas price has a very large impact on the availability of arbitrage opportunities on Ethereum. In Figure 6.5 price differences between assets, here SUSHI and WETH, are substantial when excluding gas fees. When looking at Figure 6.6 on the other hand, it can be observed that when including the sometimes incredibly high Ethereum gas fees to the calculation, the same opportunities vanish. This shows that price differences are present on the different exchanges, but the cost of exploiting them is too high in certain moments. It should be added, that the impact gas fees have on the opportunities is inversely proportional to the trade size. This arbitrage bot, operates with a rather small trade size of 0.2 WETH which increases the impact of gas fees on the arbitrage opportunity.

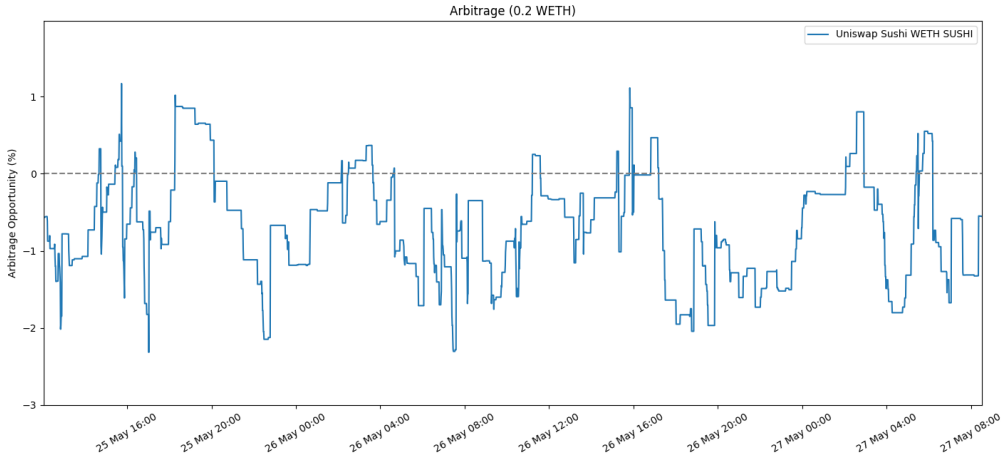

Figure 6.5: Arbitrage opportunities for pair WETH/SUSHI, gas fees excluded

The large differences between these two charts concerning the form and frequency, even though they show the same pairs and period, is due to the quickly changing gas prices. Gas prices change in every block, and therefore more often than the measured *amountOut* used for trade estimation. The bot tries out trades for the same pair every view blocks only. As a result, the graph in Figure 6.6 fluctuates more than the graph without gas fees included.

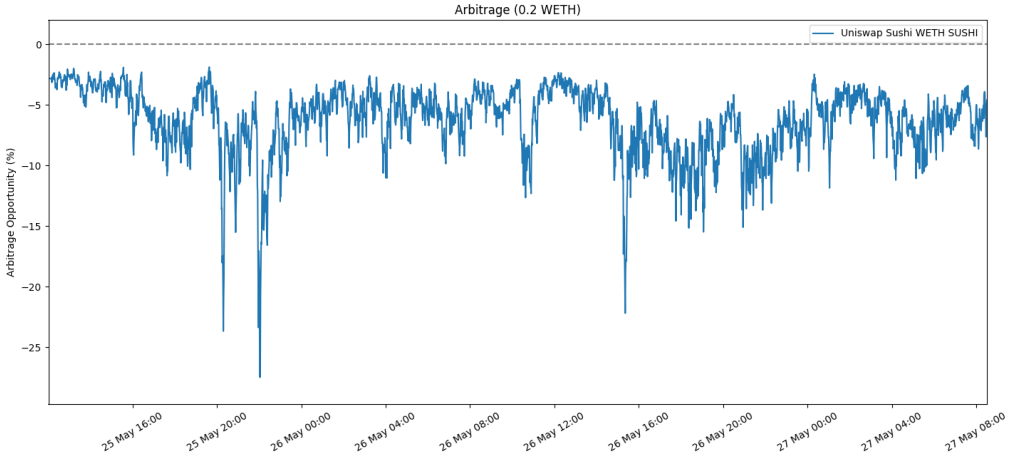

Figure 6.6: Arbitrage opportunities for pair WETH/SUSHI

The gas fees on the Ethereum network can be very diverse depending on the day and time, as seen in Figure 6.7, where the gas price development over the course of one week can be observed. On the y-axis, the price of one gas measured in Gwei (1 Gwei = 10^{-9} Ethereum) is depicted. The price of one gas is multiplied by the amount of gas needed for the transaction to get the total transaction cost. Visible on this chart is that the gas price changes drastically on an hourly basis. There were certain periods, nevertheless, where gas prices were low, which also impacted the arbitrage opportunities found during these periods.

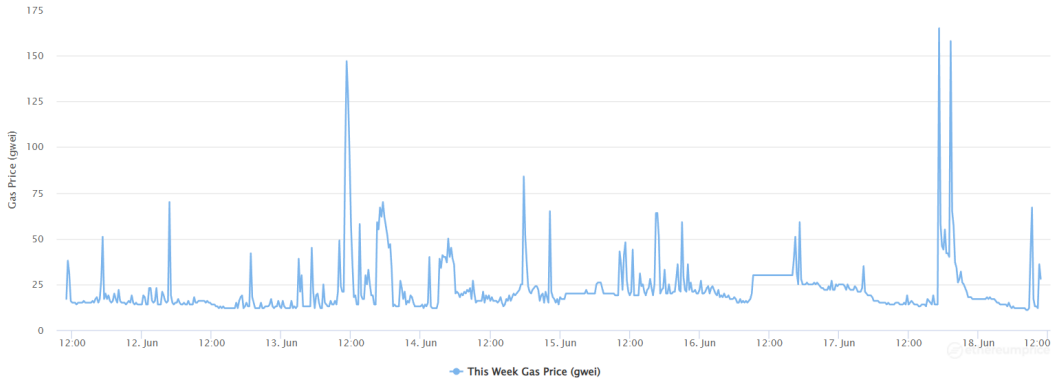

Figure 6.7: Gas prices on the Ethereum network between June 12 and June 18, taken from ethereumprice.org

Figure 6.8, where gas fees are included, shows exactly one of these moments. It is observable that during low gas price moments like the one shown in this Figure, the price differences are sometimes nearly large enough to yield a positive arbitrage opportunity. This chart shows the price difference between Sushiswap and Uniswap for the trading pair WETH/SUSHI. The arbitrage opportunity graph comes close to the 0 line on several occasions and even crosses it once. Even though this chart still doesn't show any usable opportunity, when comparing it to Figure 6.6, there is definitely a substantial improvement. As a remark, the relatively straight line between 25. May 00:00 and 25. May 02:00 is due to an API problem. During this time, the bot was not able to estimate trades.

