Arbitrage in Decentralized Exchanges

048866 - Algorithmic Challenges in Computer Networks and Blockchain
Sajy Khashab - Ron Marcus

1. Introduction

Decentralized Exchanges are used to trade cryptocurrencies on the Ethereum network. Trading on these exchanges opens arbitrage opportunities to benefit from the price discrepancy between different exchanges or different pools of the same exchange. Our goal is to take the swaps data from the three most used DEXes on the Ethereum blockchain within a specific period of time and analyze it to find, classify and collect information about the executed arbitrages.

2. Background

2.1. Ethereum Blockchain

Ethereum is a decentralized blockchain with smart contracts functionality. Each transaction on the blockchain can be either a coin transfer (of Ether) or an invocation of a smart contract. A smart contract is code that can be run on the blockchain, it has state that it carries between invocations and it has functions that can be called from other smart contracts or from regular accounts. An account is either an externally owned account or a smart contract. Externally owned accounts are owned by someone with the private key of that account, they can transfer Ether to other accounts or invoke smart contract functionality. Smart contract accounts are a specific deployment of smart contract code, they can only be called from other accounts but they have their own state that they can manipulate.

2.2. Decentralized Exchanges

Smart contract functionality enables the creation of Fungible Tokens (ERC-20). One use of these tokens is creating other cryptocurrencies than the native Ether currency of the Ethereum network. For example, USDC is a cryptocurrency that is referred to as a stable-coin since one token of it can always be redeemed for US$1.00, giving it a stable price. Decentralized Exchanges are currency exchanges that run as smart contracts, on the Ethereum network for example. These exchanges can be used to trade tokens of different types. The most well-known exchanges on Ethereum are UniSwap and SushiSwap. UniSwap has 3 versions, and SushiSwap is just a different deployment of UniSwapV2. These exchanges have an automated market maker design, unlike traditional order book exchanges. In this design liquidity providers invest their tokens in the exchange, which allows others to trade tokens and pay fees to the liquidity providers (and the dex). These exchanges typically enable creating pools that can be used to trade any pair of Fungible Tokens, therefore they are extensively used and they constitute a significant portion of the Ethereum blockchain.

2.3. Arbitrage in Decentralized Exchanges

As defined in Investopedia[1], 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. Because the relative price of the two tokens in a specific pool can only be changed through trading, trading tokens can create arbitrage opportunities that are vital for matching the price of tokens on these exchanges. Arbitrages then help the price of tokens on different exchanges to converge, and the price on different pools of the same exchange to also converge to the market's price.

[1] https://www.investopedia.com/terms/a/arbitrage.asp

3. Arbitrage in Ethereum-Based Decentralized Exchanges

In our study, we focused on analyzing arbitrage opportunities using the three most common decentralized exchanges on the Ethereum Blockchain: UniSwapV2, UniSwapV3 and SushiSwap. Arbitrage using these exchanges can take on many forms, we focus on **Cyclic Swap Sequence** - CSSs for short. A CSS is basically a sequence of N token swaps satisfying:

Swap 1: Exchange ($n_{1,in}$, T_1) for ($n_{1,out}$, T_2) on P_1 of E_1
Swap 2: Exchange ($n_{2,int}$, T_2) for ($n_{2,out}$, T_3) on P_2 of E_2
…
Swap N: Exchange ($n_{N,in}$, T_N) for ($n_{N,out}$, T_1) on P_N of E_N

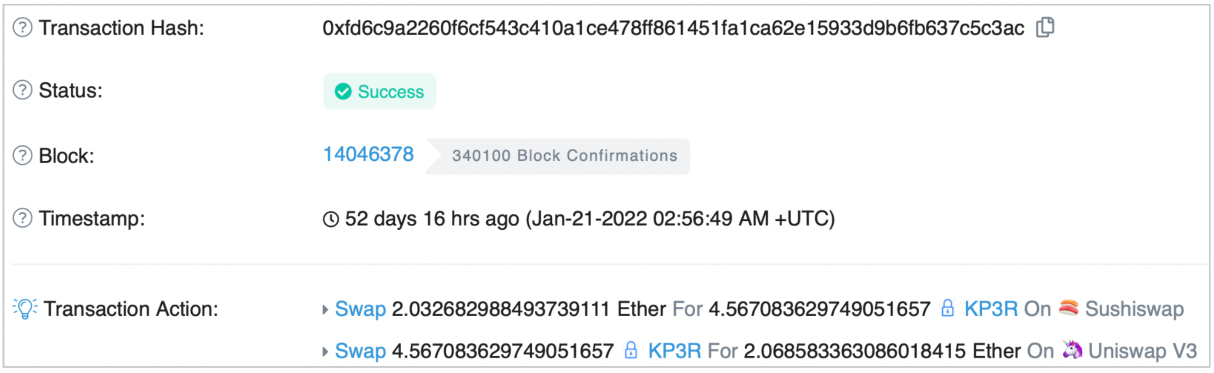
Swap i is the exchange of $n_{i,in}$ tokens of type T_i for $n_{i,out}$ tokens of type T_{i+1} using pool P_i of exchange E_i. The exchanges we focus on enable users to create pools for any pair of two Fungible Tokens[2]. UniSwapV2 (and SushiSwap[3]) enables creating at most a single pool for each pair of different tokens, but UniSwapV3 introduced multiple pools for each token pair, each with a different swapping fee.
As can be seen in the definition, a CSS ends with the same token type as the token type that it started with (T_1). To consider a CSS an **arbitrage**, it needs to satisfy these conditions:

1. Atomicity: All swaps need to be executed in transactions of the same ethereum block.
2. Chaining: The input amount of swap i ($n_{i,in}$) needs to be less than or equal to the output amount of swap i-1 ($n_{i-1,out}$).
3. Basic Profitability: At the end, the net-change for each token for the account is non-negative, and positive for at least one token.

Condition 3 basically says that the balance for the account executing the sequence didn't decrease on any of the used tokens, meaning that the account has more tokens (from one or more types) than it started with. The "basic profitability" doesn't necessarily mean that sequence was actually profitable for the arbitrageur, since it doesn't take the transaction fees into account. Conditions 1 and 2 are useful simplifications to the analysis. We note that these are assumptions that we made to ease with the analysis, it is possible however to find arbitrages that contradict one or more of them. Since our definition of an arbitrage is somewhat flexible, we arrived at multiple classifications of arbitrages that are worth examining more closely, which we do in the following.

3.1. Single-Transaction vs Multi-Transaction

The most simple classification is to the number of transactions that the swaps sequence spans. A **single-transaction** arbitrage is one whose swaps are all in the same ethereum transaction. It is basically an atomic list of swaps that is executed in the same transaction without interleaving with other transactions. It is based on the fact that the current state of pools enables a sequence of swaps that ends with a net-profit. The following transaction is an example of typical single-transaction arbitrage:

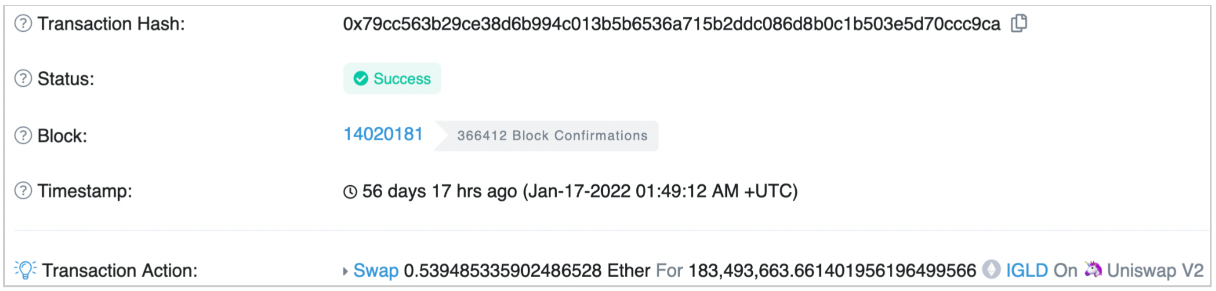


Transaction Hash:	0xfd6c9a2260f6cf543c410a1ce478ff861451fa1ca62e15933d9b6fb637c5c3ac
Status:	Success
Block:	14046378 340100 Block Confirmations
Timestamp:	52 days 16 hrs ago (Jan-21-2022 02:56:49 AM +UTC)
Transaction Action:	Swap 2.032682988493739111 Ether For 4.567083629749051657 KP3R On Sushiswap
	Swap 4.567083629749051657 KP3R For 2.068583363086018415 Ether On Uniswap V3

This transaction[4] swapped Ether for KP3R using SushiSwap, and then swapped back in UniSwapV3.

