UNIVERSITY OF COPENHAGEN

MSc in Mathematics
MSc in Computer Science

Analysis of Liquidity Provision within Uniswap V2

Alexei Condurachi

Nicolai Musing Tudborg

Supervised by Omri Ross

June 2022

UNIVERSITY OF COPENHAGEN

Alexei Condurachi
Nicolai Musing Tudborg
Analysis of Liquidity Provision within Uniswap V2
MSc in Mathematics
MSc in Computer Science , June 2022
Supervisor: Omri Ross

University of Copenhagen
Faculty of Science
Masters Degree in Computer Science
Universitetsparken 5
2100 Copenhagen Ø

Acknowledgements

We would like to give our thanks to Professor Omri Ross, our thesis supervisor, for guiding us through the past months. His expertise made the completion of this project possible.
We would also like to thanks PhD fellow Henrik Bjørn Axelsen for his counseling, for the many conversations that helped us grow and for the feedback that he always gave us promptly.

Thank you Nicolai, for the beautiful experience of working side by side. You were always ready to listen and had so many brilliant ideas. It was a pleasure working with you and getting to know you day by day.

Thanks to my family. Without you and your sacrifices this experience could have never been possible in the first place. Thank you for all the time you spent making sure I felt home even when I wasn't, and for being always there for me. I am proud of what we have accomplished in all these years.

A special thanks to all the friends that made the last two years special:

To Audrey, Cristina, Greta, Josafath, Andrea and Alessandro. You have been there when I needed you, with long video calls, listening to me complaining and talking endlessly about whatever.

To Mike and Elisa, for the incredible help you provided me in the darkest days and for all the memories we built together. You are truly amazing friends with big hearts.

To my friends Giulia, Alice, Manuel and Alessandro. For pushing me to become a better person and for the amazing time we had. Our picture was always there on my desk, reminding me that I had some amazing friends that would never leave my side.

To Aasavari, my fortune cookie in Copenhagen, for listening to me and for all the beautiful moments.

To Julia. You maybe have not realized it, but meeting you changed my life completely. You pushed me out of my comfort zone so many times and I ended meeting such amazing people thanks to you. You are one in a million, so please don't ever change.

To the person that two years ago convinced and obliged me to send my application to KU in the first place. You believed in me while I was doubting. I have no idea where I would be now, if it wasn't for you insisting. I wish things went differently. I hope you are well and that you achieved all the goals you set for yourself.

Abstract

This project comprises of an introduction to Decentralized Finance, its infrastructure, and Decentralized Exchanges, as well as an analysis of the various aspects that impact gains for liquidity providers of the largest Constant Product Market Makers: Uniswap-V2. A digital twin of the CPMM has been built in order to undertake simulations and analysis. Data collected directly from Uniswap has been used to set the initial parameters of simulations to recreate trading trends that are currently occurring in the market.

Experiments are conducted varying the number of liquidity providers interacting, their initial holdings and tokens tradable. Moreover, the impact of different trading fees on liquidity providers' returns has been investigated. Results show that providing strategically liquidity to pools proves to be a profitable strategy that yields positive annual returns up to 1%. Experiments indicate also that the presence of other active providers influences returns to different extents and in different way. Smaller liquidity providers are seen to benefit from the presence of wealthier providers, while bigger liquidity providers are negatively impacted by active providers with similar portfolios. As expected, gains are directly correlated to pools' trading volumes, as they are the source of profit for liquidity providers. This can be seen by the strategies adopted by providers.

Contents

1 Introduction

In a society that is moving more and more towards globalization, markets play a crucial role in keeping economies dynamic whilst being accessible from all around the globe. The *Bank for International Settlements* has estimated a notional value of outstanding derivatives of $610 trillion at end of June 2021 [1]. In the same period, the real estate market's value capped at an estimate of $280,60 trillion and world's Gross Domestic Product (GDP) reached a value of $85 trillion. This data shows that markets have a significant impact and influence on the global economy, as they contain a huge part of global wealth. The adoption of computers is another important factor that has contributed to markets reaching these levels. This aided in speeding up trades because they can be executed as soon as requirements are met, increasing the total volume of trades that can be handled. High-frequency trading is one example of computer-related innovation that is simply not possible in a market where trades are executed by brokers.

Markets' centrality is due to the fact that they offer a plethora of financial tools depending on what sector is taken into consideration. The range extends from the more sophisticated and complex financial contracts (such as derivatives), to currency exchange, to stocks and bonds. All of these tools were developed to fulfill a specific requirement. Consider the following:

- Employee salaries must be paid in different currencies by companies with offices in different countries.s a result, a foreign exchange market must be established. Foreign exchange markets reached a daily trading volume of $6.6 trillion a day in 2019 [4].

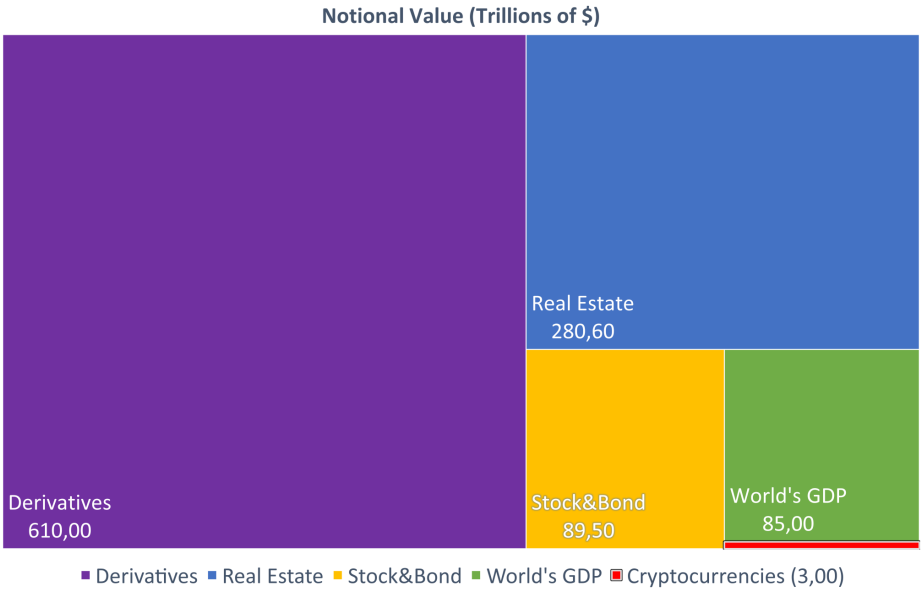

Figure 1.1.: Data shows an estimate of different markets size. Sources: BIS [1], IMF [2], Statista [3].

- To operate and grow, businesses and governments require a steady stream of cash. This necessity has led to the creation of bonds: in order to maintain a consistent cash flow, companies or nations issue bonds, which provide immediate liquidity while promising investors future rewards.

- Since capital is needed for future investments, a company aims to go public and sell assets. Owning an asset is more accurately expressed as owning a piece of a company, implying that the profit is intrinsic and determined by the difference between the buying and selling prices.

The cryptocurrency market is one that has grown at an incredible rate. It reached a valuation of $3 trillion at the end of 2021, only 13 years after the first cryptocurrency, Bitcoin, was introduced [5]. As more people became involved in this field, new financial ecosystems were created and improved to allow these new digital currencies to be exchanged. Decentralized Finance (DeFi) is a component of the crypto environment (as it can be seen in Figure [1.2]) in which securities are exchanged peer-to-peer without an intermediary and there is no centralized entity that adjusts market trends. The DeFi ecosystem includes

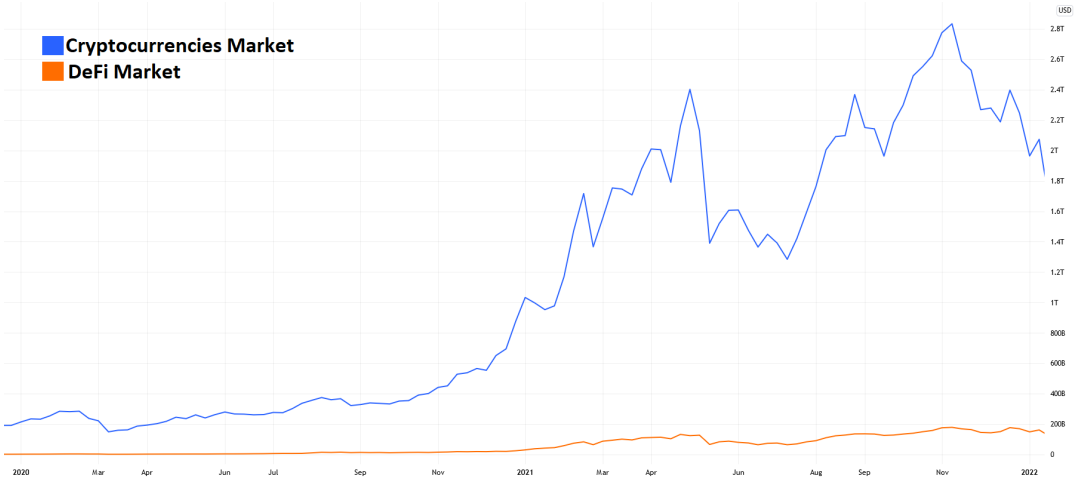

Figure 1.2.: Comparison of cryptocurrencies market and Decentralized Finance as per TradingView.

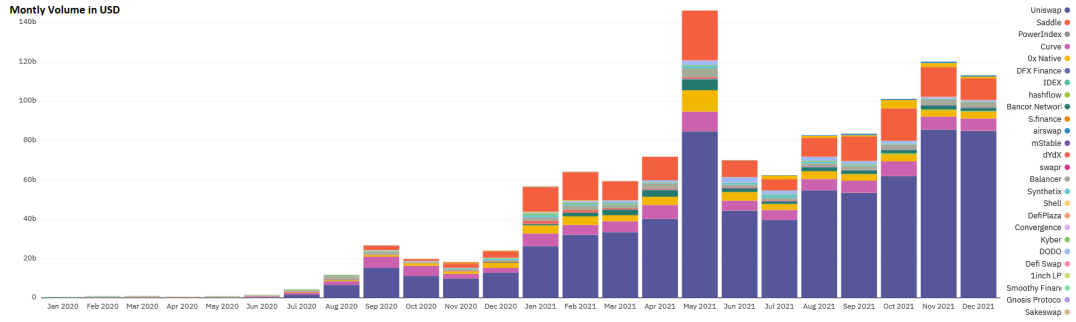

Figure 1.3.: Montly DEXs Volumes. Uniswap has a leading trading volume. Data collected from Dune.

Decentralized Exchanges (DEXs). Their popularity can be traced back to their more *transparent* approach when compared to traditional exchanges and centralized cryptoexchanges (e.g., Binance), since all information is publicly accessible. Due to their technical structure, many DEXs have been implemented with a trading style that differs from the central limit order book: instead of storing bid and sell prices and operating trades whenever they are met, assets are stored in pools where prices are algorithmically determined by reserves of each asset. This type of DEX is known as an Automated Market Makers (AMMs). As shown in Figure [1.3], Uniswap is the largest AMM in terms of total value and trading volume at the end of 2021. It is natural to ask how rewards and prices are determined. There are numerous factors that can influence price fluctuations and they are not always easily identifiable. *Liquidity* is among those factors. Its centrality can be easily seen in Figure [1.4], where the price of the S&P500 market index is compared to liquidity provided by the Federal Reserve according to their Balance Sheet. As the S&P500 reflects

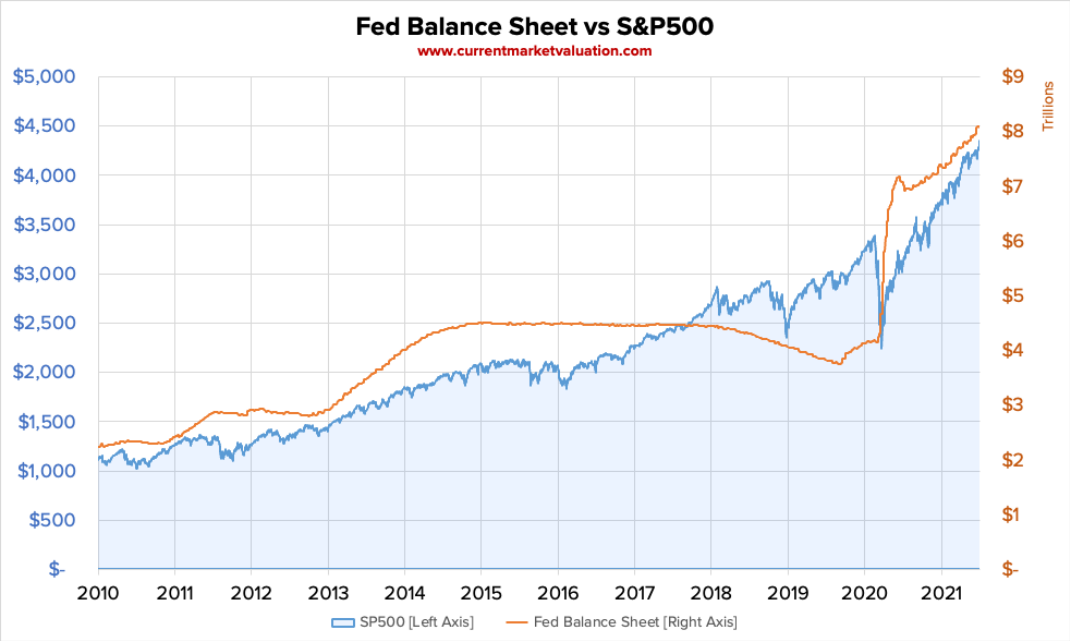

Figure 1.4.: Source: currentmarketvaluation.com [6].

market's trends of the largest companies in the US asset market, it is evident that in this case a higher liquidity boosts trading volumes that hence increases assets' values, setting an overall positive trend. This is due to prices being determined by the demand-supply ratio. Small price variations are expected if supply remains stable and capable of meeting demand. If, on the other hand, there is an imbalance between demand and supply, prices will experience large fluctuations until an equilibrium is restored. Liquidity is even more important in the case of AMMs, as it plays a significant role in determining asset prices. Analyzing the effect of price fluctuations on the wealth of liquidity providers is critical for a successful AMM. Thus we seek to answer: *What key factors determine LP gains and losses in Uniswap V2*? In this thesis, we create a digital twin of the protocol and recreate trading actions based on empirical data collected from the protocol.[1]

[1]Uniswap V2 was selected over V3, because it is more established and less likely to present noisy data.

2 Traditional Markets

2.1 Securities

Before discussing the environment and characteristics of DeFi, it is necessary to briefly discuss traditional financial instruments and markets. The following section's definitions and theory are based on [7].
As the term *security* appears a conspicuous amount of times in financial documents and literature, a definition of such is required. Accordingly to [7] a security is defined as a "*[...] legal contract representing the right to receive future benefits under a stated set of conditions*". Another possible definition is given by the the Howey Test[1]: "*an investment of money in a common enterprise with a reasonable expectation of profits to be derived from the efforts of others*"[8]. In the United States securities are regulated by the U.S. Securities and Exchange Commission (SEC), while in the European Union this is done by the European Securities and Markets Authority (ESMA).
Financial securities can be categorized as in diagram [2.1]. One possible way of differentiating securities is by determining either if the investment requires an intermediary or not [7]: investors might prefer buying shares of a fund's portfolio instead of directly building one by acquiring assets. Amongst the securities that are directly bought, it is possible make a distinction based on their time horizon: *money market instruments* (less than one year) or *capital market instruments* (more than one year). In addition to those, another category is *derivative instruments*, defined this way because the payoff of such financial securities is *derived* from prices of a basket of underlying primary assets.
Listing all the different types of securities would go beyond the scope of this project, hence only securities that will be referred to in later discussions are presented. These have been identified in:

[1] The Howey Test refers to the U.S. Supreme Court case for determining whether a transaction qualifies as a security subject to disclosure and registration requirements

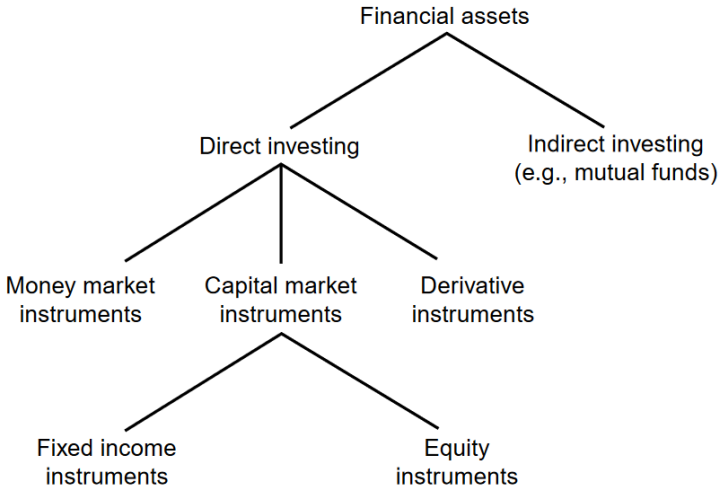

Figure 2.1.: Categorization of financial securities as presented in [7].

- Treasury bills: they are money market instruments issued by governments. They are considered to be a riskless investment, since they have no risk of default,[2] known return and a short time horizon.

- Repurchase agreements: contracts between a borrower and a lender to first sell and then repurchase a specified security (usually one issued by a government). The return is computed from the difference between selling and buying prices. Their importance is due to the fact that they allow short positions[3].

- Stocks: they can be divided in preferred stocks and common ones. The first offers periodic payments (known as dividends), that are promised but not *granted*: failed payments are cumulated until the issuer is able to repay them without calling for a default. Similarly, common stocks represent an ownership claim on earnings and assets of a company. They might grant access to dividends after all other shareholders have been paid, but it is left to management's discretion. Common stocks are the riskiest securities amongst those presented.