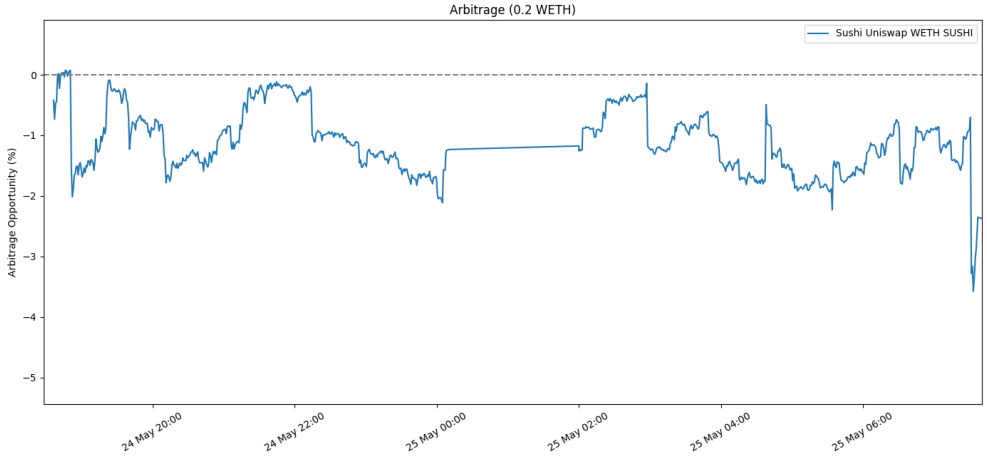

Figure 6.8: Arbitrage opportunities for pair WETH/SUSHI

In a final note about the arbitrage results on the Ethereum network presented in this chapter, it can be said that when trying to arbitrage with the trading size available for this work, there were no real arbitrage opportunities that could be exploited. The gas fees were too high even in moments of low traffic and gas prices on the Ethereum network. Several other charts not presented here showed similar or even worse results over more extended periods of time. Pairs observed and compared between *Uniswap* and *Sushiswap* included:

- WETH/USDC
- WETH/WTBC
- WETH/FRAX
- WETH/SUSHI
- WETH/TOKE
- WETH/GOVI

6.2.2 Binance Smart Chain

Due to the high impact gas fees have on the arbitrage opportunities and the clearly higher gas prices present on the Ethereum blockchain, it was expected that results on the BSC would yield better opportunities. Furthermore, the substantially smaller fluctuations in the gas price on the BSC were also expected to stabilize the opportunities found. These were the expectations going into the experiment of arbitrage on the BSC. Surprisingly the majority of pairs on BSC had similar results nevertheless, even if they were slightly closer to the 0 mark and more stable overall.

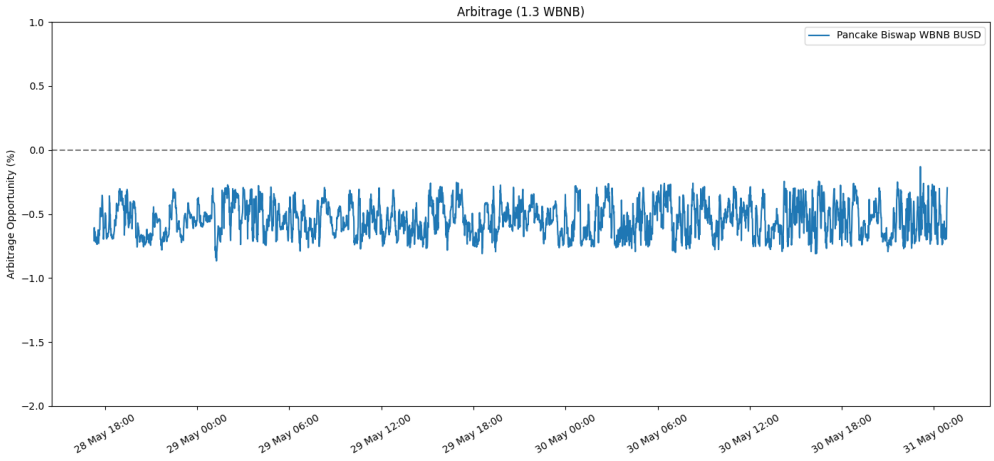

Figure 6.9: Arbitrage opportunities for pair WBNB/BUSD

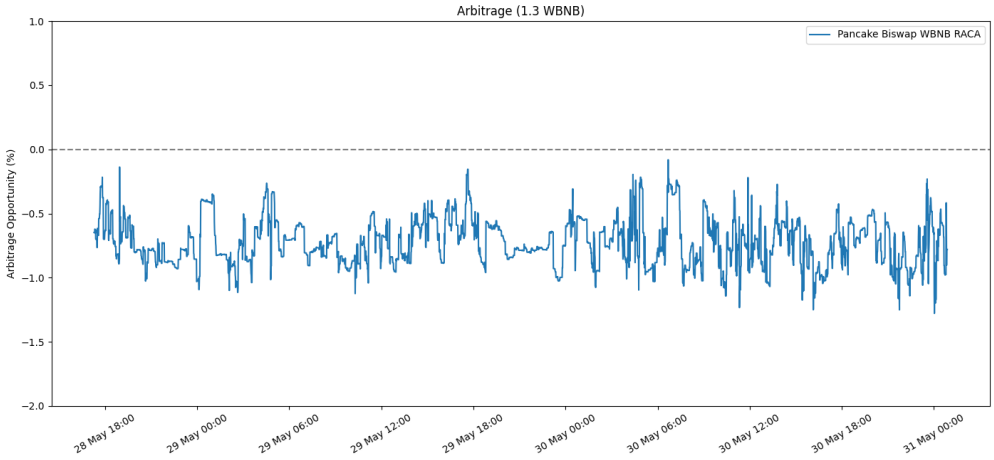

Figure 6.10: Arbitrage opportunities for pair WBNB/RACA

This can be seen in Figure 6.9 and Figure 6.10 which show the arbitrage opportunities on days with regular price action between WBNB and BUSD as well as WBNB and RACA on Biswap and Pancakeswap. What can be seen when comparing these two graphs, which both show the same time period, is that the smaller and less known token RACA is less stable and appears to have more substantial price deviations between the two DEXes. These could have several reasons:

1. Less liquidity in the pair's pool which results in a higher price impact from large trades.

2. Fewer arbitrage bots deployed for the specific pair which makes the asset less synchronized with the rest of the market.

During extreme price actions, on the other hand, a considerably higher amount of arbitrage opportunities are available between Pancakeswap and Biswap. During the crash in the cryptocurrency market, from the 7th until the 12th of May, that happened due to the Terra UST depegging event explained in section 2.3.4, prices were showing increasing inconsistencies between the observed exchanges. Arbitrageurs seemed to be unable to catch up with the quickly changing prices on all exchanges. This is visible through both charts in Figure 6.11 and Figure 6.12 that show arbitrage opportunities during two moments on May 11th involving the pairs WBNB/RACA as well as WBNB/BUSD. All these charts show the arbitrage opportunity graph peaking above the 0 line several times in the course of fewer than two hours which is particularly surprising for the pair WBNB/BUSD, which has one of the highest trading volumes as well as the largest liquidity pools on Pancakeswap and on Biswap.