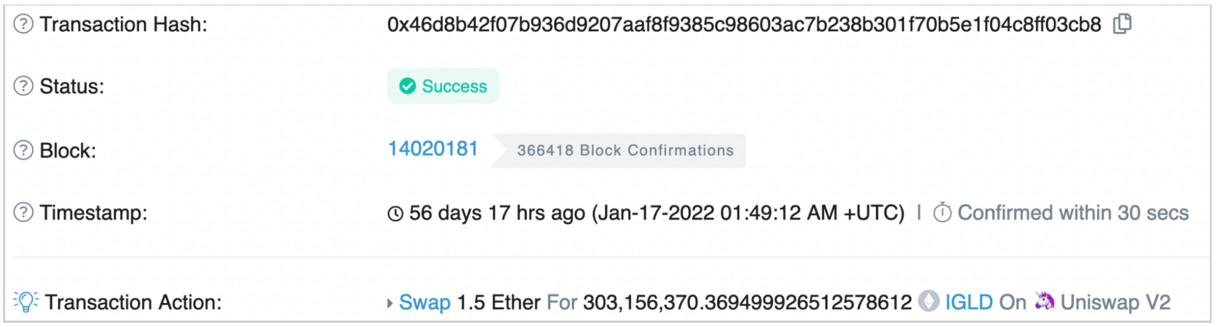
[2] Any token based on the ERC-20 standard.
[3] SushiSwap is a different deployment of the same contracts as UniSwapV2.
[4] https://etherscan.io/tx/0xfd6c9a2260f6cf543c410a1ce478ff861451fa1ca62e15933d9b6fb637c5c3ac

In contrast, **multi-transaction** arbitrages span multiple transactions. It is also known as a **sandwich-attack**, it starts by the miner identifying some other account's pending swap for exchanging token A to B. The miner then creates two new transactions, the first comes before the victim swap and it buys some amount of token B. The second transaction comes after the victim swap and it sells the purchased tokens of type B; achieving net-profit since after the victim swap, the price of B increases in the same pool. The following is an example of a typical multi-transaction arbitrage:



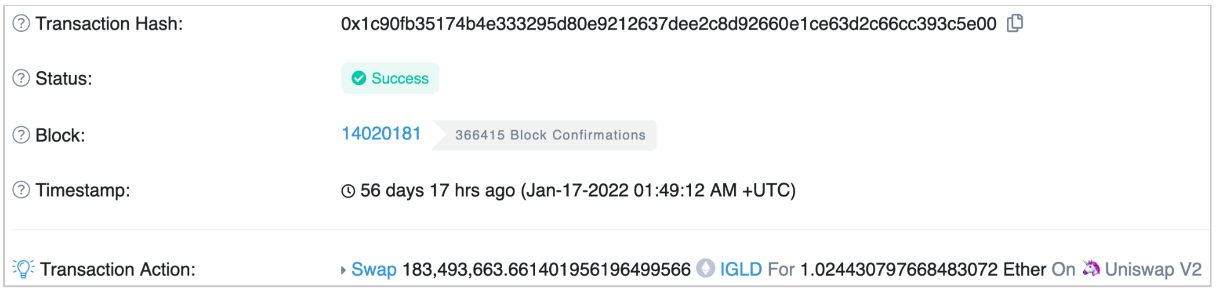
Transaction Hash:	0x79cc563b29ce38d6b994c013b5b6536a715b2ddc086d8b0c1b503e5d70ccc9ca
Status:	Success
Block:	14020181 366412 Block Confirmations
Timestamp:	56 days 17 hrs ago (Jan-17-2022 01:49:12 AM +UTC)
Transaction Action:	Swap 0.539485335902486528 Ether For 183,493,663.661401956196499566 IGLD On Uniswap V2

This transaction[5] is the first part of the sandwich, it swaps Ether for IGLD on UniSwapV2.



Transaction Hash:	0x46d8b42f07b936d9207aaf8f9385c98603ac7b238b301f70b5e1f04c8ff03cb8
Status:	Success
Block:	14020181 366418 Block Confirmations
Timestamp:	56 days 17 hrs ago (Jan-17-2022 01:49:12 AM +UTC) | Confirmed within 30 secs
Transaction Action:	Swap 1.5 Ether For 303,156,370.369499926512578612 IGLD On Uniswap V2

This transaction[6] is the victim, it swaps Ether for IGLD on the same pool as well.



Transaction Hash:	0x1c90fb35174b4e333295d80e9212637dee2c8d92660e1ce63d2c66cc393c5e00
Status:	Success
Block:	14020181 366415 Block Confirmations
Timestamp:	56 days 17 hrs ago (Jan-17-2022 01:49:12 AM +UTC)
Transaction Action:	Swap 183,493,663.661401956196499566 IGLD For 1.024430797668483072 Ether On Uniswap V2

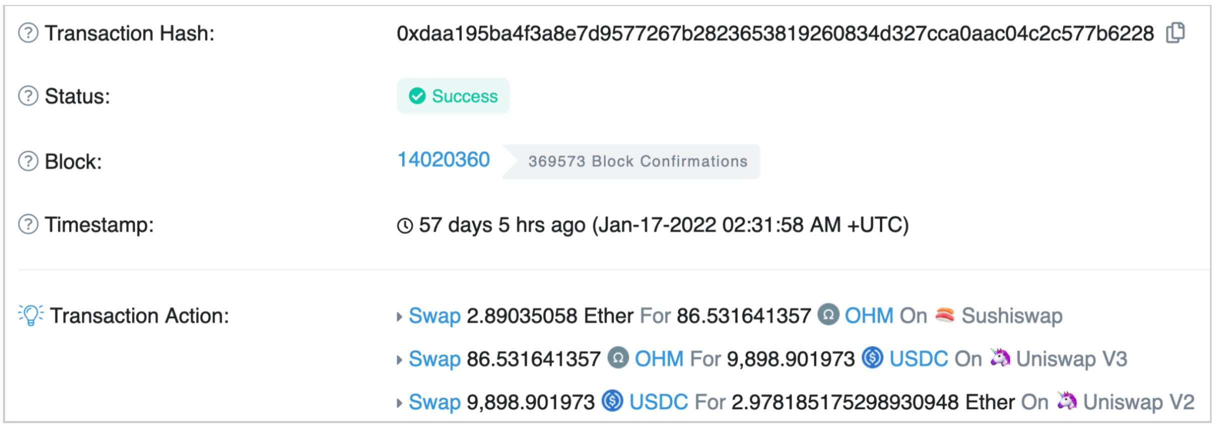
This transaction[7] is the second part of the sandwich, it swaps the amount that the first transaction got back to *more* Ether than the first transaction started with on the same pool. These three transactions appear in this order as the first three transactions of block 14020181[8].
The main difference between these two kinds is the identity of the arbitrageur. A single-transaction arbitrage can be executed by any account, it just needs to create the transaction and broadcast it on the network to miners. However, because of the synchronization needed, a multi-transaction arbitrage can only be executed by the miner of the block since it needs to carefully place the two transactions before and after the victim transaction in the mined block.

3.2. Same-Exchange vs Cross-Exchange

Basic arbitrages profit from the price difference between similar pools - same token pairs - in different exchanges. These are **cross-exchange** arbitrages since they use swaps in different exchanges. The

[5] https://etherscan.io/tx/0x79cc563b29ce38d6b994c013b5b6536a715b2ddc086d8b0c1b503e5d70ccc9ca
[6] https://etherscan.io/tx/0x46d8b42f07b936d9207aaf8f9385c98603ac7b238b301f70b5e1f04c8ff03cb8
[7] https://etherscan.io/tx/0x1c90fb35174b4e333295d80e9212637dee2c8d92660e1ce63d2c66cc393c5e00
[8] https://etherscan.io/txs?block=14020181&p=8

example above for a single-transaction arbitrage is of this kind, but arbitrages on more than two exchanges are also common:



Transaction Hash: 0xdaa195ba4f3a8e7d9577267b2823653819260834d327cca0aac04c2c577b6228

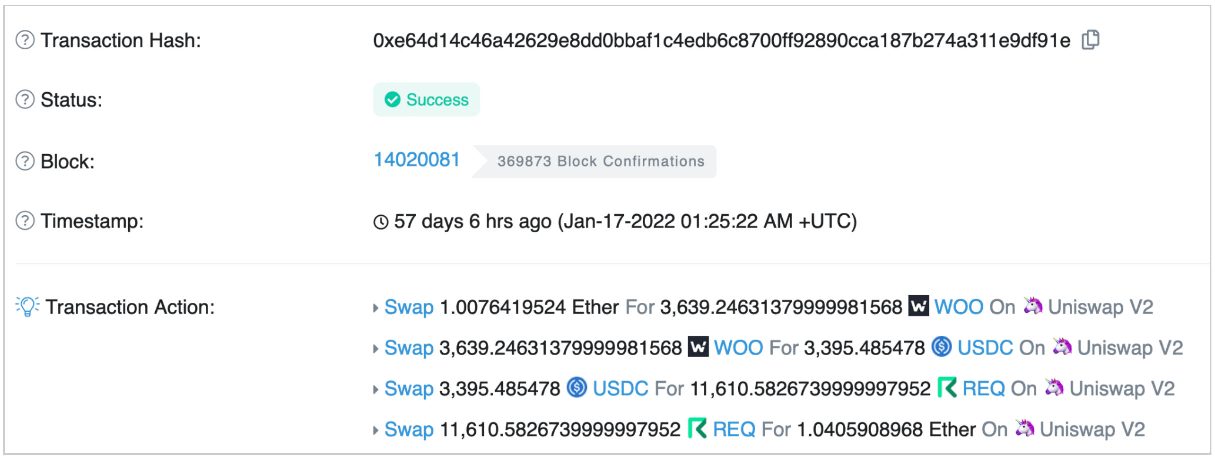
Status: Success

Block: 14020360 369573 Block Confirmations

Timestamp: 57 days 5 hrs ago (Jan-17-2022 02:31:58 AM +UTC)