[2] a security defaults when it fails to meet contract's conditions
[3] short position: when a trader sells a security with the intent of repurchasing it at a lower price

2.2 Trading Dynamics

Understanding the dynamics and the different characteristics of trading processes is necessary to fully comprehend later on the correlation that exists between those and liquidity provision.
An investor can either decide to place an order through a brokerage firm or directly with an exchange. The latter case eliminates the need of any intermediary and has become available as a result of the financial sector's embrace of electronic devices. Buyers specify the number of assets they want to purchase as well as a bid price, *i.e.* the most they are willing to pay for a unit. Sellers, on the other hand, specify how many assets they are selling and the lowest price they are willing to accept. If no existing trading meets their requirements, these orders are stored in the *limit order book* until their parameters are satisfied by a new order or the issuer cancels them. Understanding how liquidity influences these dynamics and vice versa is then important to conduct any form of analysis.
Other typologies of trading, apart from the more classical peer-to-peer one just presented, can occur. An example is the case in which an investor creates a short position. When adopting a short position traders are selling assets with the obligation of repurchasing it in the future. In the DeFi environment this type of trading strategy is made easier thanks to the atomic structure, a key element of the blockchain technology that will be presented in Section 3. Shorting strategies and all of their implications are a field of research by themselves. Because of this, further details are not going to be discussed in this project, but can be found in [7].

Markets, like securities, can be classified in a variety of ways based on their technical structure, how they operate trades, and the frequency with which trades are executed, among other factors. The first distinction that can be made is between primary and secondary markets. Newly issued securities are sold first in primary markets, then resold in secondary markets. Whatever category a market belongs to, it must possess the following characteristics:

- trades are made based on information, hence market's information like volume, previous prices, bids and offers should be available.

- Low costs for trading in the market.

- It should be liquid. Liquidity is defined as "*[...] ability to transact a large number of shares at prices that don't vary substantially from past prices unless new information enters the market*" [7].

Having introduced what securities are and where they can be traded, it is appropriate to try and understand how should an investor construct their portfolio. Each security that an investor can buy or sell, has an underlying risk element. *Risk* is a term used in finance to describe the degree of uncertainty or potential financial loss associated with a financial decision. For example, the risky component of owning an asset is caused by fluctuations of stock's value. Stock prices fluctuate because they reflect a company's performance, soundness, and dependability to investors. A sudden change in one of these characteristics could result in mass selling, lowering the value of the company's assets and impacting negatively investors financial wealth. Similarly, whenever an investor lends money, the likelihood that the contract might not be honoured, causing investors to lose the money they provided, measures the risk of that investing strategy.
From a mathematical standpoint, risk is due to the probabilistic nature of returns of each security [7]. The more a financial asset's possible outcomes (in this case, price variations) are dispersed, the more risk it carries. In 1952 economist Harry Markowitz introduced the Modern Portfolio Theory [9]. According to it investors can build optimal portfolios that maximize expected return[4] for a given amount of risk. Having an efficient portfolio for a set volatility[4] is crucial as each trader might have a different level of risk aversion. *Diversification* is another important component of an optimal portfolio that [9] emphasises. Although diversification cannot eliminate the risky component of assets, it can help to reduce volatility. This is because assets are likely to be correlated[4] with one another. The significance of Markowitz's findings can also be traced back to the fact that they apply to a wide range of scenarios involving risky assets, risk-free assets, and short positions.

It is understandable that an investor would desire to benefit without taking any risks. Although it is theoretically feasible to create a portfolio with no risk, doing so in practise is very difficult. Because it is extremely uncommon that two assets are negatively correlated. However, there are certain investments that do not involve any risk but guarantee a reward:

[4]definitions can be found in A.1

Definition 2.2.1 *An arbitrage is a transaction that involves no negative cash flow at any probabilistic or temporal state and a positive cash flow in at least one state [10].*

The absence of risk in this strategy is to be ascribed to the fact that investors do not need to invest their money as no negative flow is recorded. A more practical definition of arbitrage is "*the simultaneous purchase and sale of the same asset in different markets in order to profit from tiny differences in the asset's listed price*". It is worth noting that arbitrages are conceivable because short positions are viable as they allow the sale of assets that are borrowed to the investor.

2.3 Liquidity Providers

All of the above discussions are possible in settings such that the realization of a trade will not drastically result in leaps on assets' prices. In order to achieve that, markets need to be *liquid*, both in continuity and depth. Definitions for these concepts are provided in [7] and are:

Definition 2.3.1 *In a market that has liquidity continuity "[...] an investor can expect to transact some shares at prices close to those at which the security recently traded"*

Definition 2.3.2 *A deep market is one in which "[...] a large number of shares can be transacted without a substantial change in price"*

It is clear then that liquidity providers have a central role in making a market viable. But what is the definition of *liquidity provider*?

Definition 2.3.3 *A liquidity provider is a financial institution that acts as a middleman in the securities markets. [11]*

In practice, providers purchase huge quantities of securities from issuers in primary markets where minimum trades' sizes are conspicuous and transfer them in batches to financial institutions in secondary markets who then make them available to regular investors [7]. Because of this characteristics liquidity providers (LPs) are also called market makers.

Opposed to traders, LPs do not seek profit through coupon collection or re-selling of securities as they are the one that sell and buy the securities necessary to make those kind of transaction possible in the first place. Nevertheless providing liquidity entitles LPs of a percentage fees when trades in a market are completed, as a reward for their commitment to the market. Therefore their strategy is not directly comparable to traders' one. Changes in securities' values (either increasing or decreasing) imply a loss for LPs, either because they have not been able to extract all the intrinsic value of a security or because they hold one that has depreciated. This loss is referred to as *impermanent loss*. It is obvious then that it is in markets' interest to find the optimal settings to attract both liquidity providers and traders to operate within it. Moreover, providers enhance many other different strategies that guard them from losing money, like hedging[5].

[5] A hedge is an investment that is made with the intention of reducing the risk of adverse price movements in an asset. Normally, a hedge consists of taking an offsetting or opposite position in a related security.

Decentralized Finance

3

3.1 Cryptomarkets

As interest in and use of cryptocurrencies grew, so did the demand for platforms where these new assets could be sold and bought (Figures [1.2]-[1.3]). Numerous exchanges have been set up to allow customers to trade fiat currencies[1] for digital currencies. To distinguish them from traditional exchanges, these markets are referred to as *crypto exchanges*. They can be divided into centralized exchanges and decentralized ones. Because they keep digital order books, centralised exchanges (CEXs) work similarly to traditional asset exchanges like stock exchanges. They are referred to as *centralised*, since they are usually owned and managed by companies [13]. Decentralized Exchanges (DEXs), on the other hand, have a different implementation structure, which typically is not the order book structure, and they are designed to prevent direct manipulation or control by system administrators. These features make DEXs more transparent, which is important to attract investors. CEXs include Binance, FTX, Coinbase, Kraken, and others, whereas DEXs include Uniswap, SushiSwap, Balancer, and Curve. The latter are the ones that researchers are most interested in because they offer an intriguing alternative while also posing numerous mathematical, financial, and software engineering challenges. But what are blockchains, cryptocurrencies and their characteristics?

3.2 Bitcoin

Satoshi Nakamoto initially suggested the notion of a peer-to-peer system in 2008, allowing anyone to conduct money transactions in a trustless manner, eliminating the necessity for intermediaries as in traditional markets [5]. In Nakamoto's proposal, the system is shown as a chain of blocks, each containing

[1] Fiat money is a government-issued currency that is not backed by a commodity such as gold [12]

information that is used to generate the next block but also for validating all previous ones. The system should be run on a large number of computers each holding a copy of the chain helping to validate and proposing new block. These computers are also known as nodes in the system. All transactions should be made public for all nodes to see in order to eliminate the possibility of duplicate spending on the chain. This means that the transaction history of a coin can be recorded and compared to the public key wishing to transfer it to ensure that the public key is the true owner of the coin. A technique known as Proof-of-Work is used to keep track of time on the chain. This protocol works by requiring nodes, also known as miners, to perform some amount of work in order to suggest the next block. Figure [3.1] shows an example of this, from [5]. As it can be seen, each block contains a variety of data, including a Previous Hash, a Nonce, and transactions. The Previous Hash value is used to convert the contents of the previous block and Nonce into a series of numbers with a specific number of leading zeros. The number of zeros required to make a valid hash is determined by the chain and is an expression of the difficulty, or amount of effort, required to win/unlock the next block; each required zero increases exponentially the amount of work and thus time required to make a valid hash. When a miner wins a block, the miner broadcasts the new chain; if this is the longest chain, the block will be added if all transactions in it are legitimate. This means that in order for someone to alter a previous block, they must be able to: change the block, redo all of the work done after it, and propose a new longer chain for approval. A chain attacker would need significantly more computational power than the other nodes in the chain to accomplish this. To incentives the running of nodes in the systems miners are awarded an amount of bitcoin[2] every time they win the newest block.

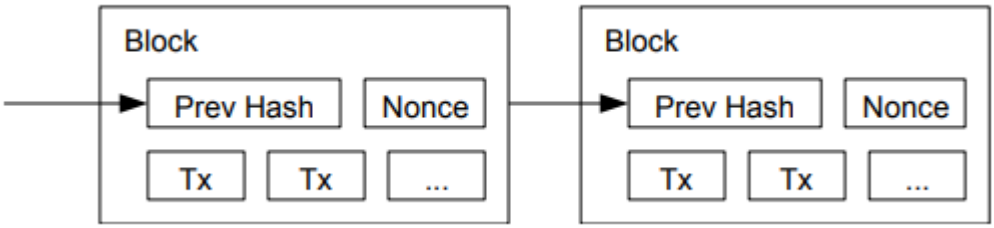

Figure 3.1.: Depiction of a block chain.

[2]Conventionally, Bitcoin, with a capital b, refers to the technology and the network, while bitcoin is used for the currency

3.3 Ethereum

The very same concept of a distributed consensus tool is what gave birth to the Ethereum chain, which was disclosed in 2014, leading to its launch in 2015 [14]. This is a far more advanced chain architecture that includes a built-in Turing-complete programming language, value-awareness, blockchain-awareness, and state. As Ethereum is currently hosting the largest AMM (Uniswap, as seen in Figure [1.3]), it is important to understand the intricacy's of the chain as this can, to some extent, explain how users of AMM's on this chain act. This chapter aims, based on Ethereum whitepaper [14], to introduce these intricacy's and how that makes AMMs different from the traditional exchanges.

Block time. On the Ethereum blockchain the block time, *i.e.* the time it takes to mine a new block, is expected to be 14 seconds. Although time might vary from one block to another, on average it is going to be the expected value: even if nodes' computational capacities increase the blockchain will adapt the difficulties to mine new blocks in order to keep a block time of 14 seconds. This means that transactions on the chain can only happen every 14 seconds, differing from the high frequency trending taking place on traditional exchanges, where algorithms claim to trade in intervals of less then 14ms [15]. Furthermore, each block can only hold a certain number of transactions. Since these are used not only to execute transactions on AMMs but also by the entire blockchain community, the number of trades that can be made are greatly limited when compared to traditional markets. In contrast to conventional markets, where each order is processed according to the time it is received by the server, on blockchains miners who win the block are responsible for building and ordering the block.

Accounts. Ethereum's state is made of objects called *accounts*. Accounts can be divided between externally owned and contract accounts. These accounts each have a 20-byte address and four distinct fields:

- A *Nonce* similarly to Bitcoin to make transactions distinguishable.
- A *Ether balance* containing balance of the accounts balance.

- The storage hold by the account, if any.
- The contract code, in case of contract account's

External accounts are used by chain users to transmit messages by first generating and then signing transactions. These messages may include information that can be used to contact a contract account. Contract accounts, also known as *smart contracts*, are pieces of code that may be placed on the blockchain. This code will then sit and wait for messages before starting its execution.

Gas Fees To execute a transaction, an external account must hold some Ether as this is the currency that drives the Ethereum Blockchain. Ether is payed to the miner for their service. This fee is known as *gas*, which is priced in gwei with 1 gwei corresponding to 0.00000000106 Ether. The cost of gas for a transaction is set to be roughly proportionate to the amount of computing labour required to execute the transaction. In general a computing step costs 1 gwei, but more sophisticated processes have a higher gas cost. Furthermore, the external account is charged 5 gwei for each byte of data in the transaction. This concept of gas is employed not only to compensate miners, but also to avoid strategies that aim to deny access to the service. Because attackers would have to pay the miners in proportion to the amount of disruption they cause, major interruptions to the system are prohibitively expensive. However, when a high number of users are interested in completing transactions, the limited number of transactions that can be completed in a block causes a bottleneck in the system. Due to the bottleneck, gas costs rise causing each transactions to become prohibitively expensive, as seen in Figure [3.2].

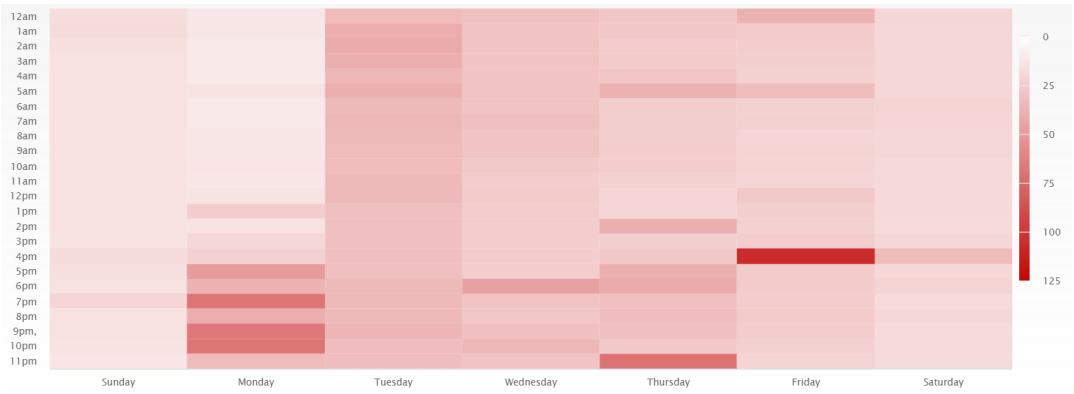

Figure 3.2.: Graph of gas price shown in gwei in the period May 1st - May 7th 2022, as from [16].

Transactions composition. Moreover, when an external account sends a message, this must be properly formatted to be accepted as a transaction. On Ethereum a transaction is defined as a signed data package including a message to be sent. The following five fields, plus an optional sixth, must be included in a well-written transaction.

- The address of the message recipient.
- The senders signature that identifies them.
- The amount of ether to be transferred from sender to receiver, which can also be zero.
- A STARTGAS value, restricting the amount of computational steps the transaction can initiate. It is used as part of the anti-denial of service model to avoid users from creating infinite loops on purpose or on accident.
- A GASPRICE with is the amount of transaction fee the senders pay per computational step.
- An optional data field. This field can be accessed by contract accounts and parses specific parameters for the computation to respect.

Token standards. The ability to run code on the Ethereum blockchain has sprouted the development of a wide range of objects known as tokens. A variety of standards have been established to assure interoperability and simplicity of integration, into other smart contracts, when generating these token types. Some of these are standards are:

- ERC-20: a standard for Fungible Tokens. It ensures that each Token has same exact properties, meaning that every Token is exactly the same as any other Token and will always be.
- ERC-721: a standard for Non-Fungible Tokens, or NFTs. Tokens implementing this standard are unique to each others and they can have different values even if originated from the same smart contract.

- ERC-777: an extension of the ERC-20 standard. That optimize transactions with tokens by allowing these to be approved and transferred in a single call where it require a double call in the ERC-20 standard

- ERC 1155: a standard for semi-fungible tokens that can combine the ERC-20 and ERC-721 allowing a single token to use the function of both Fungible and Non-Fungible Token at the same time.

The ERC-20 is the most common standard for Tokens used in DEXs' smart contracts. Nevertheless Tokens may have some different attributes that make them stand out from other ERC-20 Tokens. An example of such, is the DAI token, which is known to be a *stablecoin*. Stablecoins are coins that implement a form of token minting that helps regulate its price. DAI is regulated to have a value of almost 1:1 with respect to the US dollar. This type of coins can be compared to Treasury bills in the traditional market as they have, by design, a very stable value and hence hold no risk of depreciation. Another ERC-20 Token is WETH. This token is what is called an *altcoin*. Altcoins are usually linked to other tokens' value that do not comply with ERC-20 standards. In case of WETH, it is linked to ETH, the native coin of the Ethereum network, which is not an ERC-20 token itself. Because of this ETH cannot be used in DEX's smart contracts. This said, it should be noted that efforts are being made to update ETH and push a new standard for tokens that should include this updated version of the ETH [17].

Atomic structure. When evaluating transactions on the Ethereum chain, it is critical to remember that all executions are atomic. This means that given a sequence of computation in a transaction, one of two things can happen: either all computations are legitimate and successful, or one fails, forcing all previous computations in that transaction to be rolled back. This characteristic may be effectively used, and it has enabled a number of procedures that would otherwise be considerably more difficult or impossible to do. One of these activities is known as flash loans, in which an external account can loan a number of tokens from an exchange in a single transaction, utilize them, and pay them back in the same block. This allows users to trade tokens they do not even own in a completely risk-free manner: if the trade is not profitable because it does not pay back more than was given in, it will simply

not be executed, costing the external account only the amount of gas that was required to compute the trade.

Proof-of-Stake. Ethereum is currently running the same Proof-of-Work protocol as the Bitcoin chain, but has suggested a change to combat the excess amount of power used by nodes working to run the chain [18] as well as combat other problems that have been discussed or will be in Chapter 4. This transition, known in the Ethereum community as *The Merge* [19], will change the Ethereum chain to be a full Proof-of-Stake chain. Meaning that some factors such as front running and gas costs influencing the AMMs on the current chain my be reduced significantly. Ethereum states that this change will bring a number of improvements, some important being:

- Lower energy cost as mining blocks will no longer be needed.
- A lower hardware requirement reducing the barrier of entry for nodes to run the chain.
- A more decentralized chain as the lower requirement should increase the number of independent nodes.
- Stronger support for sub chains also known as *shard chains*, which will be important for improving the number of transactions the chain can process.