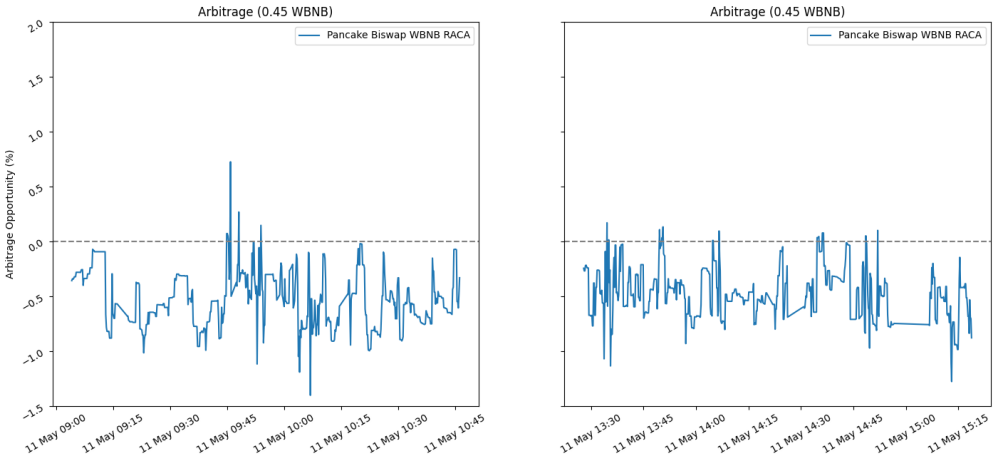

Figure 6.11: Arbitrage opportunities on May 11 WBNB/RACA

During this time, the arbitrage bot tried to execute several trades. Therefore the `swapSave` function of the smart contract deployed on the BSC was used. On multiple occasions the arbitrage detection server called the smart contract's `swapSave` function after detecting an arbitrage opportunity. However, by the time the trade was executed, there was no price

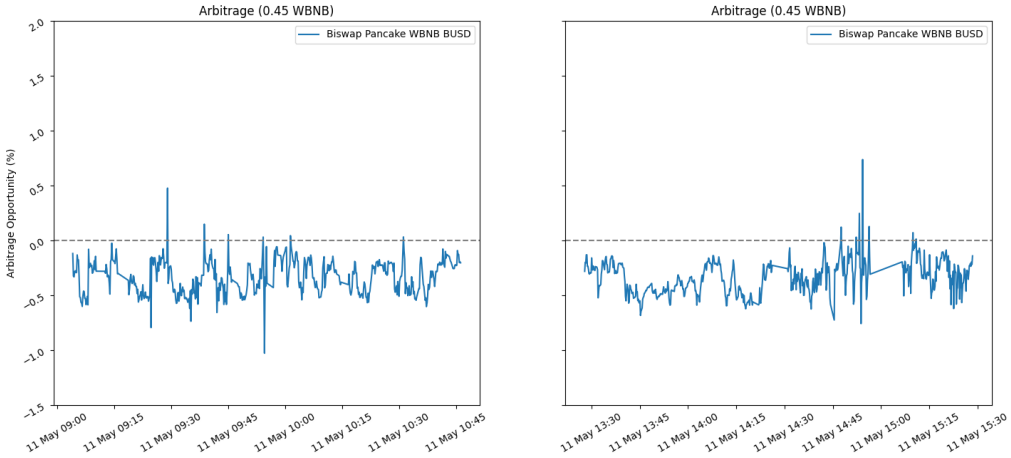

Figure 6.12: Arbitrage opportunities on May 11 WBNB/BUSD

difference left to exploit, and the contract reverted the trades, as expected. This can be seen in Figure 6.13, which is the transaction visible on the block explorer "bscScan". It shows all the essential information about the transaction, like status, block, timestamp, who executed the transaction, what contract was called, and more. Under "Status" it is visible that the transaction failed with an error message. This is the expected behavior of the smart contract's `swapSave` function when the arbitrage trades made no profit. When this happens, the transaction fee still has to be paid, which results in a minimal loss every time the estimation is wrong, or another arbitrage bot has already taken the opportunity. These types of transactions have been observed 17 times during the extreme price action on May 12th.

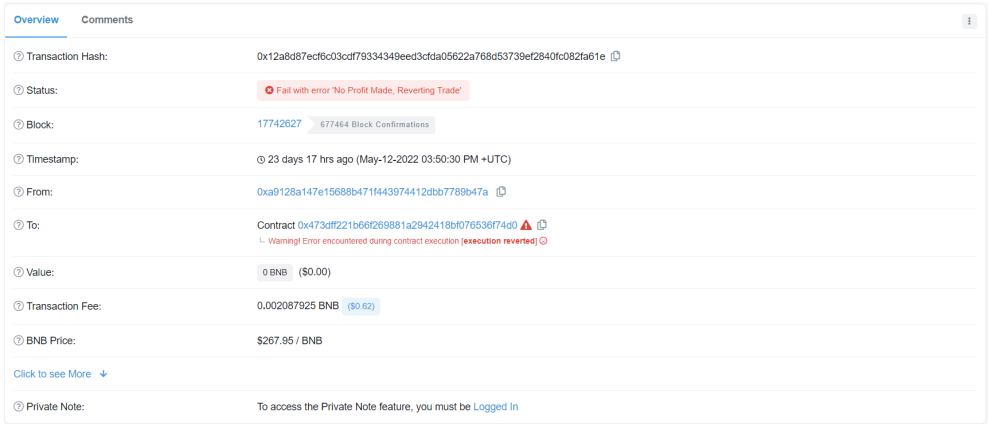

Figure 6.13: Reverted transaction, screenshot taken from bscscan.com

In Figure 6.14, a successfully executed arbitrage trade is presented. In this case, the arbitrage bot found a potentially lucrative opportunity when swapping WBNB to RACA on Pancakeswap and then swapping RACA back to WBNB on Biswap. Starting with 133.57$ worth of WBNB and ending with 133.75$, a profit of approximately 0.13%. Since this trade was performed during a testing period, the bot was running on a low input amount, and therefore the impact of gas fees that had to be paid (0.35$ worth of WBNB) was larger than the profit. The smart contract did not revert because the trade itself

was profitable, it was simply the call to the smart contract that costed more than it yielded. It could be argued that if a larger input amount was given to the bot to trade with, this and several other similar trades had a chance to be profitable. As seen from the screenshot presented in Figure 6.14, this trade also happened during the Terra UST crash, which again enforces the hypothesis that arbitrage opportunities are easier to find during extreme price action because markets are less synchronized.