Transaction Action:
- Swap 2.89035058 Ether For 86.531641357 OHM On Sushiswap
- Swap 86.531641357 OHM For 9,898.901973 USDC On Uniswap V3
- Swap 9,898.901973 USDC For 2.978185175298930948 Ether On Uniswap V2

This transaction[9] did swaps on 3 different pools, each on a different exchange, and profited on Ether. Arbitrages can also benefit from differences between different pools of the same exchange, these are **same-exchange** arbitrages. These arbitrages pass through two or more pools of the same exchange to profit, for example:



Transaction Hash: 0xe64d14c46a42629e8dd0bbaf1c4edb6c8700ff92890cca187b274a311e9df91e

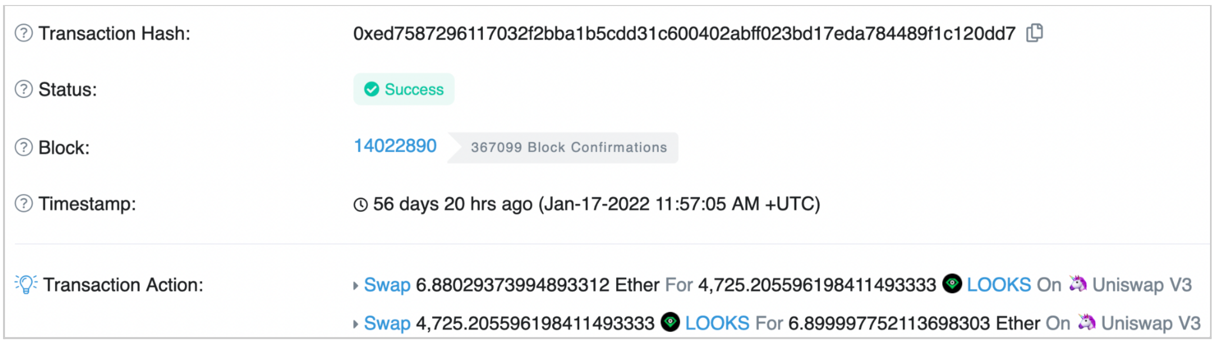
Status: Success

Block: 14020081 369873 Block Confirmations

Timestamp: 57 days 6 hrs ago (Jan-17-2022 01:25:22 AM +UTC)

Transaction Action:
- Swap 1.0076419524 Ether For 3,639.24631379999981568 WOO On Uniswap V2
- Swap 3,639.24631379999981568 WOO For 3,395.485478 USDC On Uniswap V2
- Swap 3,395.485478 USDC For 11,610.5826739999997952 REQ On Uniswap V2
- Swap 11,610.5826739999997952 REQ For 1.0405908968 Ether On Uniswap V2

This transaction[10] swapped on 4 different pools of UniSwapV2 to profit in Ether.
On both UniSwapV2 and sushiswap, each single-transaction single-exchange arbitrage involves at least three tokens. That is because wapping token A to B on the same pool and immediately swapping it back to A on the same pool can't achieve any profit. The second swap would result in the initial amount of the first swap, minus swapping-fee paid for using the pools. Additionally, For UniSwapV2 and SushiSwap, each pair of tokens could have at most one pool. However, UniSwapV3 introduced multiple pools for each pair, each with a different swapping fee. Therefore, it opened the possibility of doing single-transaction arbitrage on the same exchange using only two tokens, utilizing the different pools for the pair. For example:



Transaction Hash: 0xed7587296117032f2bba1b5cdd31c600402abff023bd17eda784489f1c120dd7

Status: Success

Block: 14022890 367099 Block Confirmations

Timestamp: 56 days 20 hrs ago (Jan-17-2022 11:57:05 AM +UTC)

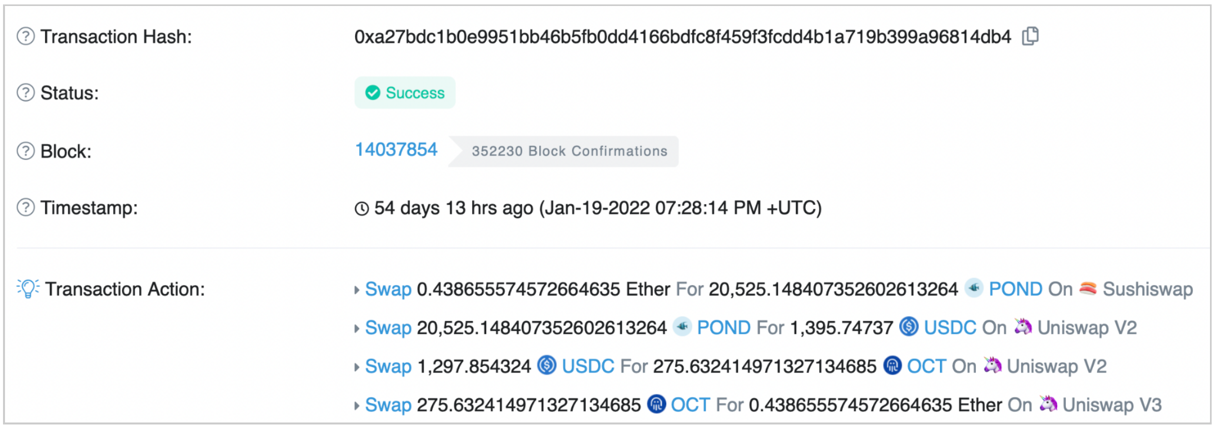
Transaction Action:
- Swap 6.880293739948893312 Ether For 4,725.205596198411493333 LOOKS On Uniswap V3
- Swap 4,725.205596198411493333 LOOKS For 6.899997752113698303 Ether On Uniswap V3

[9] https://etherscan.io/tx/0xdaa195ba4f3a8e7d9577267b2823653819260834d327cca0aac04c2c577b6228
[10] https://etherscan.io/tx/0xe64d14c46a42629e8dd0bbaf1c4edb6c8700ff92890cca187b274a311e9df91e

This transaction[11] used UniSwapV3 to swap Ether to LOOKS and immediately back to Ether. However, the first used the 3% fee pool, and the second used the 10% fee pool.

3.3. Owned-Token vs “Loaned”-Token

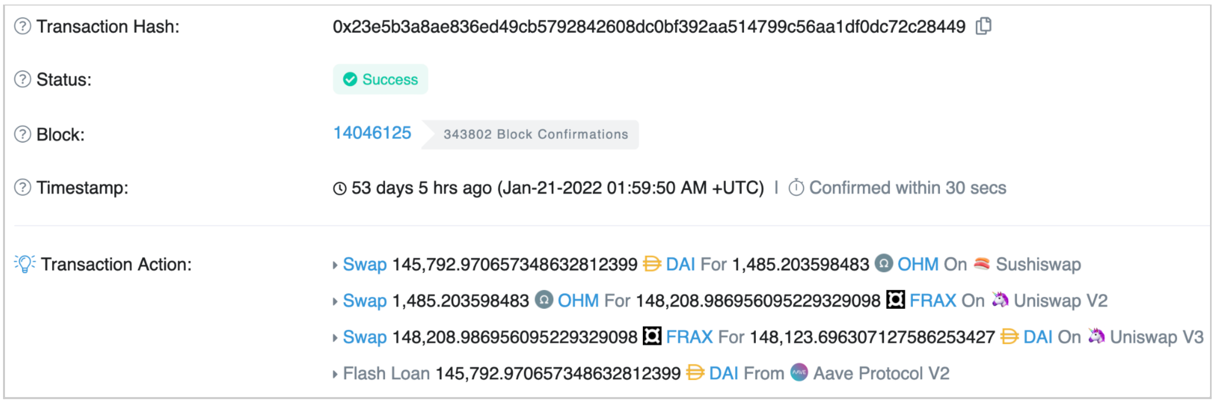
Arbitrages above are **owned-token** arbitrages, meaning the arbitrageur owned some amount of token A and he ran the arbitrage to increase the amount of token A that he owns. However, because of the way the pools’ smart contracts work, the arbitrageur doesn’t have to own any amount of tokens for executing the arbitrage. Basically, when a smart contract (the arbitrageur in this case) calls the *swap* call of a pool, the pool first transafters the output amount to the caller and then calls a callback function the caller should have implemented in which he transfers the input amount to the pool, the pool then checks if during the callback function the caller indeed transferred the needed amount. However, this callback function doesn’t have to immediately transfer the needed tokens to the pool, instead, it can start another swap with a different pool in order to complete an arbitrage cycle and then transfer the needed amount. This enables the arbitrageur to close an arbitrage without owning any amount of tokens beforehand (other than for ethereum transaction fees) - a **“loaned”-token** arbitrage. For example:



Transaction Hash:	0xa27bdc1b0e9951bb46b5fb0dd4166bdfc8f459f3fcdd4b1a719b399a96814db4
Status:	Success
Block:	14037854 352230 Block Confirmations
Timestamp:	54 days 13 hrs ago (Jan-19-2022 07:28:14 PM +UTC)
Transaction Action:	Swap 0.438655574572664635 Ether For 20,525.148407352602613264 POND On Sushiswap
	Swap 20,525.148407352602613264 POND For 1,395.74737 USDC On Uniswap V2
	Swap 1,297.854324 USDC For 275.632414971327134685 OCT On Uniswap V2
	Swap 275.632414971327134685 OCT For 0.438655574572664635 Ether On Uniswap V3

In this transaction[12], the arbitrageur profited ~97 USDC tokens using an arbitrage cycle. However, if we look at the token transfers (not shown - see link) we can see that the executing smart contract only transferred out ~1297 USDC tokens **after** it got paid ~1395 USDC tokens.

We call this a **“loaned”-token** arbitrage since it was executed with tokens that the arbitrageur didn’t own. However, the arbitrageur didn’t have to actually loan tokens from any entity and pay fees, it was purely possible because of the callback design of the pools’ smart contract. Less sophisticated arbitrages that actually loan tokens from other entities (and pay fees) do exist. For example:



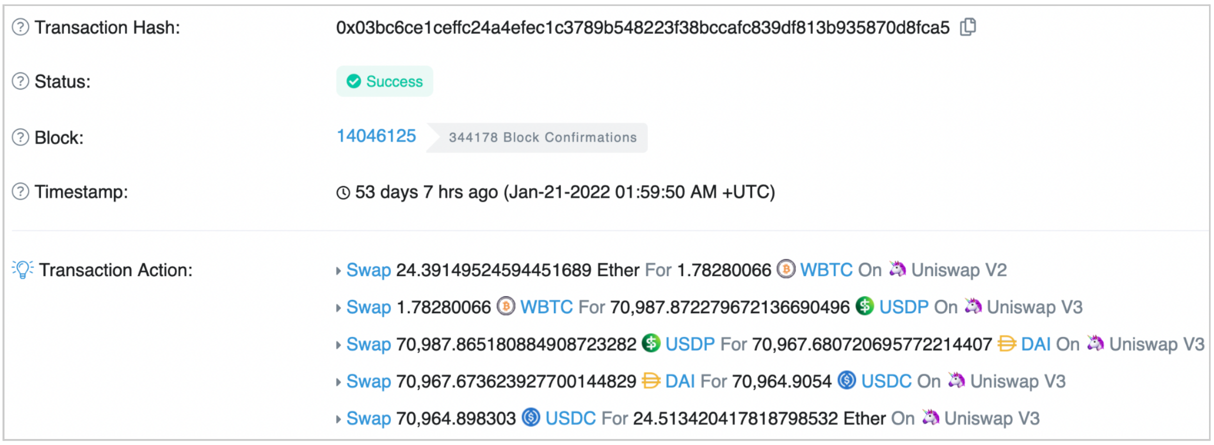
Transaction Hash:	0x23e5b3a8ae836ed49cb5792842608dc0bf392aa514799c56aa1df0dc72c28449
Status:	Success
Block:	14046125 343802 Block Confirmations
Timestamp:	53 days 5 hrs ago (Jan-21-2022 01:59:50 AM +UTC) | Confirmed within 30 secs
Transaction Action:	Swap 145,792.970657348632812399 DAI For 1,485.203598483 OHM On Sushiswap
	Swap 1,485.203598483 OHM For 148,208.986956095229329098 FRAX On Uniswap V2
	Swap 148,208.986956095229329098 FRAX For 148,123.696307127586253427 DAI On Uniswap V3
	Flash Loan 145,792.970657348632812399 DAI From Aave Protocol V2

[11] https://etherscan.io/tx/0xed7587296117032f2bba1b5cdd31c600402abff023bd17eda784489f1c120dd7
[12] https://etherscan.io/tx/0xa27bdc1b0e9951bb46b5fb0dd4166bdfc8f459f3fcdd4b1a719b399a96814db4

This transaction[13] executed an arbitrage using the DAI token, however it first borrowed ~145,792 tokens from Aave (a decentralized liquidity market). Then it executed an arbitrage cycle and ended up with ~148,123 tokens, profiting ~2,331 tokens. Then it had to pay back the borrowed tokens with a fee of ~131 tokens, achieving net-profit.

3.4. Precise vs Imprecise

The arbitrages we show above are **precise** in the sense that they profit from exactly one token. However, arbitrages can profit in multiple tokens, making them **imprecise**. For example:



Transaction Hash:	0x03bc6ce1ceffc24a4efec1c3789b548223f38bccafc839df813b935870d8fca5
Status:	✓ Success
Block:	14046125 344178 Block Confirmations
Timestamp:	53 days 7 hrs ago (Jan-21-2022 01:59:50 AM +UTC)
Transaction Action:	Swap 24.39149524594451689 Ether For 1.78280066 WBTC On Uniswap V2
	Swap 1.78280066 WBTC For 70,987.872279672136690496 USDP On Uniswap V3
	Swap 70,987.865180884908723282 USDP For 70,967.680720695772214407 DAI On Uniswap V3
	Swap 70,967.673623927700144829 DAI For 70,964.9054 USDC On Uniswap V3
	Swap 70,964.898303 USDC For 24.513420417818798532 Ether On Uniswap V3

This transaction[14] executed 5 swaps and profited in 4 different tokens. The outputs of swaps 2,3 and 4 are not equal to the inputs of their next swaps. This phenomenon happens mainly due to the different ways in which the swap functions can be called, for example, the caller can set a specific input amount and get whatever output amount he can, or he can set a specific output amount and pay whatever input needed for it. A common case[15] is where there is profit in one main token, and a profit of exactly the smallest unit of another token, which we suspect is a result of mathematical operations approximation.

4. Data Collection and Analysis

Our goal was to find all arbitrages in a given portion of the blockchain and analyze them as we do in the next section. Since we don't have a running Ethereum node, we had to use free publicly available APIs to run queries on the blockchain. We mainly used Etherscan[16], a blockchain explorer for Ethereum that provides a free (but very limited) API to access specific blockchain data. Since the API we used was very restricted, in both request rate and possible queries, we had to concentrate on a small portion of the blockchain. Specifically, we took swap information of 30,000 blocks, starting at block number 14,020,000 (Jan-17-2022 01:07:37 AM +UTC) and ending at block number 14,050,000 (Jan-21-2022 04:30:45 PM +UTC). This range contains swaps from all exchanges we considered, and we choose it since an arbitrage that we already knew existed falls in the center of it.
We collected data from the 3 exchanges: UniSwapV2, UniSwapV3 and SushiSwap. The main query we used extracts **Ethereum Log Records** from a specific smart contract within a given block range. A smart contract is able to emit topic-based log events using specific EVM opcodes. For example, a pool's swap contract emits an event on each swap that contains the swap information. We can use the queries in order to find all swaps information from the logged event records. In the following subsections we explain how we extracted and analyzed the swap data.

[13] https://etherscan.io/tx/0x23e5b3a8ae836ed49cb5792842608dc0bf392aa514799c56aa1df0dc72c28449
[14] https://etherscan.io/tx/0x03bc6ce1ceffc24a4efec1c3789b548223f38bccafc839df813b935870d8fca5
[15] https://etherscan.io/tx/0x32496ba8baa5d4a6176b5e3244ebf2888207ccee6b626a1656756143d57bd7f2
[16] https://etherscan.io

4.1. Extract Pools

Each one of the exchanges has a **Factory** smart contract whose main use is deploying **Pool** smart contracts as needed. Anyone can create a pool for any pair of Fungible Tokens if it doesn't exist yet. The three exchanges we examined: UniSwapV2, UniSwapV3 and SushiSwap had 61626, 5635 and 2529 pools created at the portion of the blockchain we analyzed. Luckily, the **Factory** smart contract emits a *PoolCreated* event whenever a pool is created with its deployment address. Therefore, we took the Factories' addresses from the official websites of the DEXes and found all of the pools that they created using topic-based log queries using the factory contract address. This gave us a list of all pools that could have swaps on these exchanges within blocks we examined.