These changes are important updates that target making the Ethereum chain a more appealing environment to host trading markets. The Proof-of-Stake achieves all these changes by multiple adjustments to the chain. First it will be introduced a main chain known as *the Beacon chain* along with 64 shard chains. This will greatly increase the throughput of the main chain as each of the shard chain will be able to make the same amount of computations and storage as the single "old" chain is currently, but with 64 of these running, the total output will be significantly higher. Moreover, instead of having miners running node, users can buy stake on the chain awarding them the role of *validators*. When a new block is to be created, a validator is randomly appointed of putting a valid block together, while the rest of the validators are to confirm that this new block is valid. If a validator fails to be active, they can be punished by

losing part of their stake. Moreover if they are found to collide by attesting incorrectly a block of being valid, they will lose their entire stake. For each of the shard chains, validators are chosen to form the *committee of 128 validators* that are to validate the following 32 slots. Those will be combined at the end and put on the main chain as a single block. After the 32 slots have been completed, all committees are then dissolved and new ones are chosen randomly. This process keeps a signal shard from having a set of malicious committees. The new system rewards validators in a similar manner as the Proof-of-work protocol does. First by being chosen as a validator and secondly via the gas fees that each block pays.

4 Automated Market Makers

In the past decade the blockchain technologies have evolved and their utilization has been seen in a plethora of different fields. One that has seen a constant growth in interest is the cryptocurrencies markets within the financial field, due to a list of factors. Firstly, because practicalities (e.g., price determination) can be completely automated, as demonstrate by early implementations of decentralized exchanges [20]. Moreover, as stakeholders of the blockchain are required to maintain a singleton state state machine, operating financial securities on this technology ensures that no direct manipulations are possible. These characteristics promoted the implementation of exchange of assets on permissionless blockchains[1] by the conventional central limit order book (CLOB) design. Nevertheless, this proved to be expensive and infeasible on a large scale: each order requires the computation and validation of the whole chain resulting in an highly inefficient and costly structure [20].
To overcome these issues Automated Market Makers, a new type of markets, have been presented. They have seen a surge in popularity and utilization over the past two years as seen in Figure [1.3]. Their structure effectively eliminates the necessity of an order book structure, as in traditional exchanges and in the first iterations of blockchain based exchanges. Liquidity is collected in pools made by pairs or groups of tokens that are available for users to trade immediately at the current price. The reasons for which a trader might want to operate are mainly two: he or she intends to use a digital currency a medium of exchange for a good or a service, or he or she thinks that the currency will increase in value in the future. This other option is due to the fact that, unlike fiat currencies, pure digital tokens face a high volatility as shown in Figure [4.1]. Hence, these cryptocurrencies have to be considered operationally as risky assets. Despite the relevance of Modern Portfolio Theory in traditional markets, [22] argues that Markowitz's approach is not suitable to the examination of crypto portfolios. The main reason being that returns of the assets in the portfolio are not normally distributed, a prerequisite for the Modern

[1] open environments accessible by all, opposed to permissioned ones that are accessible only by parties recognized by a system administrator [21]

Portfolio Theory. Because of this the implementation via the Ω-measure is preferable for optimizing a portfolio that beholds digital assets.

Figure 4.1.: Source: Coinmarketcap.com

In creating AMMs different approaches have been undertaken and implemented [23]. Major AMM protocols are: Uniswap, Balancer, Curve, DODO with unique features [23]. Although they present substantial differences in their structure there is one key element that is crucial for all of them: the *invariant* and the relative conservation function, also called bonding curve. The centrality of this element is due to the fact that, as different tokens are traded, the output amount is computed algorithmically following the bonding curve and hence the invariant. AMMs can be divided into Constant Sum Market Maker (CSMM), Constant Product Makers (CPMM) and Hybrid Constant Function Market Makers [24].

Constant Sum Market Makers. For a CSMM, the invariant k is defined by the amount x of token A and y of token B that are available in the pool accordingly to

$$x + y = k$$

Computing then the output amounts is a straightforward process: say Alice trades Δx, she would receive (in the case where no fees are applied):

$$(x + \Delta x) + (y - \Delta y) = k \implies \Delta y = \Delta x$$

Of course a market that follows such an invariant would encounter the possibility of having one reserve drained, making the pool infeasible as it is not able to meet markets' demand. Because of this, CSMMs are feasible when trading

stable coins, as their value are expected not to differ significantly over time [24][25].

Constant Product Market Makers. This type of AMM are characterized by a conservation function that is given by a (weighted) product of the reserves. For example Uniswap V2 has the following bonding curve [26]

$$x \cdot y = k$$

Trading amounts are computed as follows

$$(x + \Delta x) \cdot (y - \Delta y) = k \implies \Delta y = y - \frac{k}{x + \Delta x}$$

Another example of how CPMM can be implemented can be found in Balancer's invariant function

$$\prod_k r_k^{\omega_k}$$

where r_k is k-th asset's reserve, and ω_k its weight in the pool [27]. It is clear that in this cases the output amount not only depends on the input amount (like in the CSMM), but also on the reserves of token A and B in the pool. This characteristic will be presented and discussed later in this section. Because of this peculiarity of CPMM, it is not possible to drain a pool of a specific token as the exchange rates dynamically adjusts to the reserve present in it, contrary to the CSMM model.

Hybrid Constant Function Market Makers. Some implementations have tried to combine the two approaches above presented, as they have different benefits and downsides. An example is Curve, that derives its conservation function as follows

$$An^n \sum_{i=1}^{n} r_i + D = ADn^n + \frac{D^{n+1}}{n^n \prod r_i}$$

where A is the leverage constant when the portfolio is balanced [28], D is obtained by

$$\begin{cases} \sum_{i=1}^{n} r_i = D \\ \prod_{i=1}^{n} r_i = (\frac{D}{n})^n \end{cases}$$

and r_k, $k = 1, \ldots, n$ is k-th asset's reserve in the pool.

4.1 Uniswap V2

4.1.1 Structure

Uniswap's core element is what is called a pool: a pair of two different tokens that can be traded. Whenever a pool is created, parameters for that specific pair are automatically set. The invariant k is obtained by multiplying the two amounts of token provided [26], while the spot price is determined [23] by the ratio between the two reserves

$$x \cdot y = k \qquad\qquad P = \frac{x}{y}$$

The role of the invariant is to determine algorithmically the output amount base on current reserves. Because of this, k is set not to change after trades[2] as it acts as an indicator of supply and demand, what is updated is the spot price for the considered pair.
At the same time, actions like liquidity provision or withdrawal are set not to impact spot prices, as they do not reflect any change in demand or supply. In order to achieve that, provision has to respect the pair specific proportion and will in fact update the invariant k, but keep the spot price fixed.
So it is clear that different actions will impact pool's parameters in various determined ways. It can be summarized as follows:

Action	Input	Updated Parameter
Trade	$(\Delta x, 0)$	$\tilde{P} = \frac{x + \Delta x}{y - \Delta y}$
Liquidity provision	$(P \cdot \Delta y, \Delta y)$	$\tilde{k} = (x + P \cdot \Delta x) \cdot (y + \Delta y)$

From the formulas above it should be clear that trades' output amount are highly correlated to reserves and not uniquely to spot prices. Whenever a trade takes place, it modifies the balance of reserves resulting in a new price. Since this behaviour is not due to fees the case where no fees are applied is presented Consider the following pool:

[2] in practise this does not apply due to the payment of fees

Reserve USDC	Reserve ETH	Invariant k
10000	300	3000000

If Alice wants to trade 100 USDC she would receive

$$\Delta y = y - \frac{k}{x + \Delta x} = 300 - \frac{3000000}{10100} \approx 2,970$$

which is lower then the spot price $\Delta\tilde{y} = \frac{300}{10000} \cdot 100 = 3$ ETH.
This phenomenon is called *slippage* and it is due to the nature of the conservation function and to the fact that the pricing algorithm acts in two steps. First the input amount is added to the pool, second the output amount is computed with the updated values. The more one reserve is imbalanced compared to the other the higher the slippage, as it can be seen in Figure [4.2].

Moreover this phenomenon has a larger impact when trading amounts are

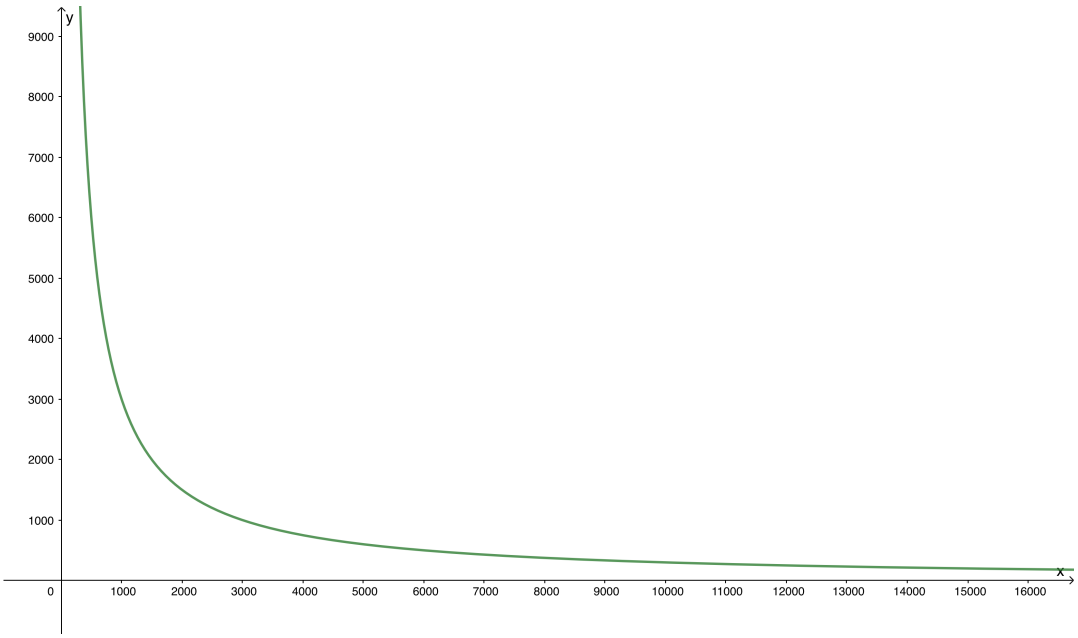

Figure 4.2.: Conservation function of Uniswap V2

of the same order as those of the reserves in the pool. Consider the same exact trade order as above in a pool with the same spot price, but different reserves:

Reserve USDC	Reserve ETH	Invariant k
10000000	300000	3000000000000

For her 100 USDC Alice would receive in this case $\Delta y \approx 2,99997$, an almost 1% difference. It is clear at this point the magnitude of the correlation that exists between reserves and slippage.

In fact, slippage is one of the biggest factors that traders have to take into account when operating within Uniswap as well as many other AMMs. Although it would appear that dividing a transaction into smaller ones might help mitigate this phenomenon, as smaller transactions are less prone to experiencing this phenomenon, it should be noted that even in the no fees condition this is not the case. This is due to the convexity of the conservation function [29]. Moreover in the real case scenario, dividing a transaction into smaller ones would heavily impact the final output amount as fees and gas prices have to be paid multiple times. Although slippage is commonly viewed as a negative component since it disincentivizes trades, it may be argued that it is in reality a crucial feature of CPMM because it disincentivizes only large trades that have a significant impact on price balance.

4.1.2 Fees Collection

The following section is based on [26] and presents how Uniswap V2 collects fees and the reasons for this implementation.
Agents who provide liquidity are eligible to receive *liquidity tokens*. These tokens keep track of how much an LP has contributed to a specific pool. It would be costly to distribute fees as soon as they are collected because every transaction from and to Uniswap's pools is on-chain. Rather, fees are collected and then distributed whenever a liquidity provider performs an action, such as providing to or withdrawing from the pool. This occurs because these actions affect the percentage of liquidity tokens each provider holds (as liquidity tokens are minted or burned) in comparison to the pool's reserve, making it impossible to reward each LP with the correct amount later.
But how are the fees collected, and how does each pool know how much each liquidity provider is owed? At first glance, it appears to be simpler to set aside the fees in a separate reserve, implying that the pool should have a

wallet containing the specific amount of tokens collected as fees. For liquidity providers, this solution has one major flaw: if the value of token A rises against token B, traders will be more likely to put tokens B into the pool in order to receive tokens A. Providers would receive fees in the form of tokens B, whose value has decreased outside the pool, resulting in a two-fold impermanent loss. Fees are collected as liquidity tokens and then returned as a pair to avoid this. This should protect liquidity providers from a sudden change in the value of a particular token. This is an important factor to consider when analysing how the value of LPs' portfolios changes.
The next question is: how does the pool keep track of the fees collected in liquidity tokens?
To accomplish this, a technique has been created that appears to go against the fundamental premise of the constant product market maker, but is yet incredibly elegant. When a trade occurs, the output amount is determined using the constant product model, with the input amount being the amount after fees have been deducted. In a situation where the fees are given by $(1-\gamma)$, if an investor trades Δx, the output amount Δy is computed starting from $\gamma\Delta x$ instead. In summary,

$$\Delta x \implies \gamma\Delta x \implies \Delta y = y - \frac{k}{x+\gamma\Delta x}$$

But the pool receives Δx instead of $\gamma\Delta x$, and hence

$$(x+\Delta x)\cdot(y-\Delta y) = \tilde{k}$$

where $\tilde{k}$ is the updated constant after each trade and $\tilde{k} > k$ [29].
Since the amount of liquidity tokens l is given by $\sqrt{k}$, it is possible to compute how many fees have been collected in the form of liquidity tokens via

$$\frac{\tilde{k}}{k} = \left(\frac{\tilde{l}}{l}\right)^2 \implies \tilde{l} = l\sqrt{\frac{\tilde{k}}{k}}$$

It should be noted that the amount of liquidity tokens l updates after each trade. Which means that after the *n-th* trade

$$l_n = l_{n-1}\sqrt{\frac{k_n}{k_{n-1}}}$$

By recursion

$$l_n = l_0\sqrt{\frac{k_n}{k_0}}$$

where l_0 and k_0 (l_n and k_n respectively) are the amount of liquidity token and the constant product variable after a LP has either *provided* or *withdrawn* (the amount of liquidity tokens and the constant product variable after n **trades** took place, respectively).
Therefore the amount of fees collected is given by $\Delta l = l_n - l_0$. This must be spread among liquidity providers via a burning operation that restores k to its original value prior to trading. The structure of the AMM does not change as a result of how this solution is implemented, and the pools' fundamental functionalities are preserved.

4.1.3 Agents

Despite having two main type of actions, agents that interact with the market can be divided into three macro categories: traders, arbitrageurs (or swap agents) and liquidity providers. Each of them have different strategies and hence different behaviour can be noticed.

Traders. As per [30] transactions made by traders, also called *uninformed traders*, can be considered to be independent from each other and follow no set rule [31]. It can be assumed that their objective is to trade token A for token B in order to operate on a platform that strictly requires token B. Another reason that would prompt traders to swap one coin for another is that they might want to hold various digital currencies in their portfolio, although as discussed in [22] classical Markowitz Theory does not apply to crypto assets portfolios. Hence, it is reasonable to consider that there is no set optimal strategy for this type of agents. It is also plausible to assume that this category encompasses the vast majority of protocol transactions.

Arbitrageurs. Another kind of agent that can be observed is what is called an *arbitrageur*. Their strategy is to exploit prices differences or unbalances in different markets or even within protocol's pools. Arbitrageurs are set into their own category as they act in a more informed way with the sole goal of making immediate profit. Because of this they can be also referred to as *informed traders*. Their strategy requires to collect information from multiple

sources: different protocols, different markets and even different blockchains which is done running complex algorithms on the blockchains state data. Even if having arbitrageurs interacting with the pool and exploiting it by removing liquidity might appear detrimental, in fact they act as a balancing element. Their actions set prices to the equilibrium ones either influenced by other markets or by influencing them [31]. The fact that this action has a positive effect for the protocol is due to the presence of slippage: an arbitrageur will never be able to drain one pool, because this would require an infinite amount of a set token as the price would dynamically adjust when the request is submitted.

In case arbitrageurs can trade with an external market that trades at a fixed spot price (*de facto* it is assumed to be infinitely liquid), hence in the case $\Delta^{'}y = P_m \Delta x$ where P_m indicates external market's spot price, the best optimal amount to trade with the pool that has tokens (x, y) is

$$\Delta^{*}x = \left(R_x - \sqrt{\frac{k}{\gamma P_m}} \right)_+$$

where R_x is pool's reserve of token x, k is the invariant, $(1-\gamma)$ is pool's fee and $(z)_+ = max\{0, z\}$ [29]. The above result does not give any information about the size of the arbitrage as it only ensures a profit opportunity. Hence, an arbitrage might not be present if its return is less then what arbitrageurs are required to pay in gas fees. This said, it should be highlighted that the optimal arbitrage amount does not change.

The previous case relates to when a gap between pool's spot price and external market's one is present. Another possibility can be seen when within the protocol two pools have different spot prices. In this latter case the optimal amount to be traded is

$$\Delta^{*}x = \frac{-R_y^{'} R_x + \gamma\sqrt{k k^{'}}}{\gamma R_y^{'} + \gamma^2 R_y} \tag{4.1}$$

where $(R_x, R_y), (R_x^{'}, R_y^{'})$ are the reserves of token x and y in the two pools, $k, k^{'}$ are pools' invariants and $(1-\gamma)$ is pools' fee (hence equal for both pools). This opportunity is unlikely to happen among two different pools with the same pair of tokens as it would be immediately arbitraged.

It should be noted that the two different opportunities have different impacts of protocol's overall liquidity. In the latter case liquidity is simply moved from one pool to another, leaving protocol's liquidity stable whilst increasing profits

made due to fees. In the first case, part of the liquidity is withdrawn from the pool leading to a partial loss in the total value locked within the AMM.

Miners. Once an action is submitted to be added to the blockchain it has to be selected and then validated by the so called *miners*. Miners' duty is to check that the requested actions respect the whole state of the blockchain. This effectively prevents invalid transactions to be executed by controlling the whole history of the chain. Hence miners can be in some way compared to brokers. Since validating and adding a block to the chain is a costly procedure, miners will prioritize those actions that are willing to pay a higher validation cost, or gas price. It is clear then that miners have the possibility of accessing the requests they are submitted and because of this they have a favourable position: they might exploit this information to profit acting as arbitrageurs within the blockchain but as they control the ordering of transactions being executed they are also able to "sandwich" trades using slippage on the AMM's to make profits on smaller arbitrages that would not be available non-miners. This profit for the miners is was is known as miner extractable value (MEV), and the behaviour has been highly researched and empirically analyzed in [32]. As many possible solutions to this issue have been presented and proposed for implementation, this project does not aim to further discuss this specific agent, as it can be labeled as an informed trader.