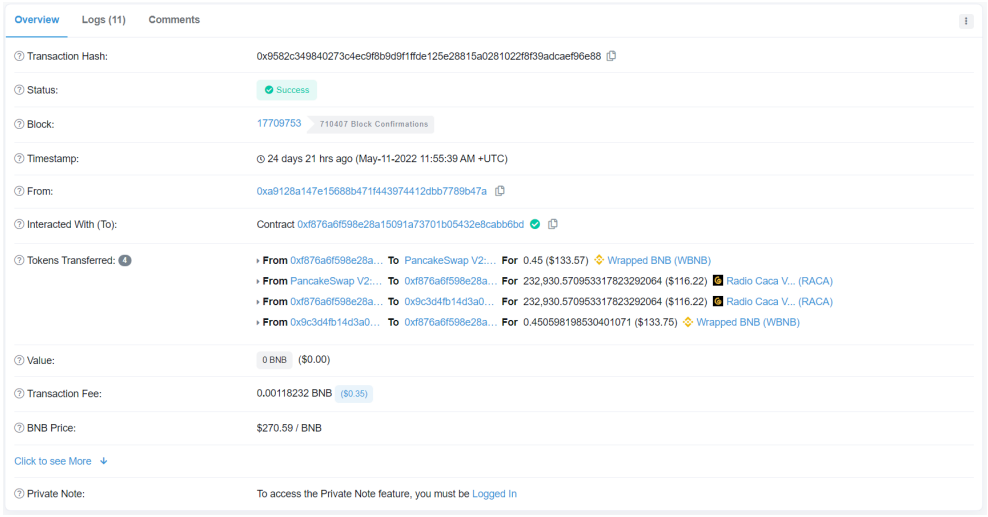

Figure 6.14: Successfully executed arbitrage trade, screenshot taken from bscscan.com

A question that arose after observing transactions similar to the one seen in Figure 6.14, is why the detection tool called the smart contract even though the transaction fees were higher than the profit possible by the trade. Two possible explanations for this are briefly discussed in the following.

- The detection measured a higher price difference between the two exchanges and calculated a profit even after deducting the correct amount of transaction fees
- The bot calculated all amounts correctly, but the gas price increased at the moment the trade was executed.

Since these problems could be caused by other arbitrage bots being faster at exploiting the possibilities, a simple approach was taken to reduce the risks of failed trade executions in these moments. In case the gas fee is slightly higher than calculated by the gas estimation, or the price difference decreases shortly before the trade can be executed, a profit threshold makes sure flipping a profitable trade to a non profitable trade through these two events becomes infeasible, by only calling the smart contract in case the profit calculated is higher than 0.2%. This percentage is an estimation of how much the changes of the environment could impact the profit in a worst-case scenario and was discovered by trial and error.

These trades were the last ones executed by the arbitrage bot. In the limited time of this experiment, no more extreme price actions like the ones triggered by the Terra UST crash were recorded. Data collected over 2 Months show no real arbitrage opportunities as can be seen in the charts available in the appendix. The pairs watched on BSC between *Pancakeswap* and *Biswap* were:

- WBNB/BUSD
- WBNB/BUSDT
- WBNB/CAKE
- WETH/RACA

On the BSC apart from the spatial arbitrage bot, a cyclic arbitrage bot was implemented and deployed. The results gathered by the cyclic arbitrage bot were very similar to the ones from the spatial arbitrage. No arbitrage opportunities were found during normal price action, as is visible in Figure 6.15. The opportunities are generally lower than the spatial arbitrage opportunities. This is likely due to prices being synchronized more effectively between different pairs on the same exchange.

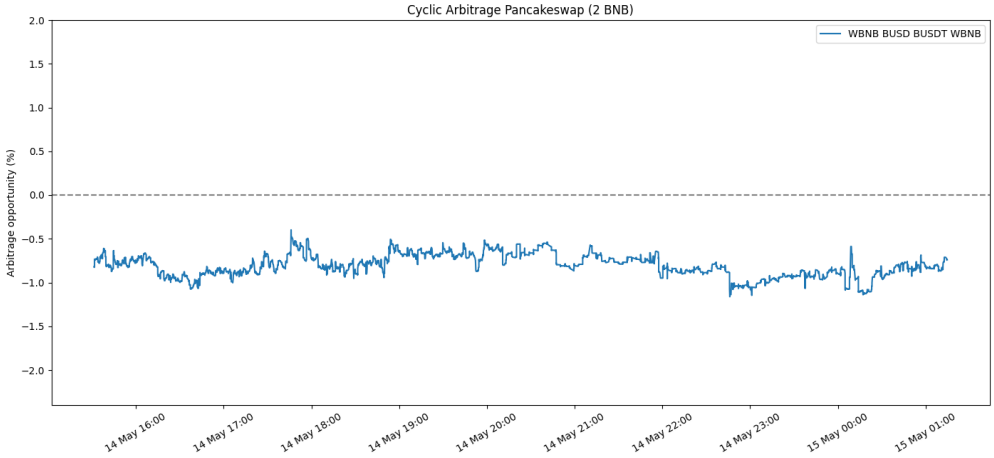

Figure 6.15: Cyclic Arbitrage opportunities Pancakeswap, path WBNB-BUSD-BUSDT-WBNB

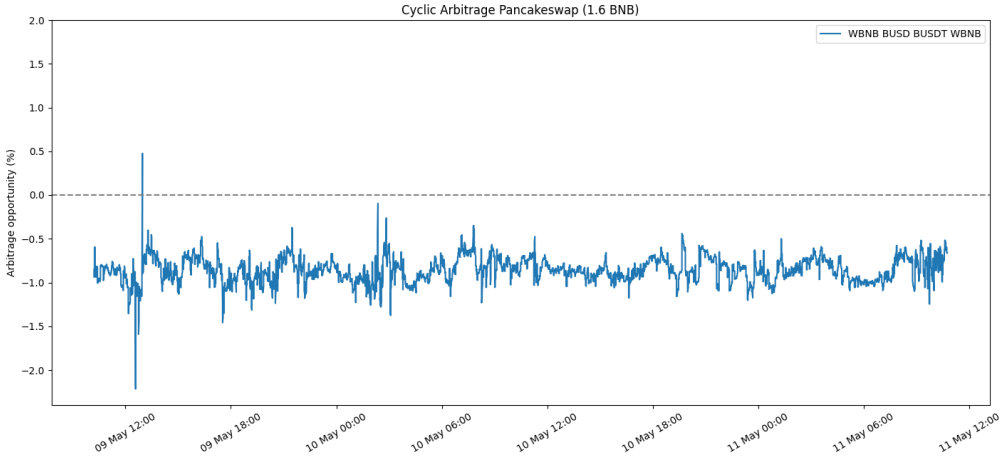

Figure 6.16: Cyclic Arbitrage opportunities Pancakeswap, path WBNB-BUSD-BUSDT-WBNB

Similar to the spatial arbitrage, during the Terra UST crash the opportunities increased slightly (visible in Figure 6.16). However, unlike the spatial arbitrage bot, no real arbitrage opportunities were found even in these more extreme price actions. The increased

consistency in prices on the same exchange can withstand the extreme price actions in this situation. It can therefore be said that in this case, there are fewer arbitrage opportunities in cyclic arbitrage than spatial arbitrage.