4.2. Extract Swaps

Given the list of pools' addresses, we again used the fact that a *Swap* event is emitted whenever a swap is executed in the pool smart contract. Therefore, we could go over all the pools and extract all swap events in the needed blocks. This is the process that took the most time since we had limited access to the API and because the number of pools is very large requiring a few hours to go through all of them. The information we got from the swap event contains the identity of the transaction where the swap happened (block number, transaction hash,..), the swap information (token amounts) and transaction fee parameters. Our arbitrage analysis is based on the information extracted in this stage.

4.3. Extract Symbols and Prices

Given the list of swaps we could perform the analysis, but we had 2 issues. The first is that all token identifiers are actually their smart contract addresses, therefore, we wanted some kind of mapping from the address to the Symbol so we can present it more friendly. The second issue is that the profits we get are in token units, this prevents us from comparing values of different tokens or comparing the profits against the paid fees (in Ether). Therefore, we used Alchemy[17] for converting the token account addresses to known Symbols, and then CryptoCompare[18] for getting the current day prices of different tokens. The prices we got aren't the trading prices at the time of the arbitrage, but they serve as a good indication of the profitability of arbitrages nonetheless.

4.4. Analysis

With all the information we extracted in hand, we ran code to find all of the arbitrages as defined in section 3. We then analyzed the arbitrages in various ways to get the results we present next.

5. Results

5.1. Scope

The scope of the results is 30,000 blocks, as we mentioned in section 4.

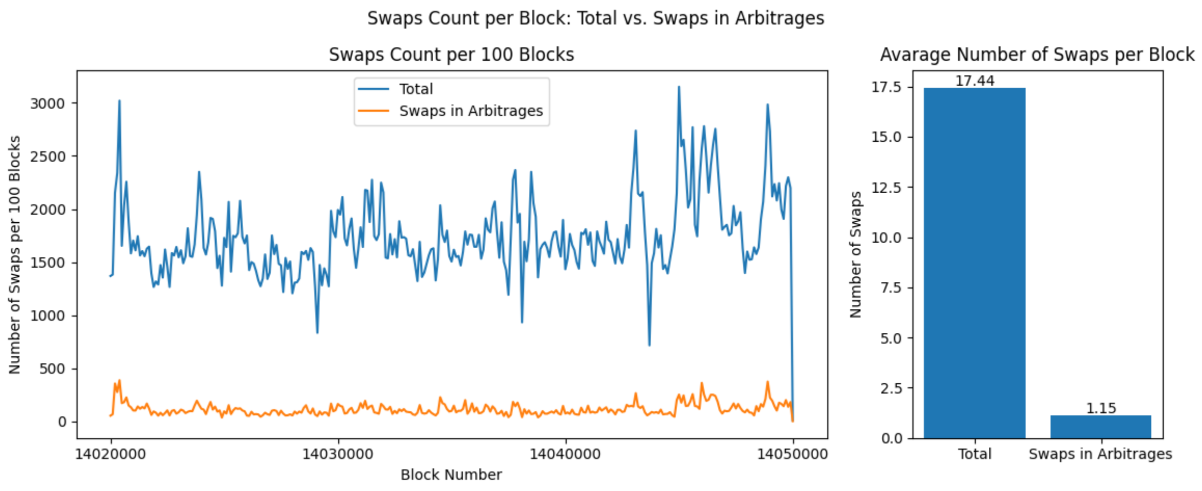
- Start block 14020000 : Jan-17-2022 01:07:37 AM +UTC
- End block 14050000 : Jan-21-2022 04:30:45 PM +UTC
- These blocks are referred to as "the blocks".

The blocks include 406,974 transactions. The majority of the blocks (95%) include at least one transaction with swaps. The blocks include a total of 523,343 swaps. The swaps were made in 5301 pools.

[17] https://www.alchemy.com
[18] https://www.cryptocompare.com

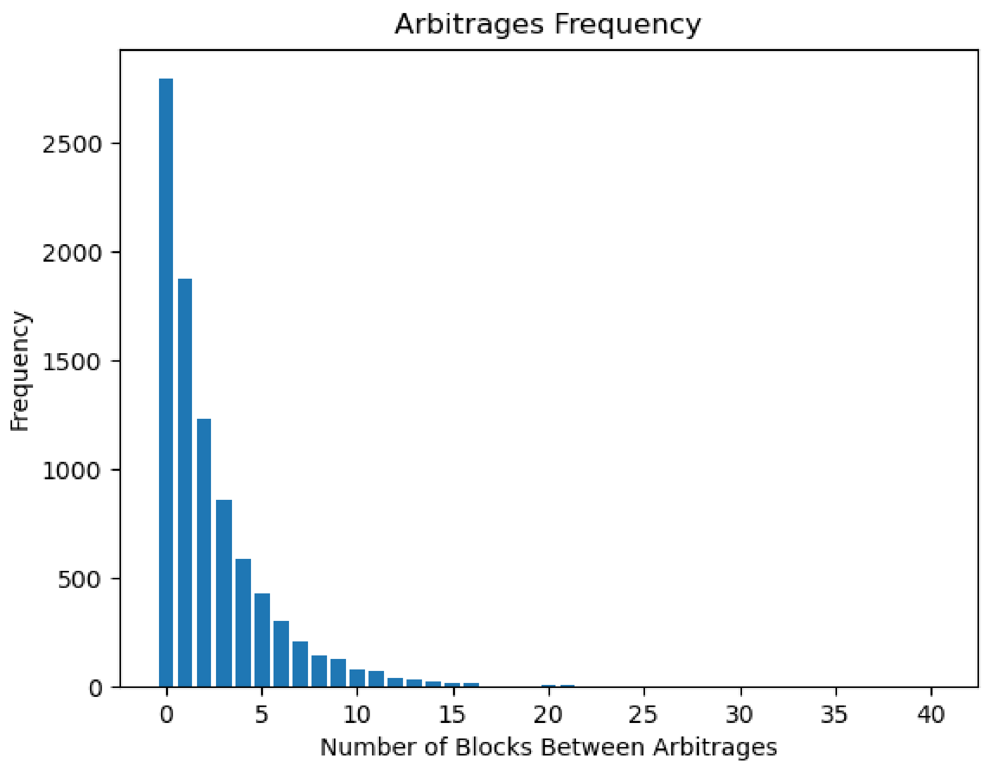
5.2. Arbitrage Incidence

We measure the arbitrage Incidence by comparing the count of swaps used for arbitrage against the total count of swaps. The following graph shows the **swaps count per a bucket blocks**:

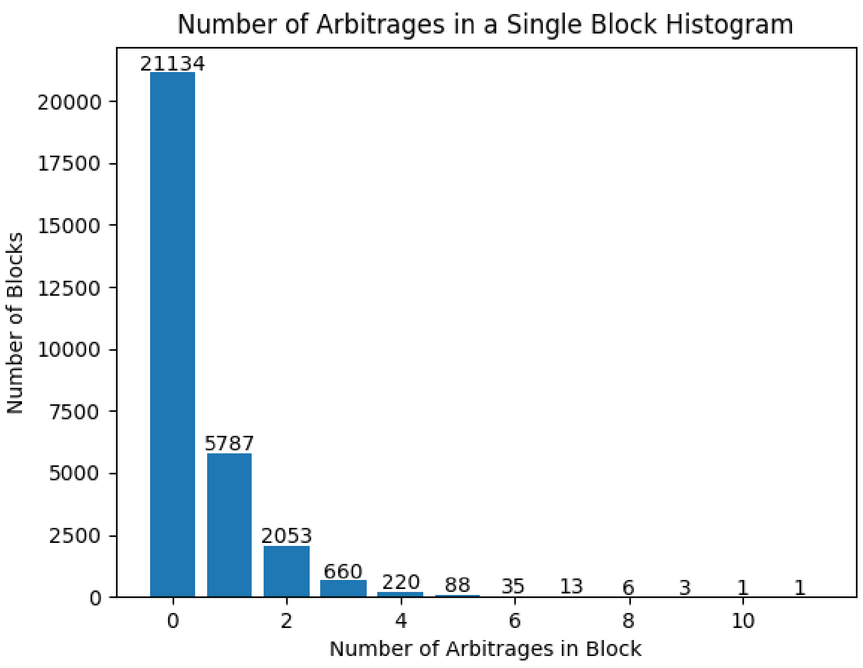


We can see that on average, each block contains about 17.44 swaps, out of which 1.15 are used in arbitrages. The number of swaps used in arbitrages is 34,375, which is 6.57% of the total swaps. Another thing to note in the left graph is that as there is more trading, more arbitrages are performed, this can be seen in the peaks of the ‘Total’ line matching the peaks of the ‘Swaps in Arbitrages’ line’.