Liquidity Providers. In order for a pool to be created, an unspecified amount of two tokens have to be provided to the protocol. *Liquidity providers* are the agents who carry out this action. To keep trading volumes high, the protocol pays out fees collected from trading actions to agents who provide pairs of tokens. As a result, liquidity providers profit from the fees traders pay to place orders in the pool. Contributing assets to the pool in exchange for shares of fess can result in them losing potential value due to price movements within the protocol when compared to holding the assets outside it. Because the depreciation of the pool value disappears and reappears as price moves, this effect is known as *impermanent loss* or *divergence loss*, and it is only realised when assets are actually removed from the pool [33][23]. This phenomenon is visualized in Figure [4.3]. An optimal strategy for liquidity providers would be to withdraw their invested tokens when the spot price matches the value at which they have invested, avoiding impermanent loss while also profiting from the fee collection.

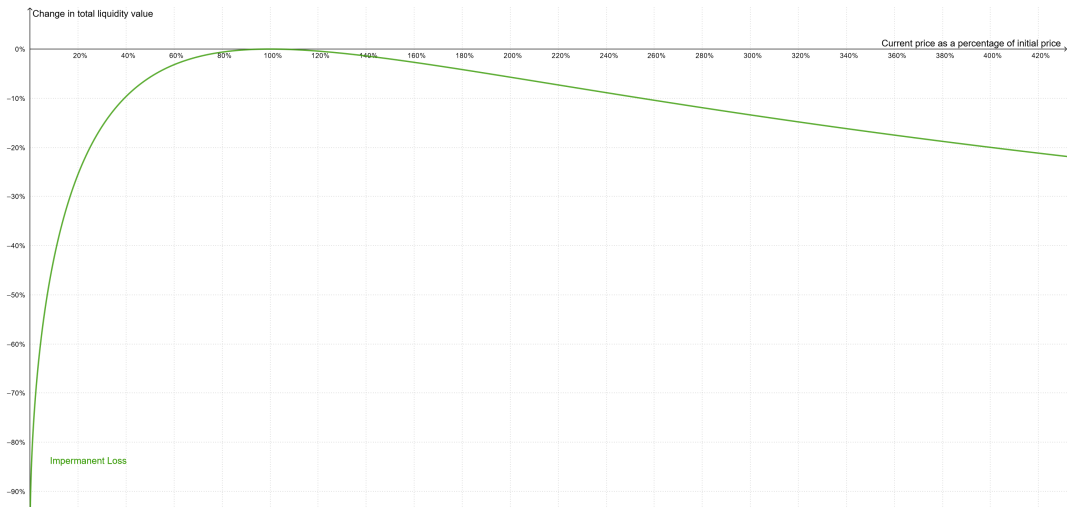

Figure 4.3.: Impermanent Loss as presented in [33].

Unlike traditional markets, where liquidity providers are typically large funds or institutions due to the minimum requirements to place trades in primary markets, the DeFi environment does not limit liquidity provision to large amounts. This allows small-cap investors to become liquidity providers. However, liquidity providers with large portfolios are common in the AMM environment. It is then required to analyze how this two type of LPs are subject to impermanent loss and how they profits move in relation to trade volumes and price movements.

5 Implementation of simulation

The simulation environment has been build as a digital twin of Uniswap V2 using Python version 3.8.10. Is has a number of required library's that it relays on to run. These can bee seen in the *requirements.txt* file. Further the repository have a *setup.py* file included. This file can be run to generate the working python environment that has been used to produce all the results to come in Chapter 6. The implementation aims to replicate the real exchange as closely as possible to how it is documented in the AMM's whitepaper [26]. It allows users to dynamically build and simulate how different scenarios of actor behaviors effect the system. The different actors implemented are Trade Agent, Swap Agent, Liquidity provider Agent and Pool Agents (see [B.4] for there full implementation). By combining these the system can be tuned to replicate trades, prices and provisions seen in the real Uniswap-V2 environment. The implementation of the simulation environment and the actors existing in it will be presented in the following sections.

5.1 radCad

RadCad is a powerful framework used for dynamical systems modelling and simulation[1]. The framework is a Rust implementation of another framework called cadCad. For the implementation it has been decided to use radCad over cadCad due to it being a more flexible and dynamical system where actors are created as instances of classes, a property not available in the the cadCad framework. The overall structure of the simulation is presented in Figure [5.1]. Squares represent the setup and running phases of the simulation , while ellipses represent Object instances that can be changed and updated to fit a given simulation parameters.
Another key functionality of radCad is that it allows to run simulations using

[1]Information on the framework can be found at https://github.com/CADLabs/radCAD

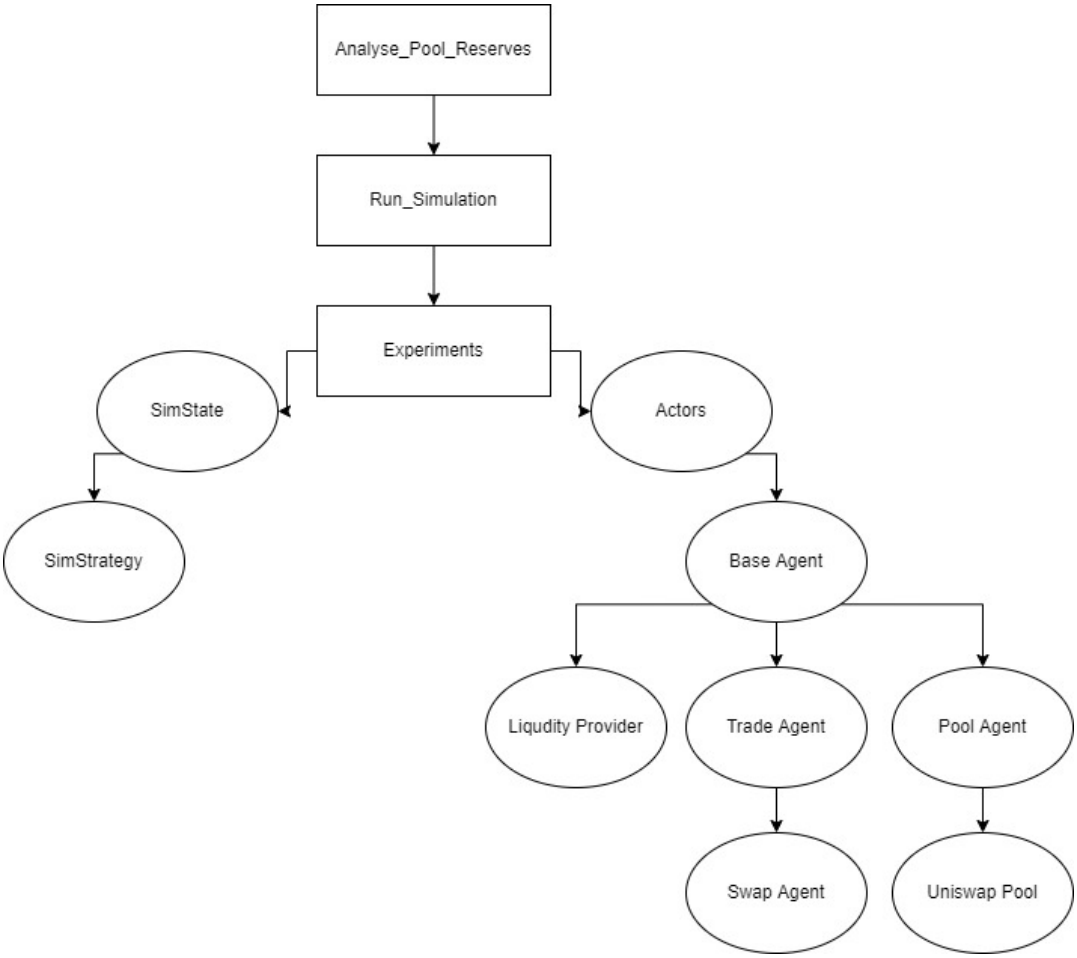

Figure 5.1.: Diagram of the overall simulation structure.

different initial parameters making is a powerful tool when it comes to exploring the impact that those might have on the system and the simulation. Moreover, radCad allows for multiple Monte Carlo runs. This is vital feature to have at disposal when analyzing a system that relies on any form of randomness as a single run might be a outlier due this randomness. Thus doing multiple Monte Carlo runs grants the option to inspect how variations of noise trades and timing of these impact the overall result of the simulation.

5.2 Data

To construct a realistic simulation of the Uniswap-V2 ecosystem, a link to the outside world will be required, as presuming that this lives in a vacuum would be naive. This relationship is made possible by a set of historical price data. The data utilized in this implementation is minute data, although the

design enables for this to be changed if different data is given. The method requires a .csv file called xxxyyy for each unique signal pool pair, where x and y must equal the token symbols in the simulation. The file must have a unix timestamp as well as the exchange rate at that moment. When the SimState is first loaded, it parses the .csv file into a numpy array, which is then used to update the signal price whenever the simulation reaches the time equal to the next unix timestamp.

5.3 SimState

The SimState is built to hold information that must be available to all actors in a given simulation such as time, available tokens, gas price and signal prices. The full implementation of the Simstate can be found in Appendix B.3.
The SimState is never updated by a single actor action, but by their policy functions which store all the actions executed. This is done to reduce the dependency of actors to the SimState. The main function of the SimState is to keep track of time. This is accomplished through the SimStrategy specifying how long each single tick of the state moves the simulation in time. This is important to ensure that the simulations time can be matched to real world events such as signal pricing. When first initialised, the SimState will start the first tick at a UNIX time. In all experiments conducted in Chapter 6 the starting time 1646092800, that translats to 2022-03-01 00:00:00, has been used. Below in the code snippet of the SimState, the definition of the *takeStep* function can be seen. It is clear from the code that, no matter what a sim tick corresponds to, either a second or a minute, prices are only updated every corresponding minute in order to reflect the empirical data collected. This can be changed to suit the data used by changing the *constants.S_Per_MIN*.

```
1 def takeStep(self, agents) -> None:
2     self.tick += 1
3     minuts_elapsed = np.floor(self.tick * self.ss.time_step / constants.
          ↪ S_PER_MIN)
4     if minuts_elapsed > self.minutes_elapsed:
5         self.minutes_elapsed += 1
6         self.update_prices()
```

Moreover as the SimState acts like a storage center for the information used by actors, this is done to reduce impact on actors when parameters are changed.

This means that changing a parameter in the SimState will propagate the change to all actors. Parameters that are used in this way are Gas fee, signal price and time between ticks. Gas fees are implemented to be static and as such will never change. This is done in order to avoid noise caused by large fluctuations in gas prices as that would make almost impossible to conduct a clear analysis. Liquidity Providers also access *pools_trade_volume* to make informed decisions for their provision strategy.

5.4 Actors

As seen in Figure [5.1], all simulation's agents inherit the functionalities of the Base Agent. Because of this they must either use base agent's functions or their own implemented versions. If an agent is to make any action that impacts either the SimState or other agents, it has to call for a policy. This then executes it and updates the state and other agents affected by that action. Because of this structure, the simulation has been implemented with an agent policy both for liquidity provision and trading. All the code implementing each actor and relative policies can be found in the Appendix B.4 along with their policies in Appendix B.5.

5.4.1 Base Agent

The base agent has two input variables: a name and a wallet. The wallet is used for keeping track of the different coins and amount that an agent owns. To interact with the wallet the base agent implements three functions: *getBalance*, *payToken* and *receiveToken*. These functions all take a Token Object as an input, while payToken and receiveToken require an amount in the form of a float. They then match the token to one in agent's wallet and update his or her wallet. The base agent also requires sub classes to implement a takeStep function which is responsible for making all agents' actions.

5.4.2 Pool Agent

Pool agents can be considered a contract accounts of the Ethereum chain and are built as a super class of the Uniswap pool. This allows for further

implementation of different pool types as they can be easily implemented with minimal changes to the existing code.
All pools are implemented with the *Uniswappool* sub-class that acts as described in [4]. All pools are made of a Pair Object which consists of two different Token Objects. A Uniswap pool thus contain: a name, a wallet, a Pair and a SwapFee parameter which sets pools trading fee. In the simulation a Uniswappool Object is updated in two ways:

- a trader that exchanges one token for another with such an action invoking a call to the *trade_update function* as presented in the snippet below. This function is given the updated amounts of token holdings, after slippage is taken into account.

```
1  def trade_update(self, token0: TokenAmount, token1: TokenAmount,
       ↪ liquidityAmount: TokenAmount):
2          if token1.token.symbol == self.token1.token.symbol or token0.
               ↪ token.symbol == self.token1.token.symbol:
3              if token0.token.symbol == "USDC":
4                  self.token0 = token0
5                  self.token1 = token1
6              else:
7                  self.token0 = token1
8                  self.token1 = token0
9
10             self.liquidityToken = liquidityAmount
```

- the pool is updated by a liquidity provider that either withdraws or provides liquidity. In this case the pool must either burn UNI tokens or mint new ones. This is done in the function *liquidity_update* that can be seen in the code snippet below. It must also reset the fees collected since last withdrawal or provision and distribute these as described in section [4.1.2]. This is all done in the liquidity provision policy.

```
1      def liquidity_update(self, delta0: float, delta1: float, deltaL:
           ↪ float):
2          ln = self.liquidityToken.amount * math.sqrt((self.token0.amount
               ↪  * self.token1.amount/self.invariant))
3          self.feesCollected = ln - self.liquidityToken.amount
```

5.4.3 Trade Agent

Because the simulation requires some kind of noise in order to function and run in a somewhat realistic manner, a new class of actors called Trade Agents have been implemented. These agents will not make informed trades like other agents, but will instead make stochastic trades designed to reflect the average buying client that operates on the Uniswap exchange, like suggested in [31]. These traders can be assigned a trade frequency parameter that controls how frequently they trade. To be able to fine tune trading these agents can be assigned to one, more or all pools when first initialized. This makes them able to trade only in assigned pools. The size of their trades is set to be proportional to the amount of holdings they have, ranging from 0% to 10% of their total portfolio and based on a normal distribution.

5.4.4 Swap Agent

Swap agents are built as a subclass of the Trade agent, which means they share the same function call but have distinct functionality. The swap agents will only conduct arbitrage, hence they will always execute zero or two trades in a sim tick. The swap agent does not implement a new takeStep function, as it inherits the trade agent's function. This then calls the *_tradePolicy* function that is implemented differently to how it has been done for the trade agent. The function runs through each known pool agent, comparing compatible coins and looking for a difference in price between them. This will result in a list of ratios, where each of those ratios reflects the price differences. The swap agent then proceeds comparing the optimal arbitrage amount to its holdings and computes if it is possible to collect a gain after fees, both the protocol one and the gas one, are being paid. If no arbitrage is available the swap agent will simply do nothing. One the other hand if an arbitrage opportunity is available, it will determine if one of the pool that it is trading with is a signal pool. This is done since swapping with a signal pool has a different optimal output amount than doing the same trade with a Uniswap pool. This is implemented as shown in the code below.

```
1  if "Signal" in pool_agent.name:
2      tokenPrice = buy_fromPool.pair.signalPrice[token.symbol]
3      amount = sell_toPool.pair.getToken(token).amount - ((sell_toPool.pair.
           ↪ invariant/ ((1-buy_fromPool.swap_fee) * tokenPrice))**(1/2))
4      bestAmount = TokenAmount(pool_agent.pool.pair.getToken(token).token, max
           ↪ (amount,1.0* 10**(-12)))
5      outputAmount, new_pair_token = pool_agent.takeSwap(bestAmount)
6      profit = self._getProfit(state, pool_agent, other_pool_agent, bestAmount
           ↪ )
7  else:
8      tokenPrice = sell_toPool.pair.signalPrice[token.symbol]
9      amount = buy_fromPool.pair.getToken(token).amount - ((buy_fromPool.pair.
           ↪ invariant/ ((1-sell_toPool.swap_fee) * tokenPrice))**(1/2))
10     bestAmount = TokenAmount(pool_agent.pool.pair.getToken(token).token, max
           ↪ (amount,1.0* 10**(-12)))
11     outputAmount = TokenAmount(pool_agent.pool.pair.getOppesitToken(token).
           ↪ token ,bestAmount.amount * tokenPrice)
12     profit = self._getProfit(state, pool_agent, other_pool_agent, bestAmount
           ↪ )
```

5.4.5 Signal pool

As previously stated, one of the key points in this simulation is to replicate trends that can be observed in the Uniswap protocol as closely as possible. As a result, it is crucial to include external factors that affect the AMM in ways that are not controlled by the AMM itself. The Signal Pool is used to implement such key factors, which in this case are outside prices. Only Swap Agents can interact with these pools while exploiting arbitrage opportunities. Moreover Signal Pools are considered to have infinite liquidity, hence to not be affected by slippage, making the Uniswap Pool the limiting factor in the size of an arbitrage. This is done because the data used for this outside pricing is pulled from a centralized exchange where prices do not face slippage but are constantly updated, similarly to a traditional high frequency trading exchange. Signal pools are all implemented in the same manner as the other pool agents. The only difference is that in the arbitrage policy, every tick of the simulation corresponds to a call to the *updatePrice* function as presented in the code snippet below. The function first resets the amount of non-WETH tokens. This is accomplished by first determining which of the two pool's tokens WETH is in (line 3), as pools have no restrictions on which coin must be which. Then,

in lines 7 and 12, the function changes the WETH amount to match the ratio provided by the eth_USD float. By doing so, the code ensures that the signal pool is never depleted in a realistic simulation, as trades of this size would never occur in such cases.

```
1  def updatePrices(self, eth_USD: float):
2      # 1 eth -> x USD thus we increase amount of USD in pool to reflect this
3      if self.token0.token.symbol == "ETH":
4          self.signalPrice["ETH"] = eth_USD
5          self.signalPrice[self.token1.token.symbol] = eth_USD**(-1)
6          self.token0 = TokenAmount(self.token0.token, 9 * 10**12)
7          self.token1 = TokenAmount(self.token1.token, self.reserve0().amount*
               ↪ eth_USD)
8      else:
9          self.signalPrice["ETH"] = eth_USD
10         self.signalPrice[self.token0.token.symbol] = eth_USD**(-1)
11         self.token1 = TokenAmount(self.token1.token, 9 * 10**12)
12         self.token0 = TokenAmount(self.token0.token, self.reserve1().amount*
               ↪ eth_USD)
```