In a final note about arbitrage on the BSC, it was observed that the bot came close to profits on several occasions. It was also visible that the arbitrage opportunities were higher than on the Ethereum blockchain. It was not possible to gather data about the state of arbitrage opportunities on the Ethereum blockchain during the Terra UST depegging. Nevertheless, these would probably be low since in extreme price action the congestion of the Ethereum network increases gas price drastically. As can be seen in Figure 6.17, on the days the spatial arbitrage bot on the BSC found the most opportunities, the gas prices on ETH were very high. This would have made it extremely difficult to find arbitrage opportunities, especially with the amounts used in the arbitrage detection when having very high gas prices.

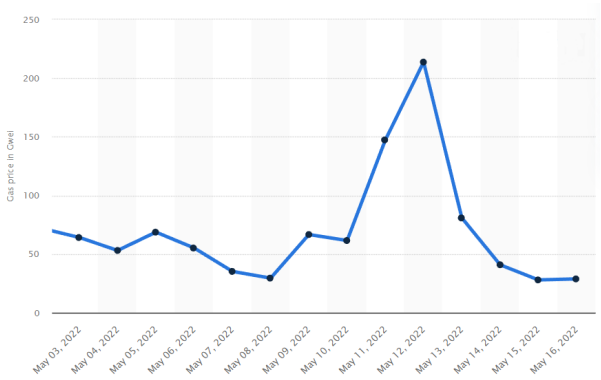

Figure 6.17: Chart showing gas prices on Ethereum network during Terra UST event, taken from statista.com

Chapter 7

Discussion

The prototypes developed in this study present a view of the current situation concerning arbitrage in cryptocurrency markets. In the following, some important notes about the results are made, and the feasibility of arbitrage on cryptocurrency in general is discussed. Finally, this chapter touches on the relevance of the results presented.

Arbitrage on centralized crpytocurrency exchanges is becoming very exclusive, just like in traditional finance. While a few years ago the markets were not as efficient yet and [32] and [33] were able to implement a simple and successful arbitrage trading bot on CEXes, the recent increase in popularity was able to give the largest CEXes the necessary liquidity to maintain a stable and reliable price that is synchronized with the rest of the market. Only minimal arbitrage opportunities are present and mostly taken by the CEXes themselves or other players in the market with high-performance computing equipment and ultra-low latency to the CEX servers. This makes it more difficult to perform profitable arbitrage trades on basic equipment and demands for more sophisticated arbitrage strategies. In this experiment, only a handful of exchanges and pairs were analyzed for arbitrage trades which still leaves room for possibilities between a higher number of markets. Furthermore, the bot was not active during any extreme price action, which can have a positive impact on arbitrage opportunities, as already discovered by [38].

On decentralized exchanges, arbitrage opportunities are also increasingly more difficult to find. Nevertheless, it is possible to find pairs that are more prone to price differences between exchanges. Especially during more chaotic moments in the market, arbitrage seems to be possible when executed with the correct trade size and execution speed. The execution of arbitrage trades has to be very fast and is one of the most important parts when trying to perform arbitrage.

A problem when it comes to the execution of arbitrage trades is that miners, in proof of work blockchains, can have full control over the selection of transactions in the mempool (a pot that stores all unconfirmed transactions). Normally a miner chooses which transaction to include in the next block by prioritizing transactions that pay a high gas price but is not obliged to do so. Therefore a miner can try to extract profits by reordering transactions [39]. This widespread phenomenon that presents a threat to the consensus-layer security of a network, according to [28], is called miner extractable value (MEV). MEV represents

the total amount of rewards that miners are able to extract by manipulating transactions over a given period of time [28]. Since these fees are large enough, miners are willing to create serious threats to the network like undercutting or time-bandit attacks. These attacks give miners the possibility to ”steal” arbitrage trades from arbitrageurs on the network [28].

Not only miners can attempt to influence the trade execution order, but also arbitrageurs themselves. In priority gas auctions (PGAs), arbitrage bots compete by trying to outbid each other's transaction fees. Since most miners choose transactions to include in the next block by checking the reward received per transaction, arbitrage bots can try to elevate the gas price of their arbitrage trade as long as it stays below the recorded arbitrage opportunity. This is achievable due to the possibility of overwriting a previously placed transaction with a newer one with a higher transaction fee. PGAs have been observed and analyzed by Daian *et al.* in [28].

MEV creates a clearly unfair competition through front running when it comes to arbitrage on DEXes, but there are already approaches to eliminate this problem. Several decentralized applications try to combat this issue by using batch orders to execute several transactions at once instead of sending and executing them individually. Using this off-chain approach decentralized applications can execute all transactions in the same block, and remove the possibility for miners to extract any value through reordering [40]. Nevertheless, as long as this or other approaches against MEV are not broadly adopted, the already marginal arbitrage opportunities present are increasingly difficult to exploit, as has been seen through the course of this work. Several recorded transactions have been reverted by the implemented smart contract after finding arbitrage opportunities because they were considered non-profitable. There is a very high probability that the detected price differences had already been exploited by other arbitrage bots.

Furthermore, there are still some limitations with the current implementation of the prototypes presented in this work that have to be considered when evaluating the results. Even if several different approached to arbitrage were implemented, these prototypes do not cover a large enough set of cryptocurrencies and exchanges to make a conclusive assessment about all arbitrage opportunities present on centralized as well as decentralized exchanges. The results should give a suggestion about the state and difficulty to find arbitrage opportunities in these newer and less explored areas of finance. With several hundred decentralized and centralized exchanges as well as thousands of trading pairs, the exploration of every opportunity on the market is infeasible and expands far beyond the scope of this work.

Chapter 8

Summary, Future Work & Conclusions

In this thesis, the design and implementation of several arbitrage bot prototypes have been presented. These implementations gave an insight into the functionality of automatic trading on centralized as well as decentralized exchanges. Despite the fact that no profits were made during the execution of the arbitrage bots on CeFi as well as DeFi, some important findings were achieved. It has been shown that the results collected presented a rather stable and consistent price throughout centralized exchanges. Thereby suggesting low arbitrage opportunities with basic equipment as used in this work. Furthermore, the results collected on decentralized exchanges presented a more mixed condition, indicating the possibility of minor arbitrage possibilities, especially in more extreme price conditions. It has also been argued that more unpopular coins have a slightly higher chance of price deviation across exchanges. These findings motivate that further investigation is necessary in arbitrage on decentralized exchanges.

There are still many areas that can be improved and further investigated in future work on this topic. Some suggestions for optimizations of the DEX arbitrage bot are presented below:

Cross Chain Arbitrage across different exchanges on the same blockchain is simpler and more secure. It is very likely that between different blockchains, price deviations are considerably larger. Improved bridging mechanisms give the possibility to explore that. This could drastically increase the chances of successful arbitrage through higher complexity. However, cross chain arbitrage comes at the cost of having no atomicity when executing the two trades on different blockchains.

Flash Loans One interesting concept present in DeFi which can be used to enhance arbitrage is flash loans. These are uncollateralized loans available in DeFi which provide the trader with a large amount of capital within one transaction [41]. When using flash loans, the amount of capital loaned is used to execute arbitrage trades, which makes even very low price deviations between exchanges highly profitable. Flash loans can be borrowed from some DEXes such as *dYdX*, *Aave*, and *Uniswap* using smart contract calls.