The number of blocks that had at least one arbitrage is 8867, which is 29.5% of the total number of blocks. Another important indication is the **frequency of arbitrages** across blocks. The following graph shows a histogram of the number of blocks between blocks that contain arbitrage:



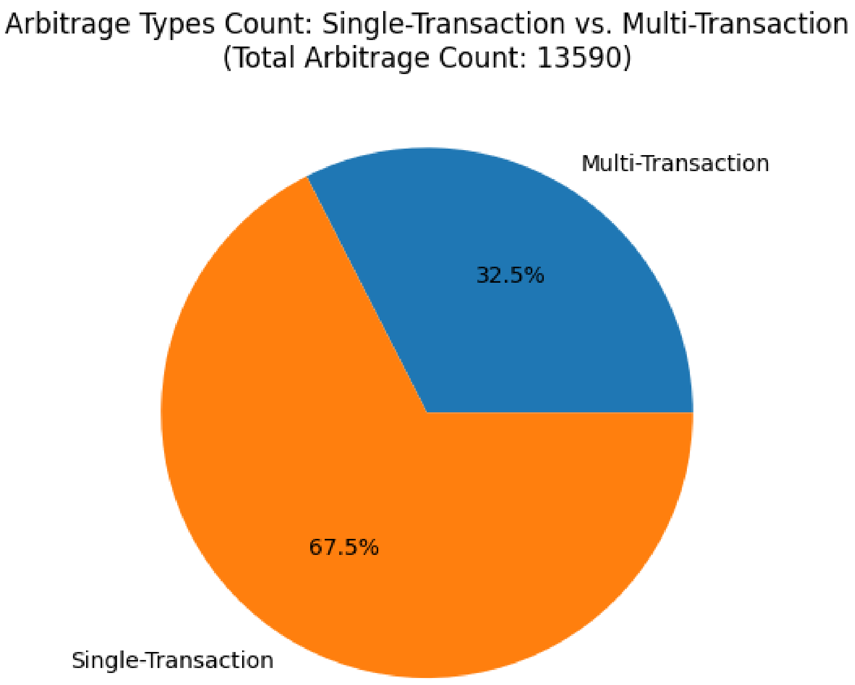
We can see that the leading interval size is 0 blocks, which means that arbitrages were found in adjacent blocks. Additionally, The median interval size is 1 block. However, some blocks contain multiple arbitrages, as the following graph shows:



The maximum is 11 arbitrages per block. However, we can see that usually there is no more than a few arbitrages per block, and that blocks with more than 4 arbitrages are scarce.

5.3. Single-Transaction vs Multi-Transaction

As described in section 3.1, we distinguish between two arbitrage kinds: single-transaction and multiple-transaction. The following graph shows the ratio between them, in terms of arbitrage count:



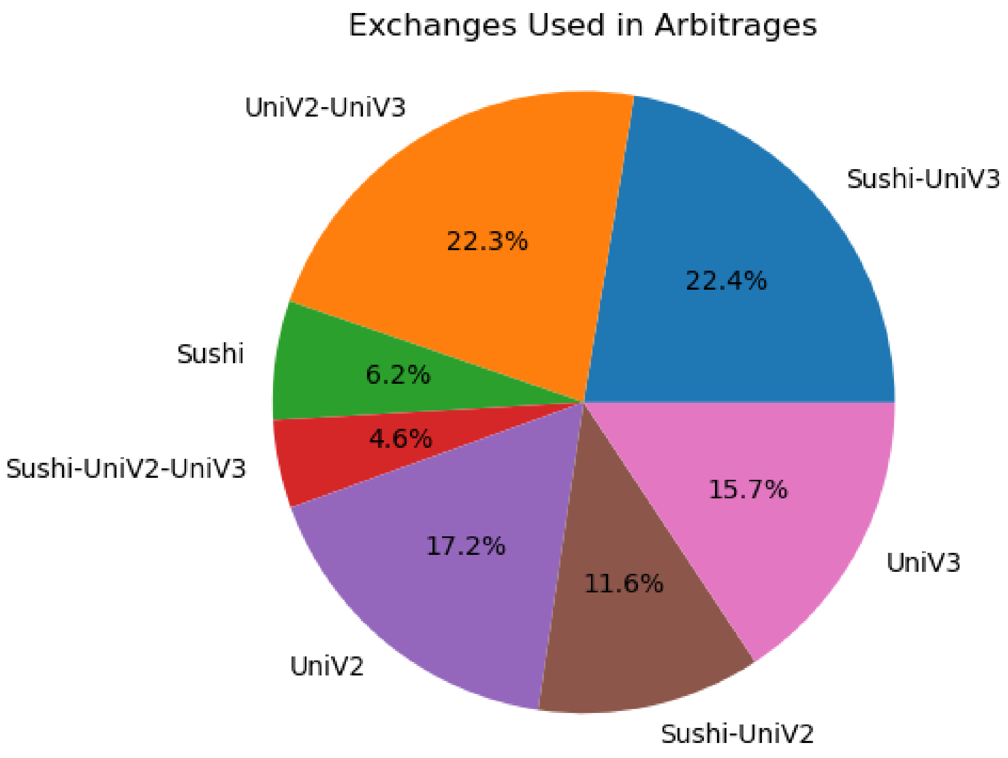
There are 13,590 arbitrages in total, out of which 4,414 are multiple-transaction, and 9176 are single-transaction arbitrages. We can see that **there are 2 times more single-transaction arbitrages than multi-transaction arbitrages**.

5.4. Used Exchanges

Arbitrages might include swaps in pools of a single exchange or multiple exchanges, as described in section 3.2. We are interested both in the uages of each exchange and in the incidence of the different combinations. Exchanges' usage were measured in terms of swap count.

Exchange	Number of Swaps for Arbitrages	Swaps Percentage
SushiSwap	78,428	15%
UniSwapV3	179,802	34.3%
UniSwapV2	265,113	50.7%

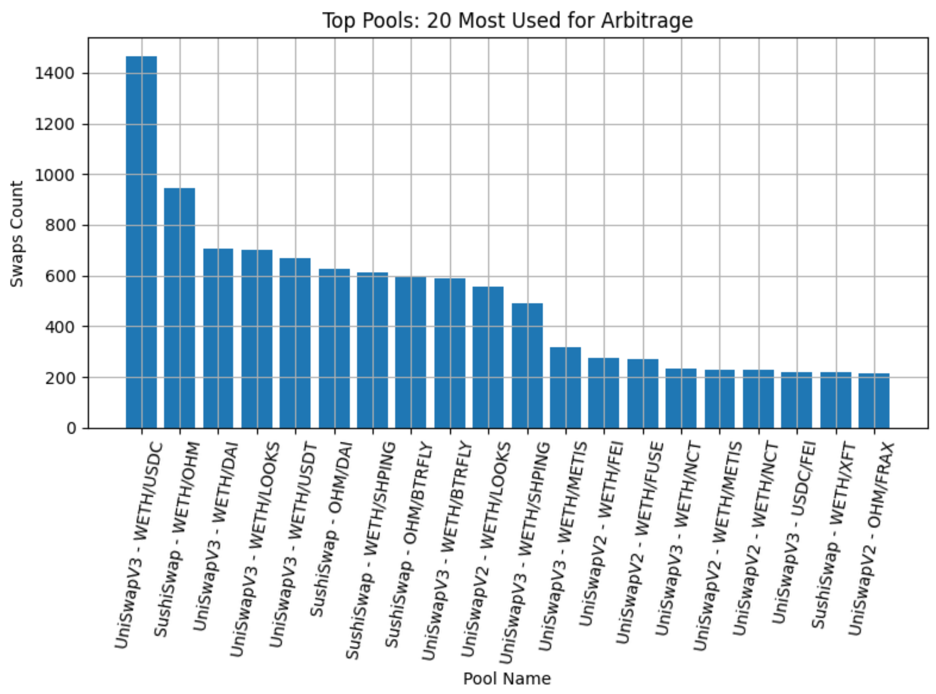
We can see that most of the swaps are made in uniswapv2. The following graphs shows the incidence of the different combinations of exchanges used in arbitrages, in terms of number of arbitrages:



We can see that the most used combination is SushiSwap with UniSwapV3, followed by UniSwapV2 with UniSwapV3. This can be explained with the fact that UniSwapV3 is the newest of exchanges, therefore it has many arbitrage opportunities as users migrate to it. The most scarce combination includes all of the three exchanges.