5.4.6 Liquidity Provider

Liquidity providers are different than any other agents since they are the only agents that must be given a list of pool agents in their initialization. This list should contain all pool agents that the liquidity provider is allowed to provide to. Moreover, the list is used to create a dictionary of all available pools and the amount of UNI Tokens that a LP holds of each pool. Recall that the UNI token is used to keep track off the proportion of liquidity that a LP has provided in the pool. When a liquidity provider calls the *takeStep* function, it first checks if there are any liquidity pools that is worth providing to. How much the LP would expect to make in return is calculated in the code snippet below. It can be seen, in line 2, that liquidity providers have a variable called *rolling_window_size*. This declares how from how far back a provider will consider historical trading volume data relevant for making a decision. If this window is greater then a pool's history it will simply consider the pool's full history. The expected return is then calculated taking the mean over the given historical window and multiplying it with the pool's swap fee. This gives the provider an indication of how much it is paid in fees on average in a specific pool.

```
1  for agent in pools.values():
2      if len(state.pools_trade_volume[agent.name]) >= self.
           ↪ rolling_windows_size:
3          roi_agent = mean(state.pools_trade_volume[agent.name][-self.
               ↪ rolling_windows_size:]) *agent.pool.swap_fee
4      else:
5          roi_agent = mean(state.pools_trade_volume[agent.name])* agent.pool.
               ↪ swap_fee
6      roi_of_agents[agent] =  roi_agent
```

When the expected returns are computed, it proceeds to decide if the returns are high enough and if it has enough funds to provide. This is implemented as in the code snippet below, on line 4. If this is the case, then the LP makes a provision to the pool with the highest expected return. Otherwise it will consider if there is any pool in which it holds liquidity and that is not making enough profit, initiating a burning action to withdraw an amount of invested liquidity.

```
1      higest_roi = max(roi_of_agents.items(), key=lambda x : x[1])
2      lowest_roi = min(roi_of_agents.items(), key=lambda x : x[1])
3
4      if higest_roi[1] >= self.roi and my_wallet_usd > self.treshold:
5          amount = TokenAmount(state.tokenA, my_wallet_usd * random.randrange
               ↪ (15,20)/100)
6          higestPair = higest_roi[0].pool.pair
7          if amount.amount / higestPair.tokenPrice(higestPair.getOppesitToken(
               ↪ amount.token).token) < my_wallet_eth:
8              return (LPPolicy.PROVIDE, amount, higest_roi[0])
9
10     elif lowest_roi[1] < self.roi and self.liquidityToken[lowest_roi[0].name
           ↪ ].amount > 0.0:
11         amount = TokenAmount(state.tokenA, my_liquidity[lowest_roi[0].name].
               ↪ amount * random.randrange(15,20)/100)
12         return (LPPolicy.BURN, amount, lowest_roi[0])
```

6 Results

6.1 Simulation data

This project's goal is to make the simulation findings as relevant and comparable to the real Uniswap-V2 as possible. Data from the genuine Uniswap-V2 and form the CEX Bitstamp were collected and utilized. Etherscan[1] a sophisticated tool that allows users to scrape data from the Ethereum blockchain, was used to obtain data on Uniswap-V2 deals. While data from Bitstamp has been downloaded from cryptodatadownload[2] since it allows to retrieve day, hour, or minute data. Bitstamp was chosen for the signal data because it provided for quick access to minute data without requiring a professional api key. Moreover it was assumed that all CEXs have identical prices, that do not diverge more than a small amount one from another, since this would open them up to arbitrage. The project's time frame was set to be from March 1 to March 17 2022 as to maintain relevance in a rapidly changing market. Although, ideally, the time frame would be expanded to provide additional insight into the broader picture. When choosing relevant pairs to use for the simulations, certain criteria had to uphold:

- CEX data must be available and updated in intervals of maximum 1 minute
- The pair must be available and traded on Uniswap-V2 over the full period
- One token of the pair must be the same across all pairs, the so called numeraire token
- Pairs must have some form of relevance eg. high trade volume, established coins, high liquidity

[1] A Ethereum Blockchain Explorer `https://etherscan.io/`
[2] Link from where data was pulled https://www.cryptodatadownload.com/data/bitstamp/

Pool	Starting Liquidity	Starting Price
WETH-USDC	210 713 262 USD	1 to 2933.68
DAI-USDC	69 988 631 USD	1 to 1.0001
WBTC-USDC	571 518 USD	1 to 44 663

Table 6.1.: Pools initial values liquidity pulled from Uniswap-V2.

These criteria resulted in choosing the following three pairs: USDC-WETH, USDC-DAI, and USDC-WBTC.[3]. WETH and WBTC are wrapped tokens of ETH and BTC coins as both of these tokens are not originally created as ERC-20 tokens.
Because USDC-WETH was the most traded pair at the time, it was included in the preliminary pool. As a result, the simulations' common token is USDC, which is a center stable-coin built on a smart contract that works to keep the USDC value 1 to 1 with the US Dollar[4]. This allowed to a straightforward conversion to USD as it has been assumed that it would remain constant throughout the entire duration of the simulations. Hence using Bitstamp data, which is priced in USD, is reasonable and does not lead to inconclusive results. USDC-DAI has been picked because it is the second most traded pair on Uniswap-V2 during the time period considered while meeting the requirements. The USDC-WBTC combination was chosen both because it has substantially less liquidity than the other two (Table 6.1), and because the BTC token is well-known and traded on CEXs.
For a meaningful study, selecting proper sets of initial values is critical. Table 6.1 shows the starting liquidity and price utilized in all simulations for each pool pair.

6.2 Simulations

Several scenarios were considered for the following simulations in three different permutations of the above pairings have been analysed. Alternative compositions of liquidity providers will be examined for each of them. The purpose is to comprehend how returns differ depending on the pools studied and the initial holdings of LPs. Three separate LPs with baseline portfolios

[3]Links to each of the pools: WETH-USDC, DAI-USDC, WBTC-USDC
[4]This can be seen from coinmarketcap.com

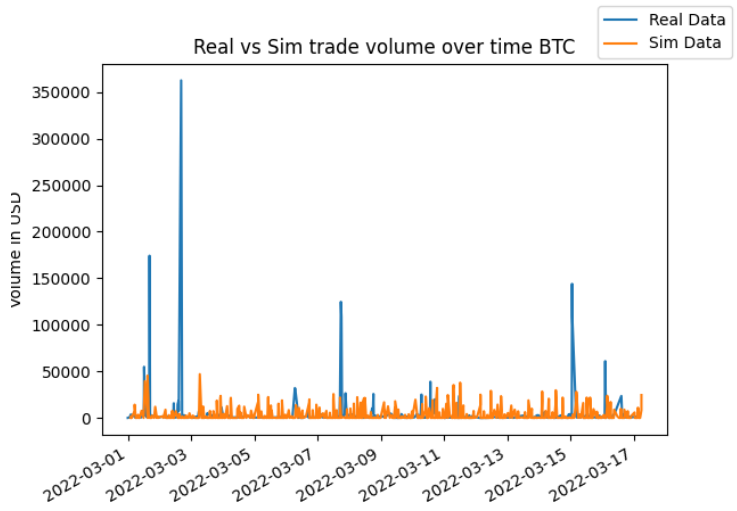

Figure 6.1.: Accumulated trade volume in USD in periods of 10 minutes for WBTC.

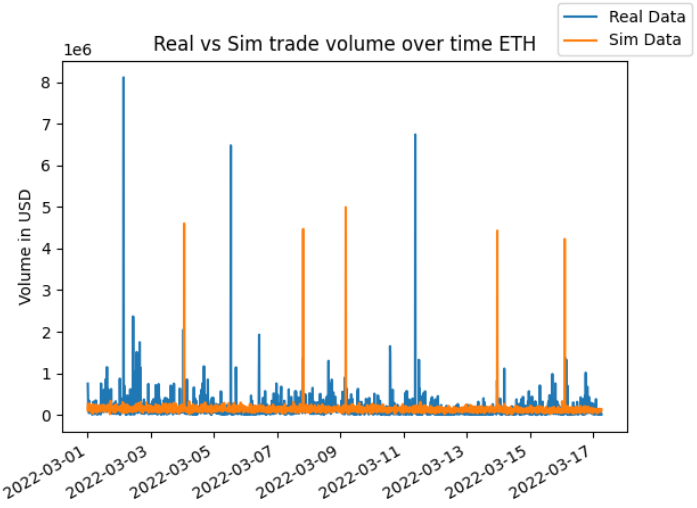

Figure 6.2.: Accumulated trade volume in USD in periods of 10 minutes for WETH.

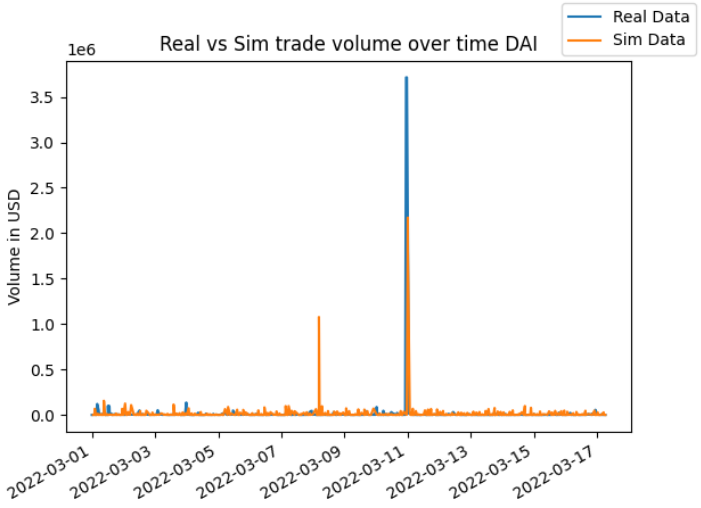

Figure 6.3.: Accumulated trade volume in USD in periods of 10 minutes for DAI.

have been identified: Because the simulations' focus has been on LPs, it has

Type	USDC	ETH	BTC	DAI
Small	100 000	2.5	10	100 000
Medium	1 500 000	400	20	1 500 000
Big	10 000 000	5000	400	10 000 000

Table 6.2.: Different liquidity providers baseline by portfolio's size.

also been vital to mimic the trading witnessed on the real exchange. This has been done, as previously indicated, using signal pools, but also by tweaking transactions to follow market trades. This was accomplished through the use of several instances of trade agents dealing with different set of pools. Each simulation includes the following Trade Agents:

- 2 traders that can trade with one of the two pools random small amounts for every step
- 2 agents trading only with the USDC-WETH pool at higher volumes, but only once every 3 timesteps
- 1 trader doing high volume trades once every 2.5 hours on the USDC-WBTC or USDC-DAI pool (based on the experiment settings)
- 1 trader doing twice the volume of the trader above, but only once every 10 days in the USDC-DAI pool.

As a result, trade patterns in the simulation environment are as depicted in Figures [6.1] - [6.3]. Although these accomplish to match total trade volumes within a +- 10%, with the margin varying across random runs, there are some significant differences. The key difference is that all three currencies are traded more often but in smaller amounts. This is notably noticeable in WETH trading, where the average deal size per minute is greater in the simulator than in the real data. However, the genuine data has far greater spikes at times. Since LPs gains are not influence by sizes of individual trades, but only by the total volume traded between provisions or withdrawals of liquidity, the setting can be considered to be coherent and adequate to pursue the analysis.

6.2.1 USDC-WETH vs USDC-WBTC

The following are four different scenarios that have been researched and analysed, all of which include WETH and WBTC. Various combinations of LPs were considered in each case in order to examine their strategies, gains, and how their actions affect each other's returns.

2 Small LPs The cumulative returns at the end of the experiment are 0.09% if only two small LPs are considered to provide to the pools. Figure [6.5] shows how an initial time period in which no provision is made results in no profit or loss in the value of the LPs' portfolios. This lack of action is due to low cumulative trading volumes: liquidity providers are only required to provide if the mean of trading volumes over a specified time period exceeds a certain threshold. The random trades are too small in this scenario for LPs to consider providing as a profitable strategy.
With the first time LPs provide to one of the pools, there is a drop in gains. This is because they need to pay gas fees in order to carry out that action. Agents with smaller holdings are more affected by this cost of service, as their fee collection returns may be on par with gas costs. The drop seen at 2022-03-08 is due to an arbitrage action that changed token reserves and thus the value of LP's portfolio. Nonetheless, arbitrage fees are collected and redistributed back to liquidity providers whenever the pool's liquidity changes. This change in LPs profits is evident when comparing cumulative returns jumps to provision actions in Figure [6.5].

2 Medium LPs and 2 Small LPs When two liquidity providers with a larger portfolio are added to the previous case, small LP gains increase by 0.14% at the end of the experiment. This is unsurprising, given that a pool with more liquidity is less sensitive to the effect of slippage for trades of similar size. This increases trades cumulative volumes and as a consequence small LPs are incentivized to provide. The increase in provision can be seen in Figure [6.7], when compared to Figure [6.5] as small LPs own an higher percentage of the pool.
Medium providers, on the other hand, have a higher return than small LPs, with returns peaking at nearly 0.30%. Larger LPs should expect higher returns because they are entitled to a larger portion of the fees (Figure [6.7]) and the impact of gas fees is minimal for these LPs. The more stable positive trends

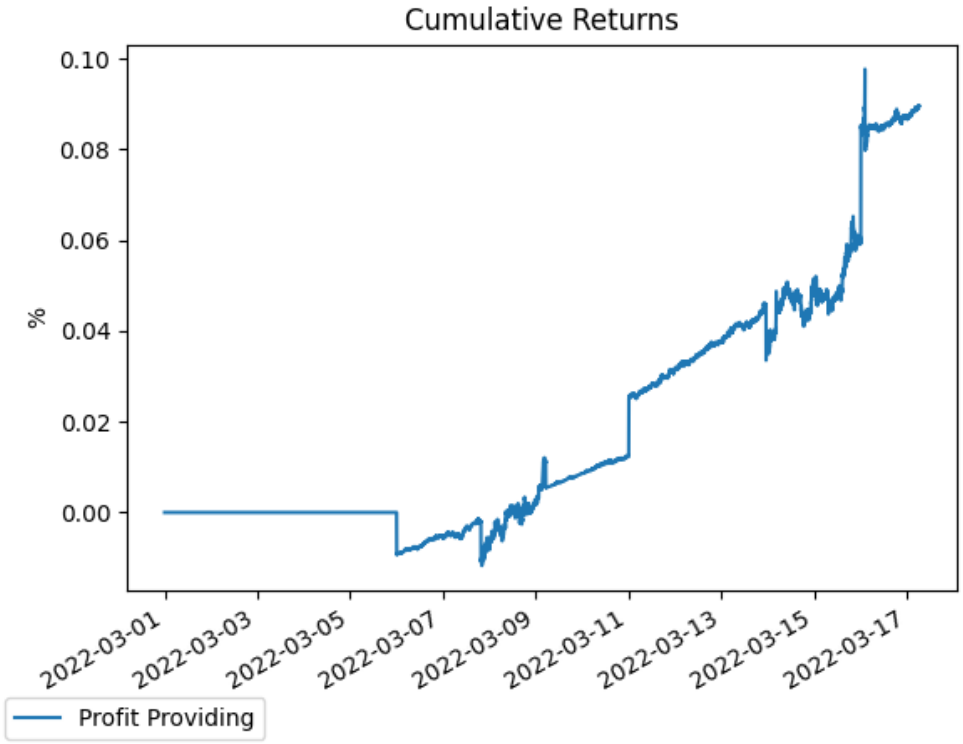

Figure 6.4.: Cumulative returns of a Small LP if provision is made.

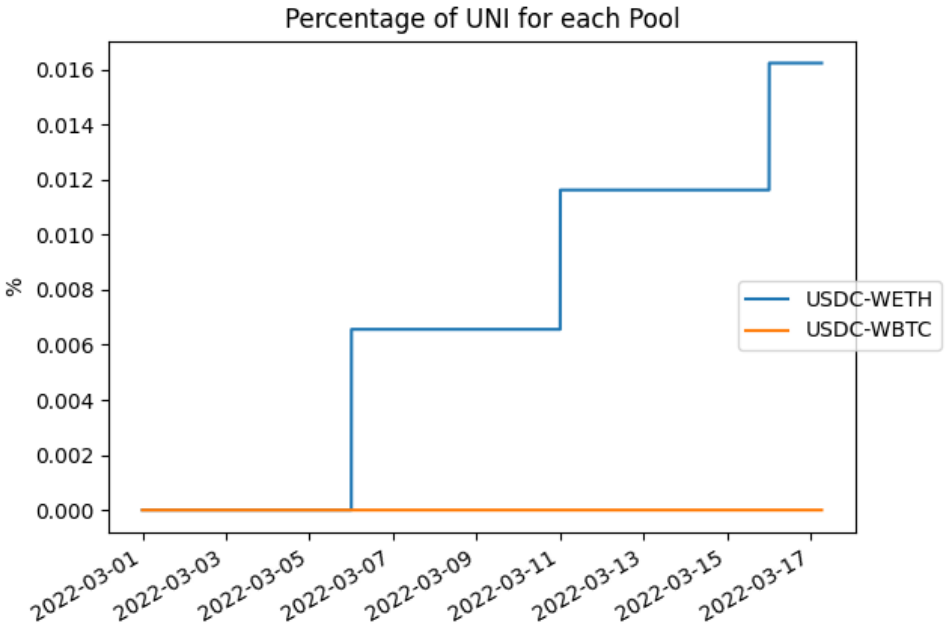

Figure 6.5.: Relative amount of UNI hold by a Small LP. They are used as a bookkeeping tool in order to determine the amount provided to pools for each LP.

in the returns are indicative of this phenomenon. Furthermore, Figure [6.6] shows how price changes have a smaller impact on gains for medium LPs. Gains for both Small and Medium LPs are more stable than in the previous case.