Performance Using an API to access a node through a provider is coupled with delays when executing trades. This performance can be enhanced by having a running node available on sight which can decrease delays. It therefore lowers the possibility of a trade being triggered by the arbitrage detection tool and not successfully executed by the smart contract. Arbitrage is a competition between different bots, where every millisecond counts.

Variation The more pairs, exchanges, and blockchains are explored when it comes to price differences, the higher the chance of finding arbitrage opportunities. In future work, more variation in these three points should be targeted.

Combined Arbitrage While in the presented prototypes, spatial and cyclic arbitrage were looked at in isolation, an interesting approach would be to explore the possibility of combining these two arbitrage strategies. For example, executing cyclic swaps to an asset that has a different price on another exchange and using a spatial swap to exploit this difference. This could lead to more arbitrage opportunities being found and gives the option to accumulate several smaller opportunities to a larger one with a set of trades, all executed atomically.

In this thesis a foundation for CeFi and DeFi arbitrage has been presented that can be used to compare prices on the market and execute trades to exploit them if wanted. Despite the fact that still many improvements have to be made to achieve profitability of the arbitrage bot, the purpose of this work was to demonstrate an idea for a working implementation and analyze the market through the results and performance of the prototypes. In conclusion, it can be said that there is a hard competition on arbitrage throughout the markets of CeFi and DeFi, and it demands a very thorough analysis of the market and likely more sophisticated strategies to be able to perform it in a manner that produces profits.

Bibliography

[1] P. Blows. "Defi revolution: The factors driving the growth of decentralised finance". (2022), [Online]. Available: https://www.finance-monthly.com/2022/03/defi-revolution-the-factors-driving-the-growth-of-decentralised-finance. last visit July 1, 2022.

[2] K. Qin, L. Zhou, Y. Afonin, L. Lazzaretti, A. Gervais, "Cefi vs. defi - comparing centralized to decentralized finance", *CoRR*, vol. abs/2106.08157, 2021.

[3] U. W. Chohan, "Decentralized finance (defi): An emergent alternative financial architecture", *Econometric Modeling: International Financial Markets - Foreign Exchange eJournal*, 2021.

[4] Coingecko. "Top decentralized exchanges on coingecko by trading volume". (2021), [Online]. Available: https://www.coingecko.com/en/exchanges. last visit June 2, 2022.

[5] Y. Wang, Y. Chen, H. Wu, L. Zhou, S. Deng, R. Wattenhofer, *Cyclic arbitrage in decentralized exchanges*, 2021.

[6] S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system*, 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf, last visit April 10, 2022.

[7] S. Geiregat, "Cryptocurrencies are (smart) contracts", *Computer Law & Security Review*, vol. 34, no. 5, pp. 1144–1149, 2018.

[8] Chainlink. "What is miner-extractable value (mev)?" (2021), [Online]. Available: https://blog.chain.link/what-is-miner-extractable-value-mev/. last visit May 19, 2022.

[9] V. Buterin, "Ethereum white paper: A next generation smart contract & decentralized application platform", 2013. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper, last visit June 21, 2022.

[10] Binance. "Know your customer (kyc)". (2017), [Online]. Available: https://academy.binance.com/en/glossary/know-your-customer. last visit April 7, 2022.

[11] A. Bloomenthal. "Market maker". (2021), [Online]. Available: https://www.investopedia.com/terms/m/marketmaker.asp. last visit April 17, 2022.

[12] A. Milton. "Bid, ask, and last prices defined". (2022), [Online]. Available: https://www.thebalance.com/trading-definitions-of-bid-ask-and-last-market-prices-1031026. last visit June 28, 2022.

[13] J. Xu, N. Vavryk, K. Paruch, S. Cousaert, "Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols", *CoRR*, vol. abs/2103.12732, 2021.

[14] V. Buterin. "Let's run on-chain decentralized exchanges the way we run prediction markets". (2016), [Online]. Available: https://www.reddit.com/r/ethereum/comments/55m04x/lets_run_onchain_decentralized_exchanges_the_way/. last visit June 25, 2022.

[15] Cryptopedia. "What are automated market makers?" (2021), [Online]. Available: https://www.gemini.com/cryptopedia/amm-what-are-automated-market-makers#section-liquidity-pools-and-liquidity-providers. last visit April 10, 2022.

[16] F. Schär. "Decentralized finance: On blockchain- and smart contract-based financial markets". (2021), [Online]. Available: https://research.stlouisfed.org/publications/review/2021/02/05/decentralized-finance-on-blockchain-and-smart-contract-based-financial-markets. last visit May 5, 2022.

[17] V. Mohan, "Automated market makers and decentralized exchanges: A defi primer", *Financial Innovation*, 2022.

[18] J. Frankenfield. "Atomic swap". (2022), [Online]. Available: https://www.investopedia.com/terms/a/atomic-swaps.asp. last visit July 1, 2022.

[19] Uniswap. "Smart contracts". (2018), [Online]. Available: https://docs.uniswap.org/protocol/V2/concepts/protocol-overview/smart-contracts. last visit July 1, 2022.

[20] T. T. Techie. "Ethereum network: Abis and smart contracts". (2022), [Online]. Available: https://medium.com/@deshayk/ethereum-network-abis-and-smart-contracts-eadb0e186845. last visit July 1, 2022.

[21] Quantent. "Explained | what is cross-chain protocol and why is it important?" (2022), [Online]. Available: https://www.cnbctv18.com/cryptocurrency/explained--what-is-cross-chain-protocol-and-why-is-it-important-12151502.htm. last visit April 17, 2022.

[22] G. Caldarelli, "Wrapping trust for interoperability: A preliminary study of wrapped tokens", *Information*, vol. 13, no. 1, 2022.

[23] L. Herzs. "A comparison of various blockchain protocols". (2022), [Online]. Available: https://www.leewayhertz.com/comparison-of-blockchain-protocols/. last visit June 23, 2022.

[24] S. Bachmann, *Proof of stake for bazo*, Zürich, Switzerland: Communication Systems Group, Department of Informatics, Jan. 2018. [Online]. Available: https://files.ifi.uzh.ch/CSG/staff/bocek/extern/theses/BA-Simon-Bachmann.pdf.

[25] J. Royal. "What are stablecoins and how do they affect the cryptocurrency market?" (2022), [Online]. Available: https://www.bankrate.com/investing/stablecoin-cryptocurrency/. last visit June 27, 2022.

[26] M. L. Nihar Shah. "The depegging of ust". (2022), [Online]. Available: https://jumpcrypto.com/the-depegging-of-ust/. last visit June 16, 2022.

[27] "Terraclassicusd to usd chart". (2022), [Online]. Available: https://coinmarketcap.com/currencies/terrausd/. last visit June 27, 2022.

[28] P. Daian, S. Goldfeder, T. Kell, *et al.*, *Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges*, 2019.

[29] J. Xu, K. Paruch, S. Cousaert, Y. Feng, *Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols*, 2021.