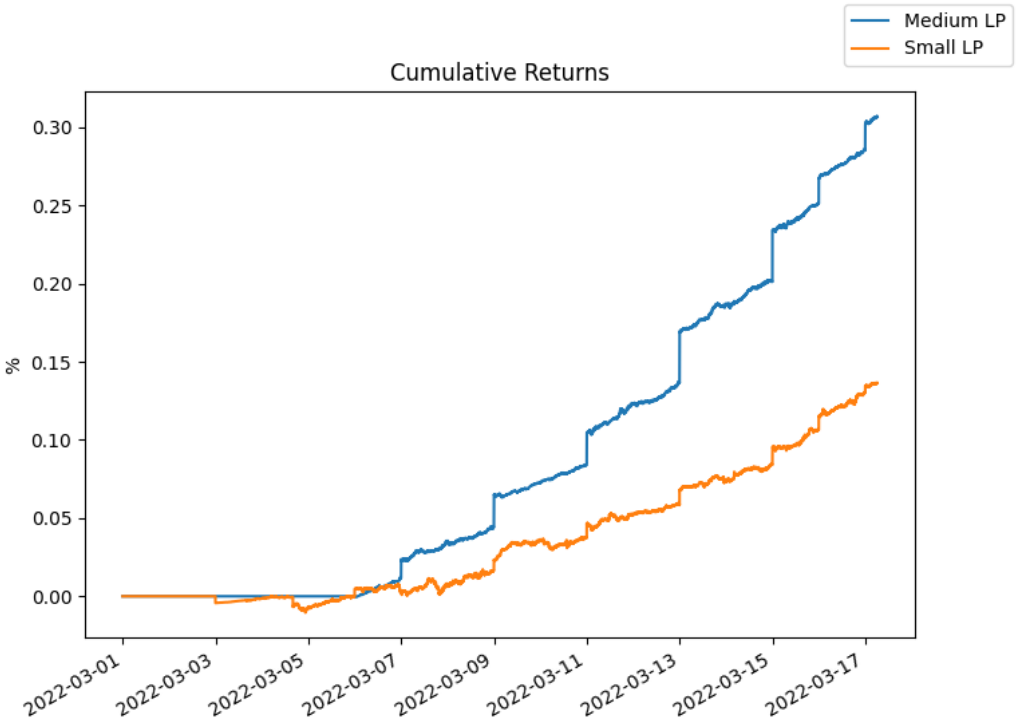

Figure 6.6.: Comparison between different LPs returns.

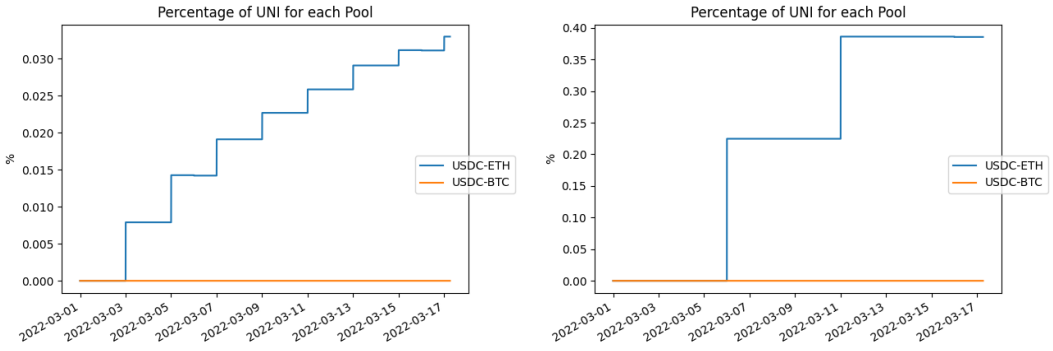

Figure 6.7.: Percentage of UNI in each pool for Small and Medium LPs.

1 Big and 2 Small LPs Returns for small liquidity providers peak at 0.20 % when a Big LP provides to pools, according to simulations. Unlike medium LPs, big providers appear to be more exposed to price fluctuations, as fees collected are insufficient to cover the difference due to impermanent loss over a short period of time. In fact, in the simulation, this type of LP has a less stable rate of return (for example, between 2022-03-15 and 2022-03-17), more similar to that of small LPs than to that of medium LPs. These jumps, as shown in Figure [6.9], do not correspond to actions taken by small LPs, implying that they are solely due to price variations. An interesting behaviour in LPs profits can be noticed around 2022-03-09. While the Big LP faces a drop in its profits, it would appear that, surprisingly, the same does not happen to Small LPs. But this is not the case. A closer look highlights that also Small LPs are subject to a loss in their gains due to a price variation, but this loss is mitigated by the redistribution of fees happening in conjunction with the Big LP providing to the pool. Moreover, it can be seen that big LPs interact later than small LPs. This is due to the fact in these simulations they are considered to have lower frequency for provision.

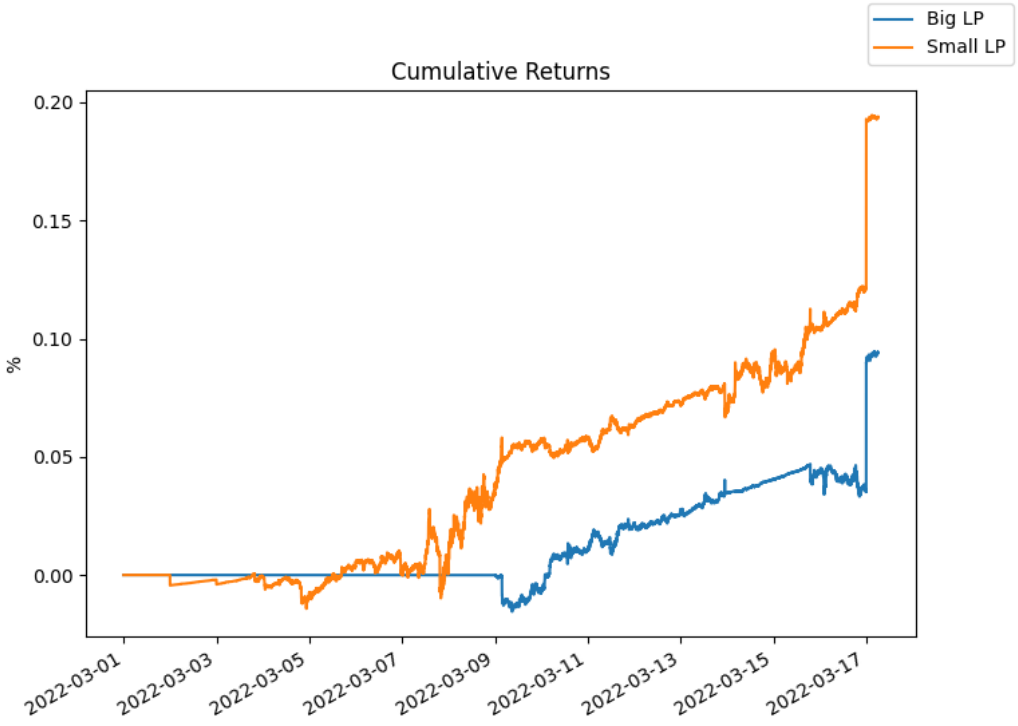

Figure 6.8.: Comparison of returns for Big and Small LPs.

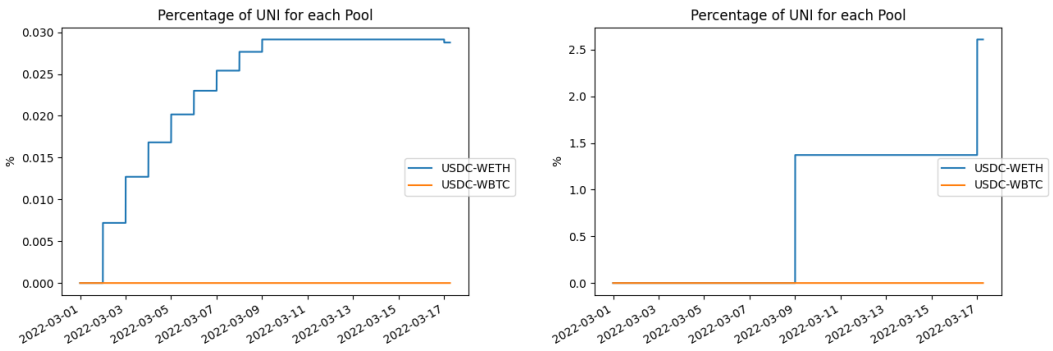

Figure 6.9.: Portfolio history of Small and Big LPs with WETH and WBTC tradeable.

Pool	Experiment 1	Experiment 2	Experiment 3	Experiment 4
WETH-USDC	335 424 104	318 719 973	333 409 951	346 629 039
WBTC-USDC	2 334 554	2 153 941	2 349 424	2 564 385

Table 6.3.: Trading volumes in USD in pools at the end of simulation.

2 Big and 2 Small LPs As one might expect, adding one more Big LP has a significant impact. The returns on small LPs have dropped to 0.15% from 0.20%, but the gains on Big LPs have dropped to 0.02%. As expected, strategies are similar among LPs in the same category, and they are identical to the previous case scenario, Figure [6.10].
Having LPs with large portfolios provide at the same time appears to have a negative impact on all LPs to varying degrees.

In conclusion, regardless of the LP combinations considered, it appears that providing is the most profitable strategy in these circumstances. It's worth noting that trade volumes vary little from one experiment to the next (Table 6.3). Having agents provide liquidity to pools will inevitably change the market because traders will experience less slippage. Nonetheless, ensuring that trade volumes do not fluctuate dramatically ensures that the above results are not due to the simulations' randomness. No liquidity provider has invested in the USDC-WBTC pair, as shown in the examples above. This is due to the small volume of trades that take place in the pool. As a result, LPs are unlikely to provide to a pool where fees are unlikely to cover the volatility of the USDC-WBTC pair. It is thus more profitable to avoid providing WBTC than to lock any amount of it into the protocol.
Comparing these findings to [34] it would appear that the experiments run

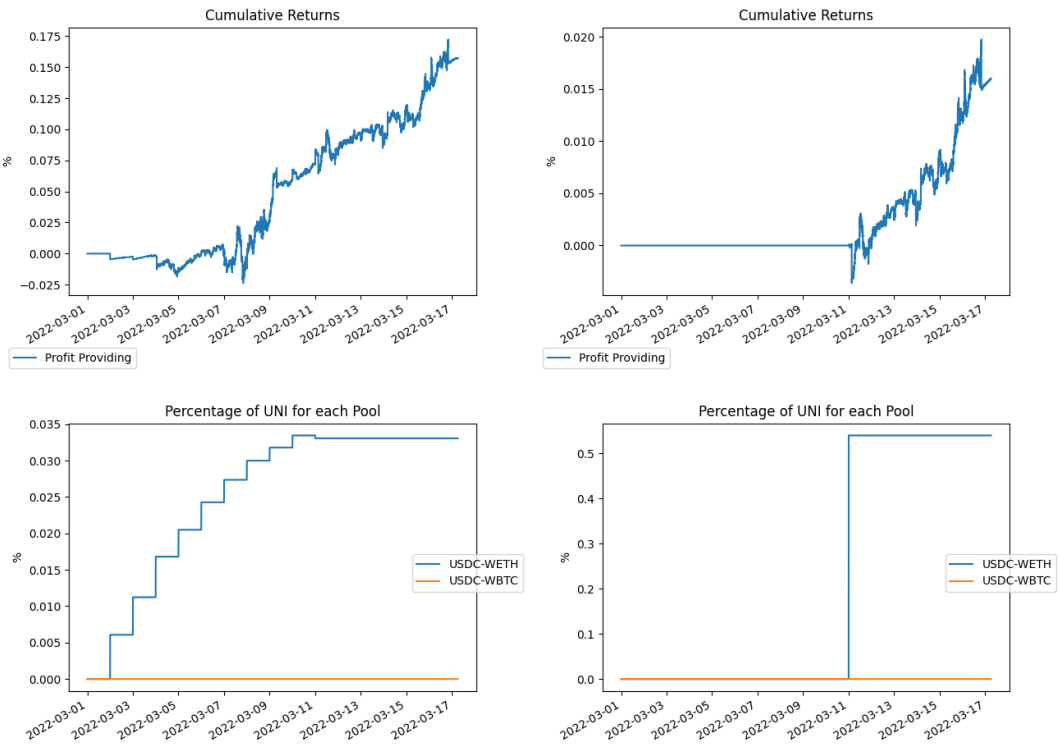

Figure 6.10.: Portfolio history of Small and Big LPs with WETH and WBTC tradeable.

in this project present significant differences compared to real world scenario. In that analysis monthly returns up to 5% are recorded and annual ones up to 20%. The difference in this project's results is to be appointed due to the fact that the whole portfolio's performance is considered. Hence net profits consider impermanent losses both for the USDC-WETH and the USDC-WBTC pairs, lowering the returns registered.

6.2.2 USDC-WETH vs USDC-DAI

Simulations with similar settings as before, but with pool trading USDC-WETH and USDC-DAI, were conducted in order to study how returns are correlated to tokens traded. When using the same LPs with this new pool pairs, some differences have emerged.

2 Small LPs When only small LPs are taken into account, their strategy is very similar to the previous cases, but their returns are only 0.05%. This is because, as shown in Figure [6.11] LPs provide to the USDC-DAI pool in this scenario, avoiding the fee distribution that occurs in the previous setting.

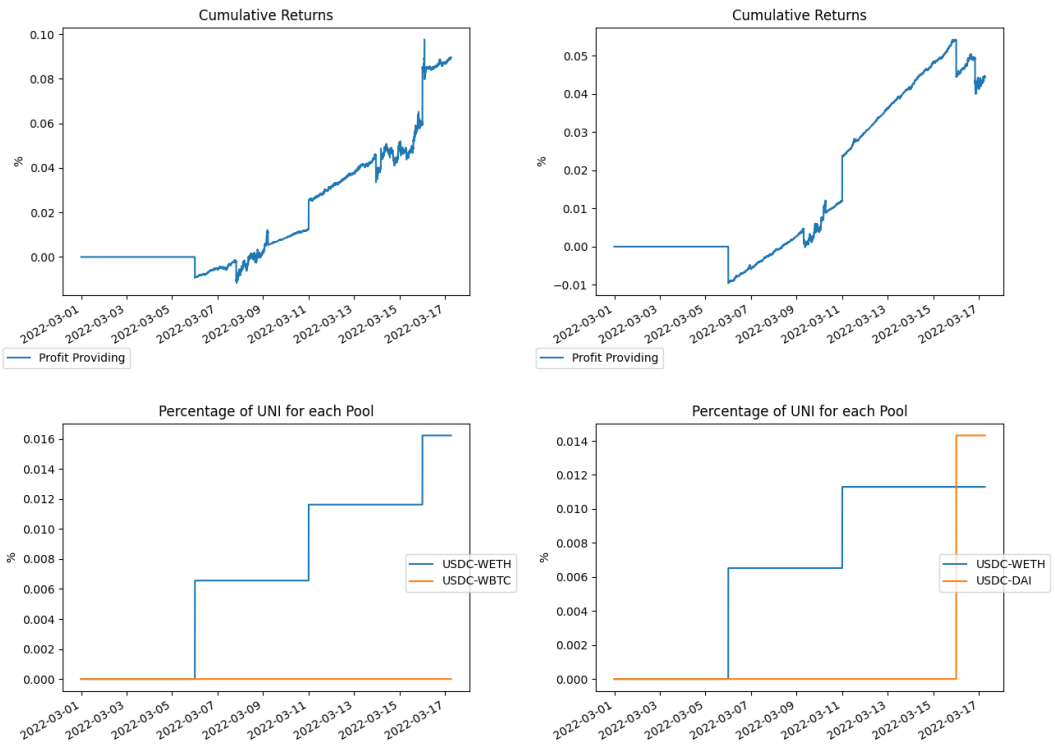

Figure 6.11.: Differences in returns and providing strategies of Experiment 1 when WBTC (left) or DAI (right) are considered.

2 Medium and 2 Small LPs Having two medium and two small LPs increases profits for agents with smaller portfolios once again, as they achieve 0.30 percent returns. Figure [6.12] shows that LPs are once again providing to both pools, implying that volumes are more favourable to overcome impermanent loss.

1 Big and 2 Small LPs Another difference can be noted in the case of one big and two small LPs: agents with smaller portfolios make a peak profit of 0.30% (compared to 0.20% with WBTC tradeable), while the larger LP's return drops to 0.05%. As shown in Figure [6.13], these results are due to the fact that USDC-WETH fees are not redistributed at the end of the experiment.

2 Big and 2 Small LPs The final scenario is similar to its WBTC counterpart in that LPs with larger funds see their profit shrink to 0.03%. Returns on small LPs, on the other hand, are increasing to nearly 0.40%. Figure [6.14] shows that profits have increased as a result of a higher percentage of UNI tokens in the USDC-WETH pool, resulting in Small LPs receiving a larger portion of the fees collected.

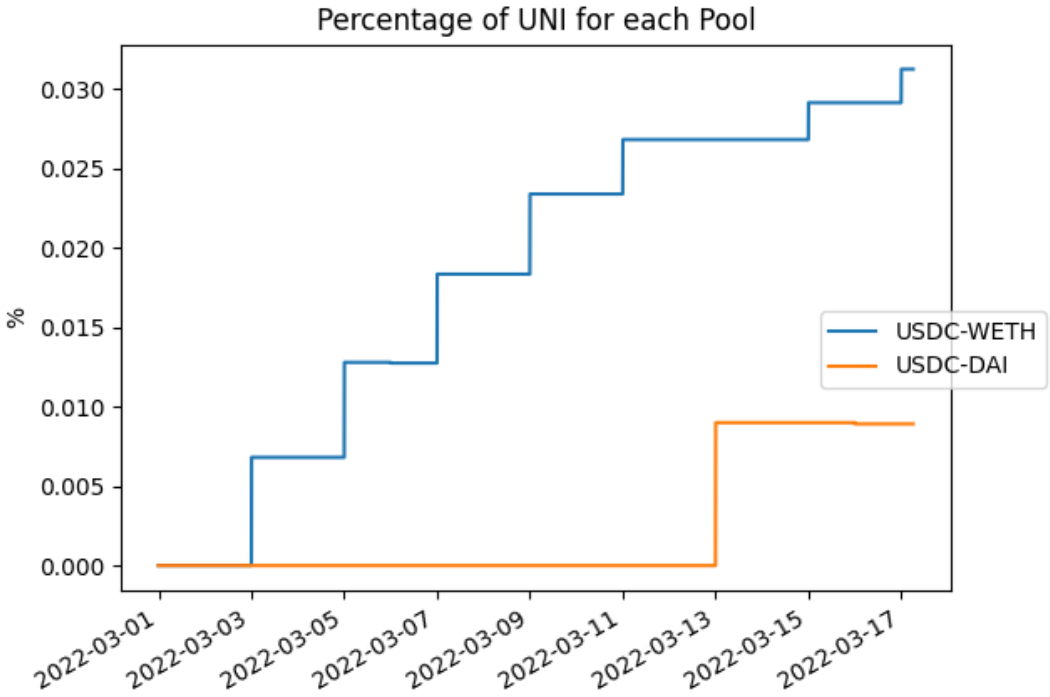

Figure 6.12.: In this setting LPs are providing to both pools.

All of the above findings are not surprising. Liquidity providers face little to no impermanent loss due to the stable nature of DAI, as prices do not fluctuate as much as the other two pairs. This allows LPs to collect fees for trading that takes place in the USDC-DAI pool while only incurring service costs.

Pool	Experiment 1	Experiment 2	Experiment 3	Experiment 4
WETH-USDC	328 816 987	334 989 671	328 027 231	337 706 364
DAI-USDC	12 069 090	8 206 081	10 678 035	7 867 679

Table 6.4.: Pools volumes in USDC.

LPs consider providing in both pools to be a viable strategy, but refrain in the previous case scenario due to trading volumes being too low in the USDC-WBTC pool, as can be seen from Table 6.4.

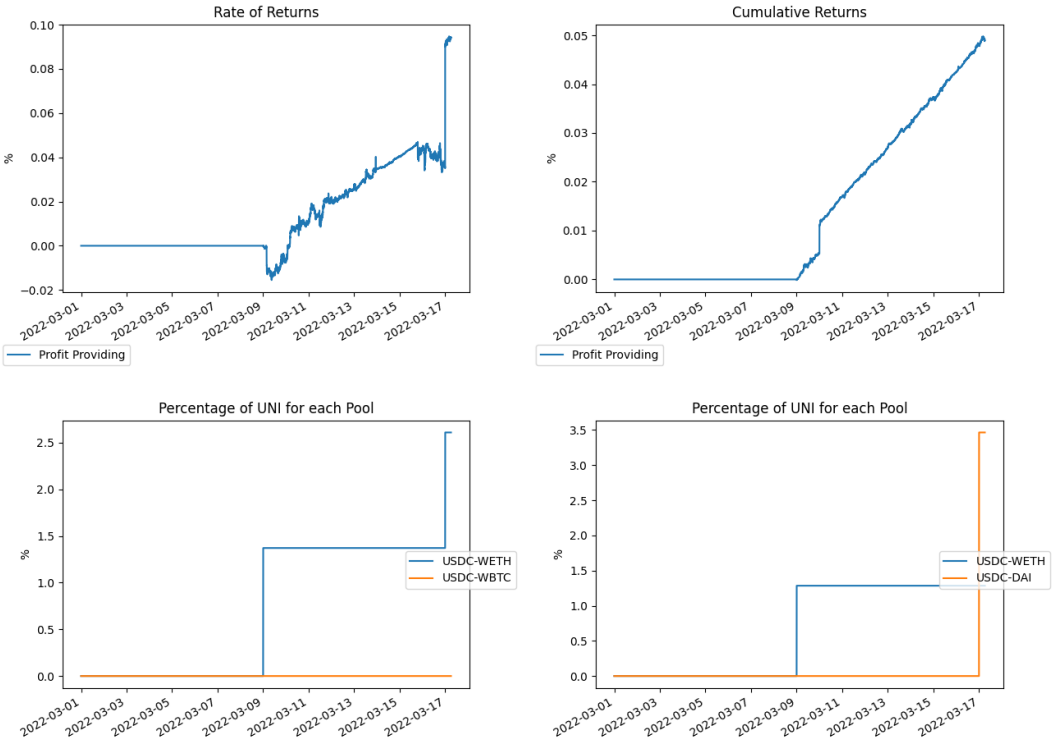

Figure 6.13.: Differences in returns and providing strategies of Experiment 3 when WBTC (left) or DAI (right) are considered.

6.2.3 USDC-WETH vs USDC-WBTC vs USDC-DAI

Another experiment was carried out to see how the presence of all three pools would affect LPs returns. Two Big and two Small LPs have been considered to interact with the different pools in this case scenario. To keep the results as consistent as possible, all parameters were left unchanged. Different strategies are used by liquidity providers: it appears that LPs consider investing in only one pool to be more profitable, and only one of the small LPs provides liquidity to two pools. Providing to the USDC-WBTC pool is not a profitable strategy for any agent, once again. Returns differ from one agent to the next due to their LPs' different approaches, as shown in Figure [6.15]. Providing to the USDC-DAI pool results in a return increase of only 0.002%, which is a low rate when compared to providing to the more volatile USDC-WETH pool, which yields a tenfold higher return.

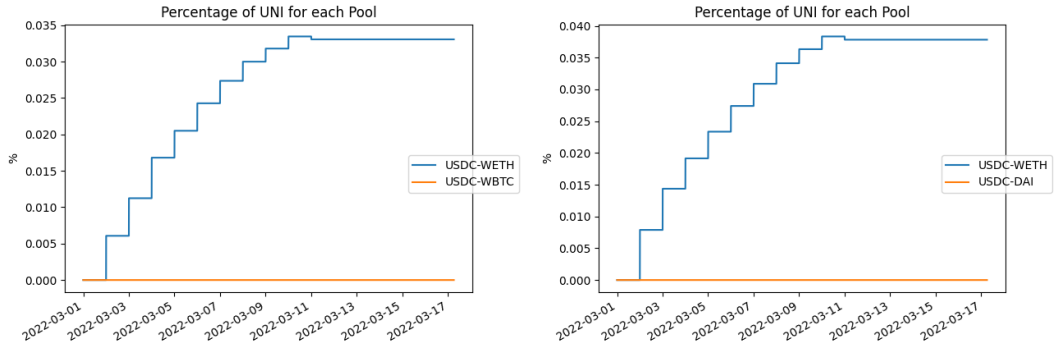

Figure 6.14.: Differences in UNI tokens for Experiment 4 when WBTC (left) or DAI (right) are considered.

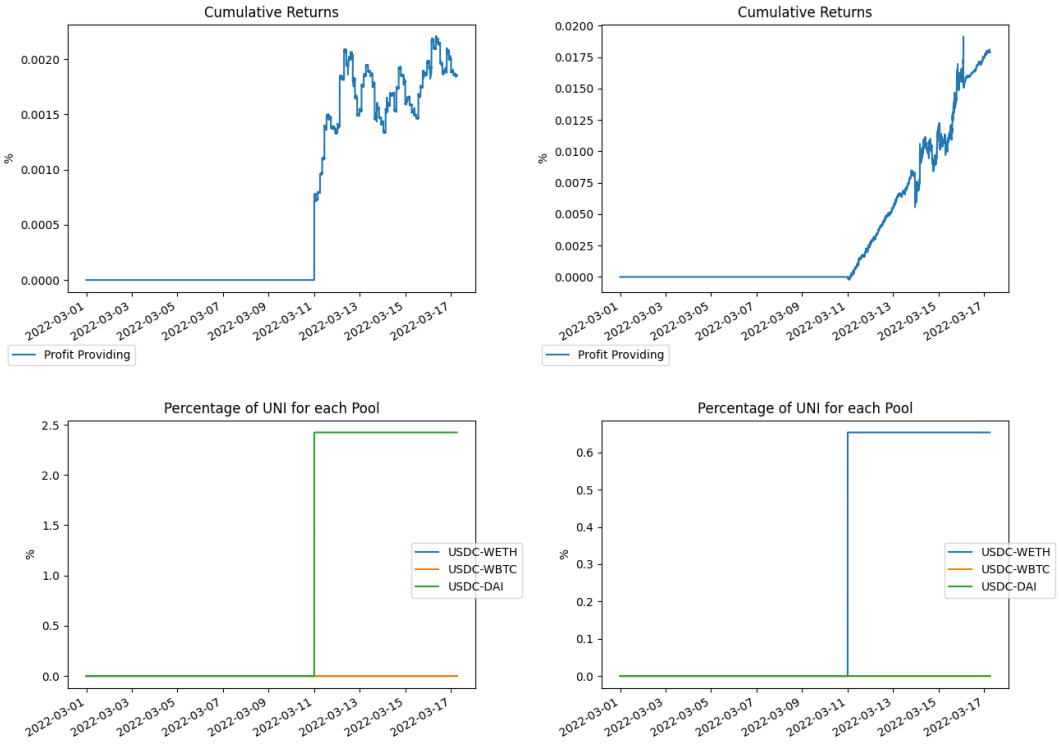

Figure 6.15.: Differences of LPs strategies and returns when considering all three tokens to be tradeable.

6.2.4 Different fees

Because fees are the only way for LPs to profit from interacting with the protocol, it is reasonable to see if any tweaks could result in an increase in profits, especially since different fees imply different trading volumes.
To achieve this, previous simulations were run with trading fees of 0.5% and 0.1%, respectively. Figure [6.16] shows the results of these new experiments. When it comes to USDC-WETH and USDC-WBTC, Small LPs do not appear to benefit from either an increase or a decrease of the trading fee. A slight drop in trading volume translates to a significant loss for this group of LPs, which is already impacted by gas prices. Simultaneously, the increased volume obtained by lowering the fee is insufficient to result in higher profits. This is not the case in experiments in which DAI is used instead of WBTC. If fees are increased to 0.05%, Small LPs will see an increase of 0.02% (nearly 1.5 times the returns).
When LPs with larger portfolios interact with the pool, a different pattern emerges. Higher fees encourage LPs to provide, resulting in lower levels of slippage for traders. From the perspective of traders, it appears that the fee increase compensates for the slippage. As a result, trading volumes are comparable, and LPs' total returns increase.

As shown in Figure [6.16], LPs with larger portfolios appear to benefit the most from a higher fee. Given that trade volumes are within the same range for all three fees, the reasons for the disparities in cumulative rates of return should be investigated in any differences in LPs providing strategies. Because each LP has a different minimum return threshold, having higher fees helps forecast a higher return and thus incentivizes providing. This is best illustrated in Figure [6.17], where it can be seen that when two Medium and two Small LPs are considered, returns for a fee of 0.3% suddenly outperform others. This is due to the fact that fees from the USDC-WETH have yet to be collected in the other strategies, demonstrating how strategies change and adapt for different scenarios.

Changing trading fees has a number of consequences, not only for trading volumes but also for providing trading strategies. Overall, it appears that when only small providers interact with pools, trading volumes decrease, resulting in a lower cumulative return. Gains on larger LPs, on the other hand, are the polar opposite. They benefit from higher transaction fees because they are less exposed to gas costs and their actions have a greater impact on attracting trade agents to operate on protocol pools.

Figure 6.16.: Differences in returns for several fees. Top: Small LPs in the first two experiments. Small LP (left) and Big LP (right)in experiment 3, 4 and 8.

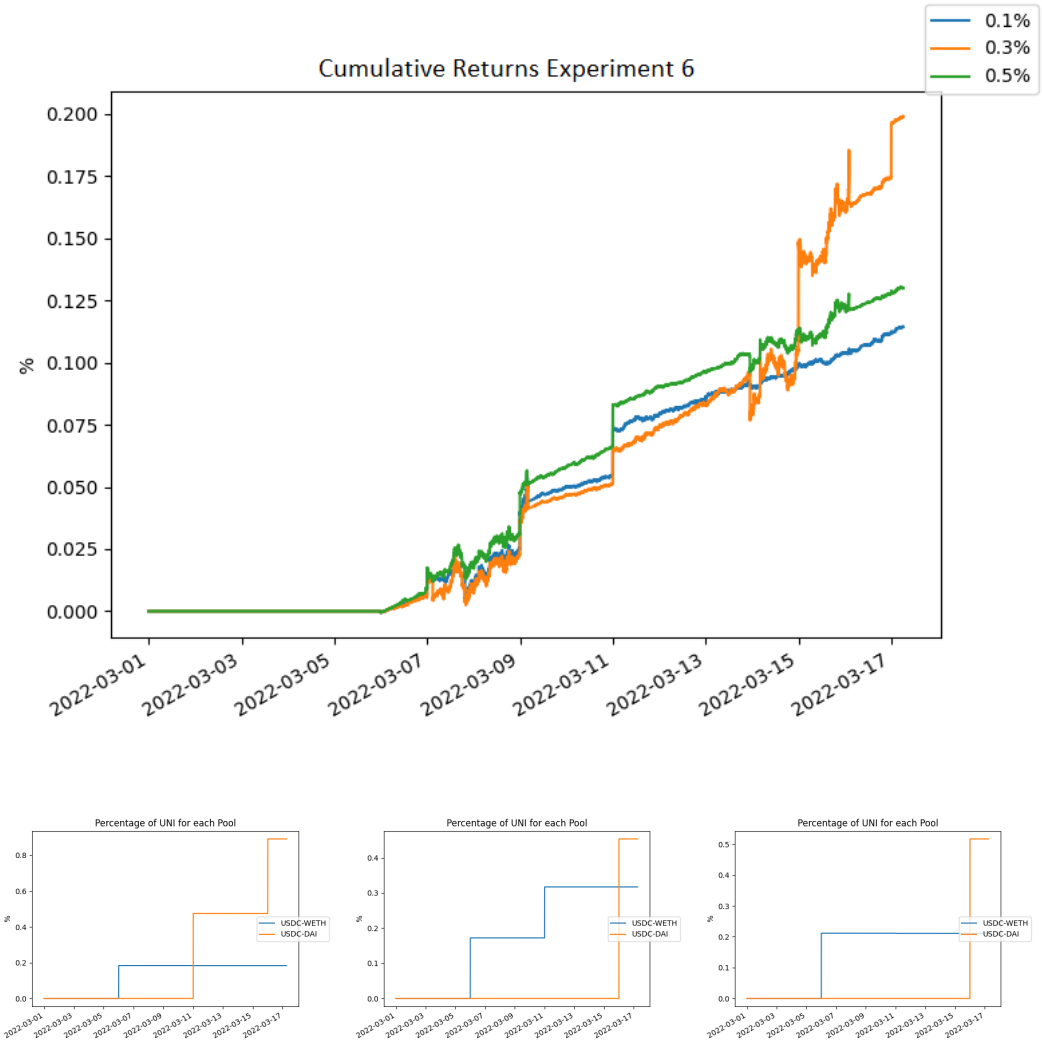

Figure 6.17.: Different strategies for each fee cause different returns. From left to right: 0.1%, 0.3% and 0.5%.

7 Discussion

Looking at how prices fluctuate for the three different tokens considered in simulations (Figure [7.1]), it is expected for liquidity providers to make profits. As discussed extensively in previous sections LPs are exposed to impermanent loss whenever they provide. Meaning that their portfolio depreciate as prices fluctuate. Although, if after a period of volatility prices settle back to the initial levels the impermanent loss faced by LPs is zero. Hence providing results in no depreciation or lost extracted value, whilst accumulating value through collection of transaction fees leading to positive returns. Results also show that liquidity provider are encouraged to be dynamic by improving pools' liquidity depth as it ensures higher percentage from fee collection.