[30] R. Levus, A. Berko, L. Chyrun, V. Panasyuk, M. Hrubel, "Intelligent system for arbitrage situations searching in the cryptocurrency market", in *Modern Machine Learning Technologies and Data Science Workshop. Proc. 3rd International Workshop Volume I: Main Conference*, vol. 2917, Lviv-Shatsk, Ukraine, 2021, pp. 407–440.

[31] I. Makarov, A. Schoar, "Trading and arbitrage in cryptocurrency markets", *Journal of Financial Economics*, vol. 135, no. 2, pp. 293–319, 2020.

[32] S. Bistarelli, A. Cretarola, G. Figà-Talamanca, I. Mercanti, M. Patacca, "Is arbitrage possible in the bitcoin market?", in *Economics of Grids, Clouds, Systems, and Services*, Cham, Switzerland: Springer, 2019, pp. 243–251.

[33] E. Han, *Intra-exchange cryptocurrency arbitrage bot*, 2018, Master Thesis, San José State University, USA, Supervisors: Thomas Austin, Mark Stamp, Jon Pearce.

[34] N. Boonpeam, W. Werapun, T. Karode, "The arbitrage system on decentralized exchanges", in *2021 18th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, Chiang Mai, Thailand: IEEE, 2021, pp. 768–771.

[35] CCXT. "Ccxt – cryptocurrency exchange trading library". (2021), [Online]. Available: `https://github.com/ccxt/ccxt`. last visit March 21, 2022.

[36] Ethers. "What is ethers?" (2022), [Online]. Available: `https://docs.ethers.io/v5`. last visit May 26, 2022.

[37] C. Gebbia. "Arbitrage bot". (2022), [Online]. Available: `https://github.com/claudioge/arbitrageBots/`. last visit July 08, 2022.

[38] A. Kabašinskas, K. Šutienė, "Key roles of crypto-exchanges in generating arbitrage opportunities", *Entropy*, vol. 23, no. 4, 2021.

[39] A. Judmayer, N. Stifter, P. Schindler, E. R. Weippl, "Estimating (miner) extractable value is hard, let's go shopping!", *IACR Cryptol. ePrint Arch.*, p. 1231, 2021.

[40] F. Ernst. "How to fix ethereum's mev problem and give traders the best price". (2021), [Online]. Available: `https://www.coindesk.com/markets/2021/07/27/how-to-fix-ethereums-mev-problem-and-give-traders-the-best-price/`. last visit July 08, 2022.

[41] K. Qin, L. Zhou, B. Livshits, A. Gervais, "Attacking the defi ecosystem with flash loans for fun and profit", *CoRR*, vol. abs/2003.03810, 2020.

Abbreviations

CeFi	Centralized Finance
DeFi	Decentralized Finance
CEX	Centralized Exchange
DEX	Decentralized Exchange
AMM	Automated Market Maker
LP	Liquidity Pool
MEV	Miner Extractable Value
ABI	Application Binary Interface
PoW	Proof of Work
PoS	Proof of Stake
PGA	Priority Gas Auction
API	Application Programming Interface
SC	Smart Contract
ETH	Ethereum
BSC	Binance Smart Chain
AVA	Avalanche

List of Figures

List of Tables

Appendix A

Additional arbitrage results

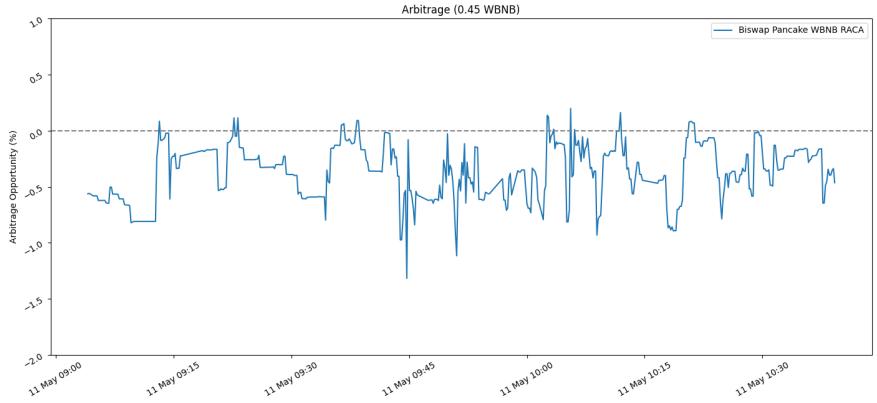

Figure A.1: Arbitrage opportunities for pair WAVA/RACA, during Terra UST depegging

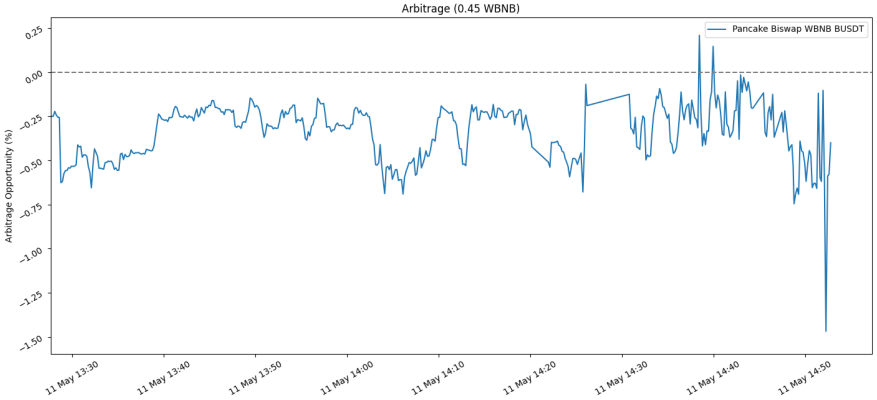

Figure A.2: Arbitrage opportunities for pair WAVA/BUSDT, during Terra UST depegging