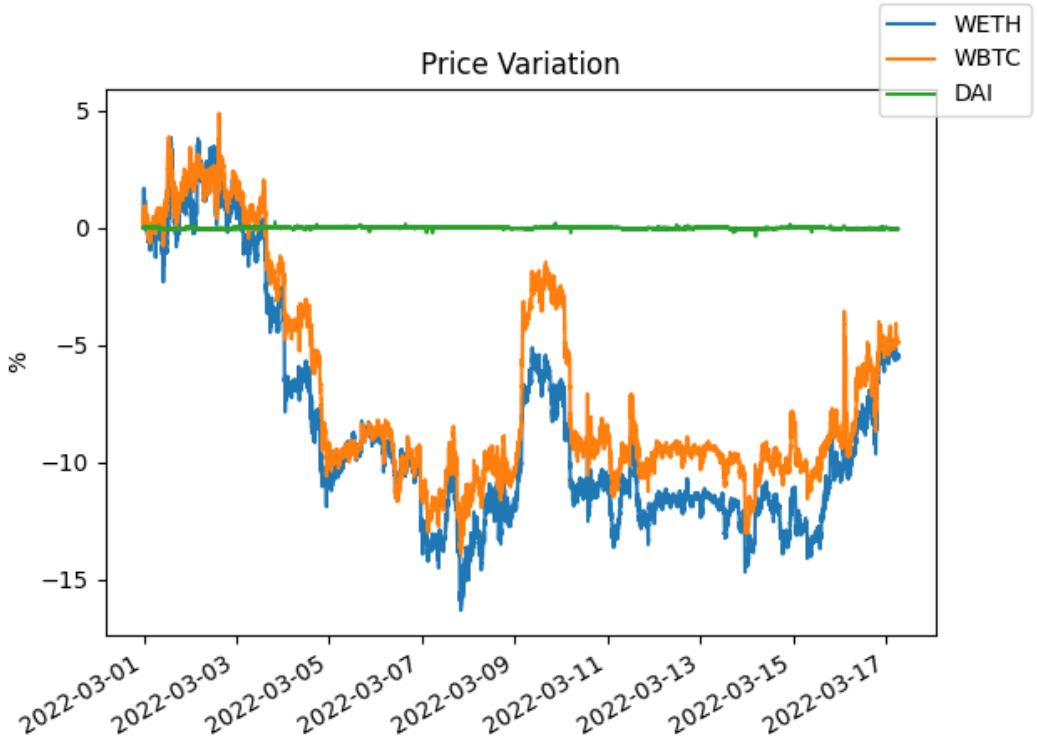

Figure 7.1.: Price fluctuations of WETH, WBTC and DAI relative to their value as per 2022-03-01

Considering a 17 days time window for the analysis might lead to considering results not completely conclusive, as other analysis indicate higher returns for the same pairs [34]. Nevertheless the analysis conducted in this project differs slightly from such, as gains are considered in regards of LPs portfolio's sizes, token composition, how they interact with each other, what coins are provided and what are the effects of gas fees. Hence discrepancies are to be expected.
Although Ethereum's block times is approximately 14 seconds, it should be noted that several transactions are received to be added to each block, are not uniquely Uniswap's trades. Then considering an interval of 1 minute between each block in the simulations appears a feasible assumption that does not cause for unrealistic results in the scope of this project. Moreover, this is backed by the trading volumes reached in simulations that are within 5% for the USDC-WETH, the pool with the highest trading volume and liquidity among the three considered.
It is important to note that the three token pairs chosen for this analysis represent only a small portion of the total number of token pairs available on Uniswap V2. WETH, USDC, and DAI, on the other hand, are the tokens with the highest trading volumes and liquidity within the protocol[1]. Restricting the analysis to these coins and WBTC (chosen to include coins that are primarily related to the blockchain environment) was not deemed restrictive or limiting.
An important factor of trading on the Ethereum blockchain is the associated gas cost that comes with doing these trades. This cost will be higher depending on both the amount of other transactions competing for acceptance [3.2]. These might be caused by all sort of events, one of them being the potential profits that can be made by doing arbitrage. Such trades will often be discovered by more than one user driving up the price they are willing to pay in order to secure the arbitrage. In the simulations conducted, gas costs have been considered constant as to avoid noise to the data that would have made complex to identify how the several factors and variations in profits are correlated. Moreover fluctuating gas prices would have also had an impact on LPs providing strategies, in particular for Small ones as their returns are more dependant on gas costs then larger LPs.
Figure [7.2] shows that the presence of arbitrageurs allows spot prices within the pool to be adjusted to reflect CEX prices. This phenomenon is critical for the experiments to be consistent and reflect real-world scenarios, because otherwise, LPs would not experience impermanent loss, resulting in consistent positive returns regardless of fluctuations in other exchanges. However,

[1]Uniswap V2 token overview

spot prices in Uniswap do not perfectly match outside prices, implying that traders use different strategies than the actors in the simulations. This difference was also evident in the early stages of the analysis, when traders were assumed to be equally likely to interact with the various pools: volumes for the USDC-WBTC protocol were 400 times higher than those for the real Uniswap protocol, despite the fact that price fluctuations for WETH and WBTC are similar. Because of these inconsistencies, it was necessary to create a different type of uninformed trader. The stability of prices within the pool for some experiments (e.g. Experiment 2) is the reason for the more stable gains' trends as LPs face a stable impermanent loss while outside prices are fluctuating, as seen in Figure [7.2]. In this project only one external source of price fluctuation has been implemented. Since CEXs' prices are determined via the more traditional ordering book model it would be very unlikely to have major differences amongst these exchanges. An interesting question could arise at this point: is Uniswap, and DEXs in general trailing and following price changes that are registered in centralized exchanges or is it the other way around? Experiments in this project show that prices in the protocol adapt following information from CEX. To answer to this question a more empirical analysis is needed and has not been carried out in this project.

Figure 7.2.: Price fluctuations of WETH, WBTC and DAI within Uniswap relative to their value as per 2022-03-01

8 Conclusions

All the experiments suggest that providing liquidity to multiple pools is a profitable strategy with an yearly return of 1% for the whole portfolio in the majority of scenarios. In a highly volatile market, a return of 1% might not be too appealing to liquidity providers as they might consider this strategy to too slim gains considered the amount of risk they would expose themselves to. Nonetheless, return sizes vary and are influenced by a number of factors. Large provisions are more likely to incentivize trading as they contribute to reducing slippage. Although the presence of multiple large providers reduce their individual profit as a effect of having a smaller a portion of fees collected. It is then clear that the best strategy for an LP would be to provide to a pool where trades are such that a provision would lower the level of slippage faced by traders. In fact if a pool has a deep liquidity increasing trading fees would not result in a sudden and noticeable diminish in trading volumes, meaning that LPs would have higher returns. One the other hand reducing trading fees would most likely not lead to an increase in trades volumes that would results in an overall increase in returns.

Experiments with two Big LPs, in which their returns are lower than when only one LP with similar portfolios is considered, raise the question of how LP strategies should adapt to the presence of other LPs and a pairs status. The protocol's ultimate goal is to create pools with deep liquidity to ensure trades. Simulations on LPs providing strategy in settings where the spot price and trading volumes of the USDC-WBTC pair were left unchanged, but with increased liquidity have been conducted. These experiments have not been presented as no difference for LPs gains has been recorded, due to them still not providing to that pool. Low trading volumes would result in low fee gains, which would be further reduced by the fact that fees would be proportionally

distributed among LPs.

Furthermore, introducing controlled fluctuations in gas costs in order to analyse their impact on liquidity providers' profits would be an interesting extension and continuation of this research. Approximating gas prices based on historical data could be one way to recreate such a phenomenon in a controlled manner. This could have a greater impact on Small LP gains, but it could also lead to a more cautious providing strategy, as the cost of each action would have a different impact on profits. Gas price fluctuations obviously have an impact on all agents interacting with the protocol, as both liquidity providers and trading agents must consider an additional deterring variable when deciding on an investment strategy and determining whether or not executing transactions on the protocol is profitable. This would result in more dynamic volumes and gains for LPs, but it would also introduce noise into the data, as previously stated. As a result, it is the ideal next step to bring analysis of liquidity providers' profits to higher level of precision.

Bibliography

[1] Bank for International Settlements. „OTC derivatives statistics at end-June 2021“. In: (2021). URL: https://www.bis.org/publ/otc_hy2111.pdf.

[2] International Monetary Fund. *GDP, current prices*. URL: https://www.imf.org/en.

[3] Statista. *Overall cryptocurrency market capitalization per week from July 2010 to April 2022*. URL: https://www.statista.com/statistics/730876/cryptocurrency-maket-value/.

[4] Bank for International Settlements. „Foreign exchange turnover in April 2019“. In: (2019). URL: https://www.bis.org/statistics/rpfx19_fx.pdf.

[5] Satoshi Nakamoto. „Bitcoin: A Peer-to-Peer Electronic Cash System“. In: (2008). URL: https://bitcoin.org/bitcoin.pdf.

[6] www.currentmarketvaluation.com. *Fed Balance Sheet vs S&P500*. URL: https://www.currentmarketvaluation.com/posts/2021/07/Fed-Balance-Sheet-vs-SP500.php.

[7] Elton Edwin J., Gruber Martin J., Brown Stephen J., and Goetzmann William N. *Modern Portfolio Theory and Investment Analysis*. Wiley, 2014.

[8] U.S. Securities and Exchange Commission. *Framework for “Investment Contract” Analysis of Digital Assets*. 2019. URL: https://www.sec.gov/corpfin/framework-investment-contract-analysis-digital-assets.

[9] Markowitz Harry. *PORTFOLIO SELECTION*. 1952. URL: https://doi.org/10.1111/j.1540-6261.1952.tb01525.x.

[10] Pascucci Andrea and Runggaldier Wolfgang J. *Financial Mathematics*. Springer, 2009.

[11] Robert C. Radcliffe. *Investment: Concepts, Analysis, Strategy*. Addison-Wesley Longman, Incorporated, 1982.

[12] Maynard Keynes John. *The Classification of Money*. Cambridge University Press, 1930.

[13] Qin Kaihua, Zhou Liyi, Afonin Yaroslav, Lazzaretti Ludovico, and Arthur Gervais. „CeFi vs. DeFi — Comparing Centralized to Decentralized Finance“. In: (2021). URL: https://arxiv.org/pdf/2106.08157.pdf.

[14] Ethereum development team. *Ethereum Whitepaper*. 2022. URL: https://ethereum.org/en/whitepaper/#ethereum.

[15] FxPro team. *Fx Pro Trade Like a Pro*. 2022. URL: https://www.fxpro.com/.

[16] ethereumprice. *Gas Price by Time of Day*. 2022. URL: https://ethereumprice.org/gas/.

[17] Radar Relay. *WTF IS WETH?* 2022. URL: https://weth.io/.

[18] Ethereum development team. *PROOF-OF-STAKE (POS)*. 2022. URL: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/#top.

[19] Ethereum dev team. *The Merge*. 2022. URL: https://ethereum.org/en/upgrades/merge/.

[20] Jensen Johannes Rude, von Wachter Victor, and Ross Omri. „An Introduction to Decentralized Finance (DeFi)“. In: (2021). URL: https://doi.org/10.7250/csimq.2021-26.03z.

[21] Olga Labazova. „Towards a Framework for Evaluation of Blockchain Implementations“. In: (2019). URL: ICIS%202019%20Proceedings.%2018..

[22] Javier Gutiérrez Castro, Edison Américo Huarsaya Tito, Luiz Eduardo Teixeira Brandão, and Leonardo Lima Gomes. „Crypto-assets portfolio optimization under the omega measure“. In: *The Engineering Economist* 65.2 (2020), pp. 114–134. eprint: https://doi.org/10.1080/0013791X.2019.1668098. URL: https://doi.org/10.1080/0013791X.2019.1668098.

[23] Xu Jiahua, Paruch Krzysztof, Cousaert Simon, and Feng Yebo. „SoK: Decentralized Exchanges (DEX) with Automated Market Maker (AMM) Protocols“. In: (2021). URL: https://arxiv.org/abs/2103.12732.

[24] Berenzon Dmitriy. *Constant Function Market Makers: DeFi's "Zero to One" Innovation*. 2020. URL: https://medium.com/bollinger-investment-group/constant-function-market-makers-defis-zero-to-one-innovation-968f77022159.

[25] Chainlink. *How to Bring More Capital and Less Risk to Automated Market Maker DEXs*. 2020. URL: https://blog.chain.link/challenges-in-defi-how-to-bring-more-capital-and-less-risk-to-automated-market-maker-dexs/.

[26] Adams Hayden, Zinsmeister Noah, and Robinson Dan. „Uniswap v2 Core“. In: (2020).

[27] Fernando Martinelli and Nikolai Mushegian. „A non-custodial portfolio manager, liquidity provider, and price sensor.“ In: (2019). URL: https://balancer.fi/whitepaper.pdf.

[28] Michael Egorov. „StableSwap - efficient mechanism for Stablecoin liquidity“. In: (2019). URL: https://curve.fi/files/stableswap-paper.pdf.

[29] Angeris Guillermo, Kao Hsien-Tang, Chiang Rei, Noyes Charlie, and Chitra Tarun. „An analysis of Uniswap markets“. In: (2019). URL: https://arxiv.org/abs/1911.03380.

[30] Tassy Martin and White David. *GROWTH RATE OF A LIQUIDITY PROVIDER'S WEALTH IN XY = c AUTOMATED MARKET MAKERS*. 2020. URL: https://math.dartmouth.edu/~mtassy/articles/AMM_returns.pdf.

[31] Jun Aoyagi. „Liquidity Provision by Automated Market Makers“. In: (2020). URL: https://ssrn.com/abstract=3674178.

[32] Daian Philip, Goldfeder Steven, Kell Tyler, Li Yunqi, Zhao Xueyuan, Bentov Iddo, Breidenbach Lorenz, and Juels Ari. *Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges*. 2019. URL: https://arxiv.org/abs/1904.05234.

[33] Pintail. *Uniswap: A good deal for liquidity providers?* 2019. URL: https://pintail.medium.com/uniswap-a-good-deal-for-liquidity-providers-104c0b6816f2.

[34] Messari. *State of Uniswap Q1 2022*. 2022. URL: https://messari.s3.amazonaws.com/pdf/4.11_State+of+Uniswap+(6).pdf.

Appendix

A

A.1 Definitions

A list of mathematical definition are presented from [7]

Definition A.1.1 *The expected return*[1] *of a security is given by*

$$\bar{R} = E[R] = \sum_{i=1}^{n} P_i R_i$$

where R_i *are the* n*-possible return's outcomes and* P_i *the respective probability of it occurring.*

In order to measure the dispersion of the outcomes and subsequently the risk that is faced, it is necessary to introduce the concept of *standard deviation*.

Definition A.1.2 *The variance of the return is given by*

$$\sigma^2 = \sum_{i=1}^{n} [P_i (R_i - \bar{R})^2]$$

Moreover, σ *is called standard deviation*

Therefore a smaller standard deviation implies that outcomes are within the same range and big leaps are unlikely to happen.
It is possible now to introduce the definition of *portfolio*

[1] Return: change in value of an investment over time

Definition A.1.3 *A portfolio is represented by a vector*

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}, \qquad x_1 \dots x_M \in \mathbf{R}$$

where $x_1 \dots x_M$ represent the amount of i-th security owned.

The expected return of the portfolio is

$$\bar{R}_P = E(\sum_i^M x_i R_{ij}) = \sum_i^M E(x_i R_{ij}) = \sum_i^M (x_i \bar{R}_i)$$

using linearity of expected returns. The variance of the portfolio is equal to

$$\sigma_P^2 = E(R_P - \bar{R}_P)^2 = E[\sum_i^M x_i (R_{ij} - \bar{R}_i)]^2$$

In the case of a 2-asset portfolio, this can be reduced to

$$\sigma_P^2 = x_1^2 \sigma_1^2 + 2 x_1 x_2 E[(R_{1j} - \bar{R}_1)(R_{2j} - \bar{R}_2)] + x_2^2 \sigma_2^2$$

The term $E[(R_{1j} - \bar{R}_1)(R_{2j} - \bar{R}_2)]$ is called *covariance* and can be indicated as σ_{12}. It is now clear that the covariance measures how two assets are linked to each other. Rescaling this measure in order to let it range in $[-1, 1]$ it is possible to introduce the *correlation coefficient*

$$\rho_{ik} = \frac{\sigma_{ik}}{\sigma_i \sigma k}$$

If asset i and j have a correlation coefficient of 0 it means that they are independent one from another, if it is 1 that they behave in the same way, while if it is -1 that they have opposite trends.
An investor strategy then would be to find the minimum variance portfolio, that is the portfolio that provides the highest return for the lowest volatility. Of course different compositions are to be expected depending on what is the highest level of risk an investor is willing to face. In [Figure A.1] it is possible to see how returns are vary based on portfolios' variances. The frontier of

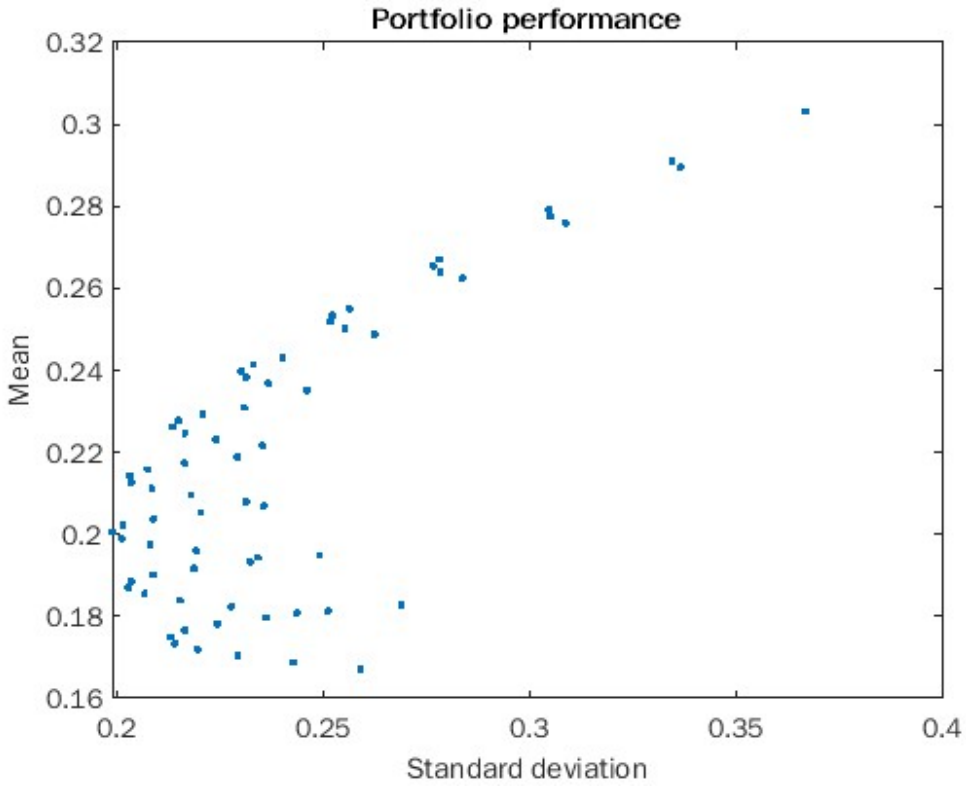

Figure A.1.: Examples of return and variance based on portfolios with three assets.

this set denotes the so called *efficient frontier*, where each pair represents the characteristic of a specific portfolio. Therefore an investor will choose their portfolio amongst those belonging to it. It is important to specify that efficient frontiers have different, but similar, aspects depending on what type of securities are considered: having a risk-free asset or considering short selling will allow for a more vast set of possible combinations. As it is noticeable, the efficient frontier draws a concave line, hence delimiting a convex set. Hence the the research for optimal portfolios reduces itself to the following convex problem

$$\begin{cases} max & \bar{R_P} = x_1\bar{R}_1 + \cdots + x_M\bar{R}_M \\ min & \sigma_P^2 = \sum_{i=1}^{M}(x_i^2\sigma_i^2) + \sum_{i=1}^{M}\sum_{\substack{j=1 \\ j\neq i}}^{M}(x_i x_j \sigma_{ij}) \end{cases} \tag{A.1}$$

that has theoretical solutions via the implicit function theorem. The solution is then obtain solving a set of linear equations that can be delegated to computers.