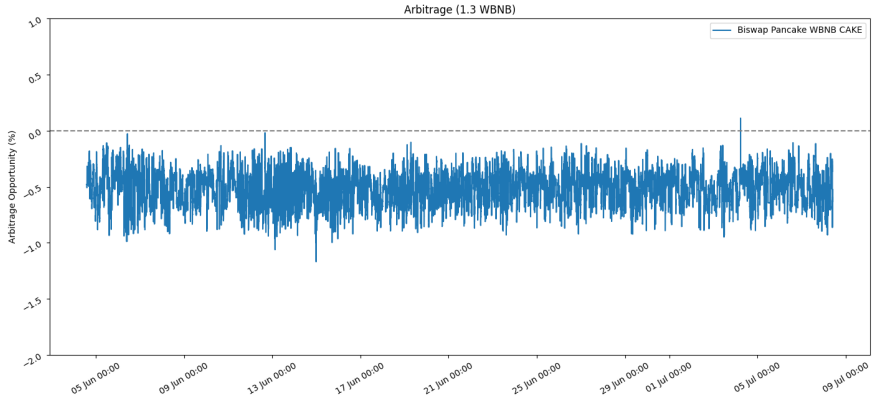

Figure A.3: Arbitrage opportunities for pair WAVA/CAKE, over 1 month

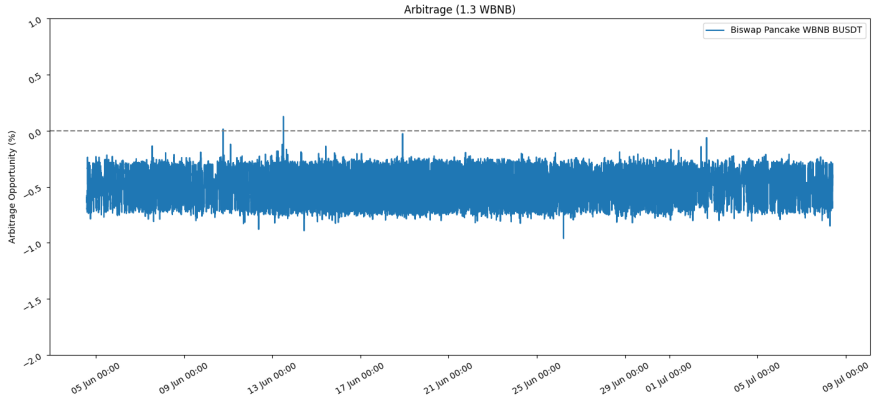

Figure A.4: Arbitrage opportunities for pair WAVA/BUSDT, over 1 month

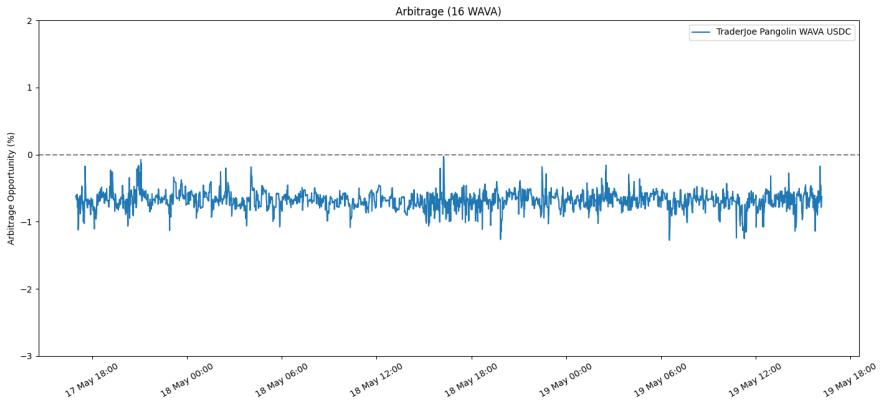

Figure A.5: Arbitrage opportunities on Avalanche blockchain for pair WAVA/USDC

Appendix B

Installation Guidelines

Prerequisites

1. The source code of this project is available on GitHub: "github.com/claudioge/arbitrageBots"
2. The necessary libraries and tools to build and run this project:
   - Node v14
   - npm
   - Python 3.10
   - pip

B.1 Spatial Centralized Exchange Bot

NOTICE: Assets on the CEXes are necessary to execute arbitrage trades.

1. Add API Keys and Secrets to the .env file
2. Enter the following commands into the terminal:
   - `pip install json`
   - `pip install time`
   - `pip install dotenv`
   - `pip install os`
   - `pip install ccxt`
   - `pip install datetime`
   - `python Start.py`

B.2 Decentralized Exchange Bots

1. The smart contract for the arbitrage bot has to be deployed:
    - Connect Wallet to the Metamask extension for the browser.
    - Choose Network to check for arbitrage opportunities on Metamask (ETH, BSC, AVA are available for the spatial arbitrage bot, BSC for the cyclic arbitrage bot).
    - Go to "remix.ethereum.org" and connect the Metamask wallet.
    - Choose "Load a local file into current workspace" and choose contract for desired arbitrage strategy. SpatialArb.sol for spatial arbitrage bot or CyclicArb.sol for cyclic arbitrage bot (available only on BSC).
    - Open the file on remix, then go to the compile tab and compile the smart contract.
    - Go to the deploy tab, make sure the Environment is set to "Injected Web3" and choose the right contract to deploy. Then deploy smart contract.

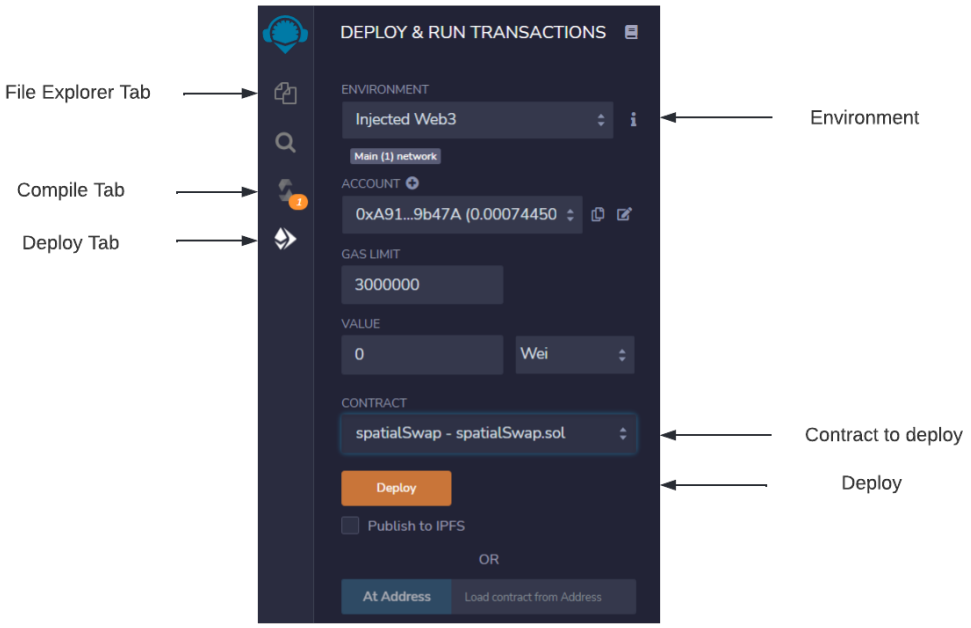

Figure B.1: Interfact to deploy smart contract taken from remix.ethereum.org

    - Use Metamask to send base asset (WETH on ETH, WBNB on BSC, WAVA on AVA) to the address of the deployed smart contract.
    - Assets can be withdrawn by calling the "recoverBalance" function through remix. The token address of the token to be withdrawn has to be passed as input to this function.

2. The Secretes.json file has to be filled with the following information:
    - Private key of wallet

- Address of wallet
- API link of provider depending on network
- Address of deployed smart contract

3. Enter the following commands into the terminal:

   - `npm i`
   - `npx ts-node ./nameOfArbitrageBotFile`

Appendix C

Contents of the ZIP Archive

1. This thesis as PDF
2. This thesis as LaTeXsource in a .zip file
3. The source code of all the arbitrage bots
4. Midterm presentation slides as PDF
5. Data sets for charts generated as a .txt file with a download link