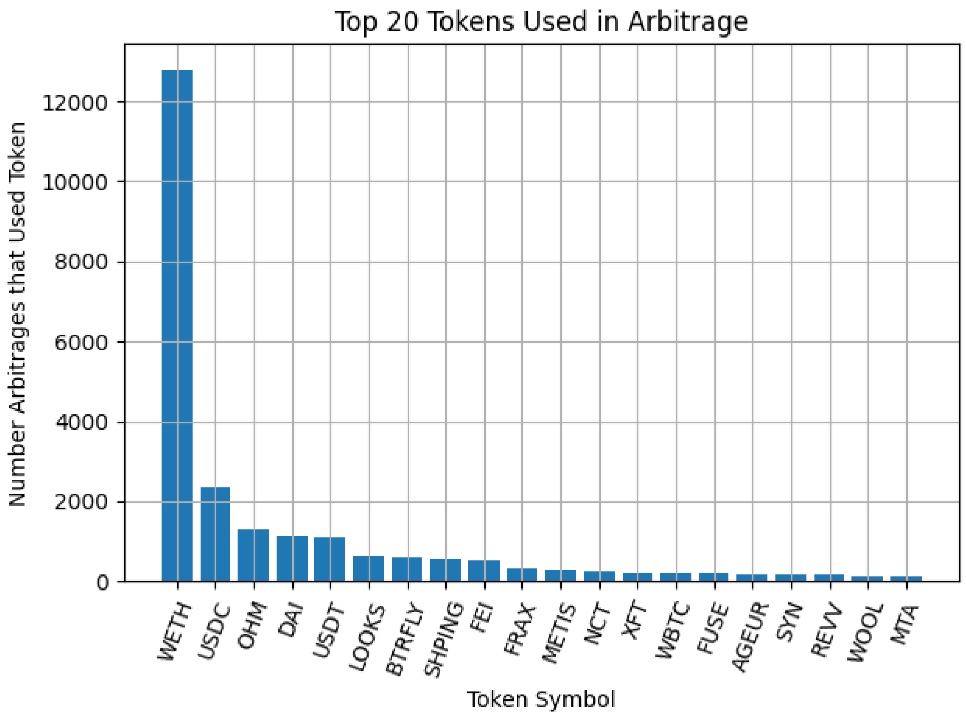
5.5. Used Pools and tokens

The following graphs shows the 20 most used pools for arbitrages, out of 2202 used pools:



We can see that the most used pool is between Ether and USD Coin; USD Coin is a digital stable-coin that is pegged to the United States dollar. The 2nd and 3rd pools are similar - swaps between Ether and popular stable-coins. Thus, if we assume that tokens liquidity correlates to arbitrage opportunities in their corresponding pools, then these stable-coins pools see most trades and liquidity changes.

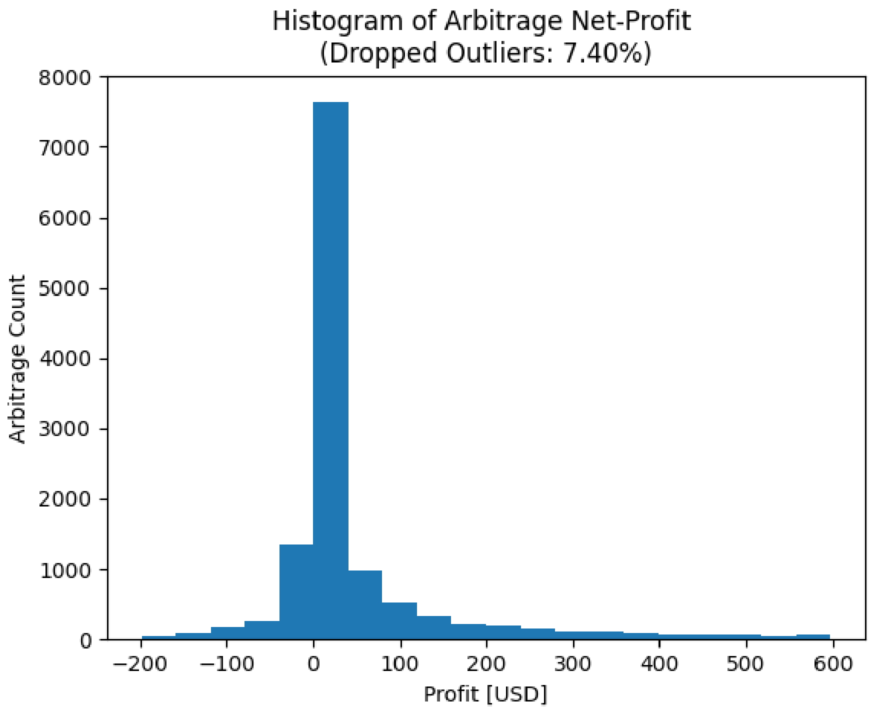
Similarly, the following graph shows the top token use, in terms of arbitrage count:



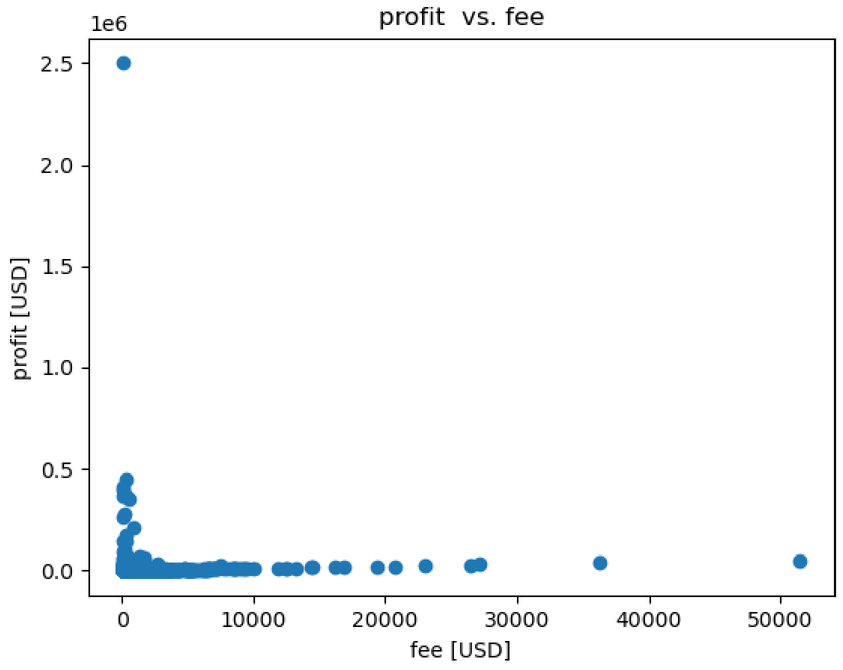
The most used token is WETH - Wrapped Ether - which is a wrapped version of the Ether currency as a Fungible Token. Then we can see 4 stable-coins matched to the USD: USDC,OHM,DAI and USDT.

5.6. Profits and Fees

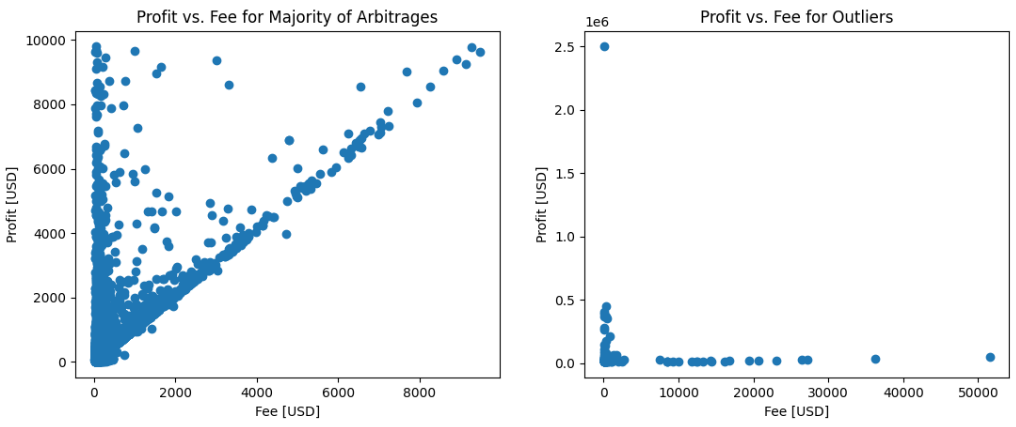
We are interested in the profit made from arbitrages. We distinguish between profit, and net profit. When we refer to the profit, it is the difference in all tokens amount before and after the arbitrage, translated into USD. The net profit is the profit minus the fee, so it might be negative as well. The following graph shows the net profit distribution:



We can see that most of the net profit is positive, and smaller than 50 USD. Similarly, most of the loss (negative profit) is smaller than 50 USD. that are 12,758 profiting arbitrages vs 832 losing arbitrages.
Another interesting parameter is the correlation between fee and profit. Is it likely that higher paid fees are exploited into bigger profit?
The following graph shows all the arbitrages in the fee-profit plane:



It is obvious that in this graph, with all the data points, some outliers hide the common-case date that we are more interested in. The following graph filters the outliers:



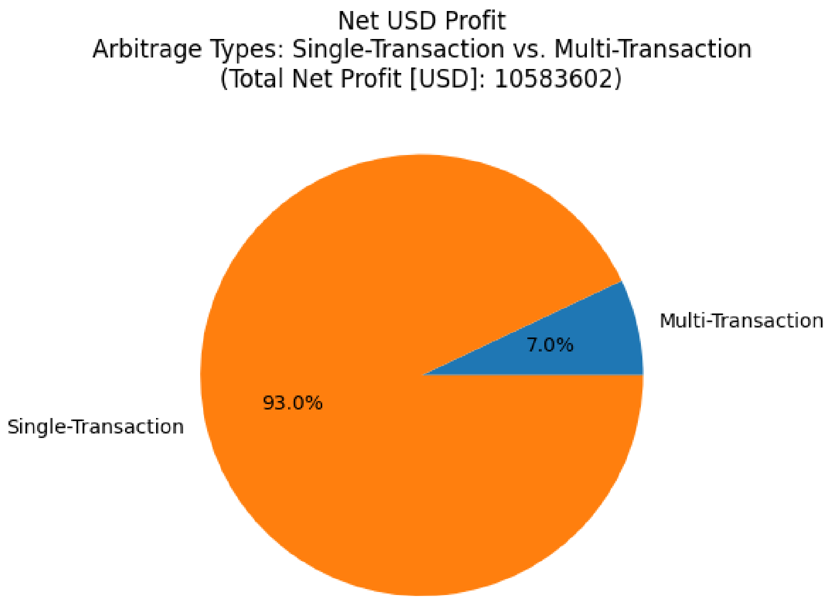
We can see that, indeed, most of the data-points are close to two linear lines. One linear line is the identity between fee and profit, meaning profit margins are very small. Another line describes more profiting arbitrages and shows that a little fee can translate to big profit. It is interesting to see that there are not many data-points away from these two lines.

In total, 1237.97 Ether was paid in fees for arbitrages, which translates to 3.6 Million USD. Total profit is 14.2 Million USD, but total net-profit from arbitrages is 10.58 Million USD. Out of the 13,590 arbitrages we found, 12,758 are profitable, and 832 aren't.

The arbitrage[19] with the largest fee paid 51,530 USD in fees, and it happens to be the most non-profitable transaction with a net-loss of 1,492 USD. In terms of most profiting arbitrages, the top 3 arbitrages profited: 2.5 Million USD[20], 448,917 USD[21] and 408,750 USD[22]. They are all single-transaction arbitrages. In Appendix A you can find a list of the most profitable arbitrages.

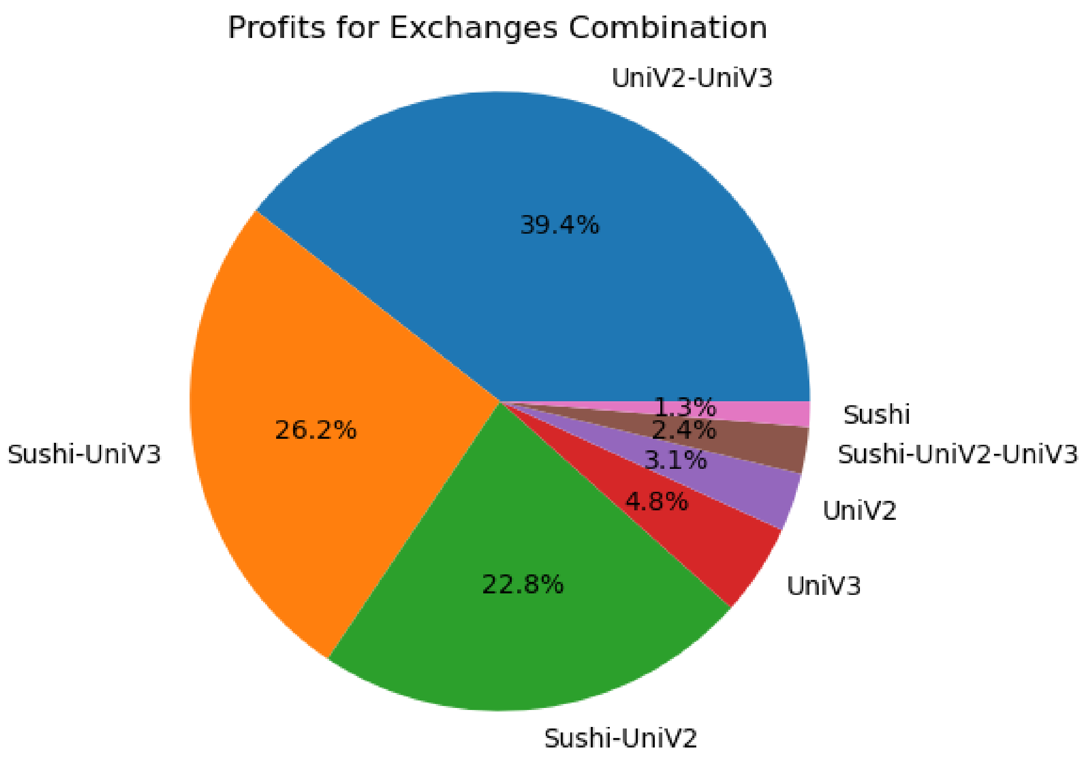
[19] https://etherscan.io/tx/0x355f1722adb904e4cbf2e1c14466f5fffa5d1038eb82ce6fa8a1b17d1da4b368
[20] https://etherscan.io/tx/0xc21a480140008dfa0af0864fbde9a69825b37effbc53fab954224e903fa9999e
[21] https://etherscan.io/tx/0xd0c5a4cc95f537e4a92489cdc2770bacf062ae4ab7358acadeedfca6a1680c1a
[22] https://etherscan.io/tx/0xfd1aea29aa64ca964bec64cda7b4fe313b69d5a1aa8ea97f200e886db72d4288

Looking at the classification of Single-Transaction and Multi-Transaction, we can see the division of profits:



Single-transaction arbitrages are twice as common as multi-transaction, and they are more dominant in terms of profit too, implying that single-transaction arbitrages tend to be more profitable than multi-transaction arbitrages.

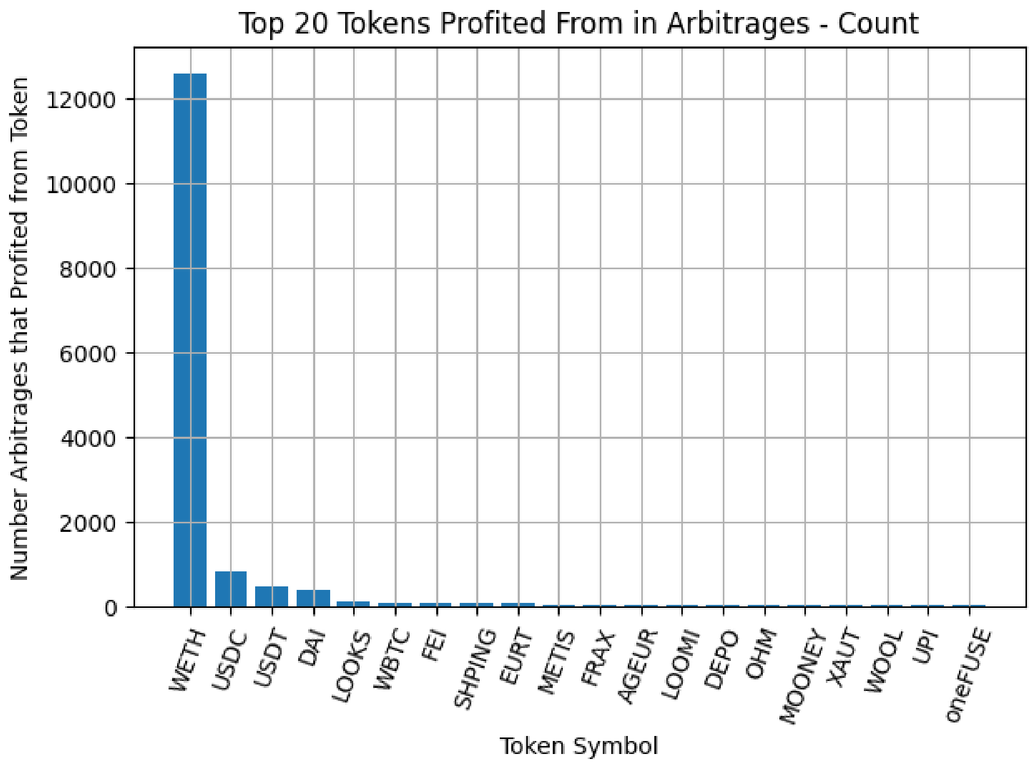
Looking at the exchange combination, we can see how the net-profit is divided among different combinations:



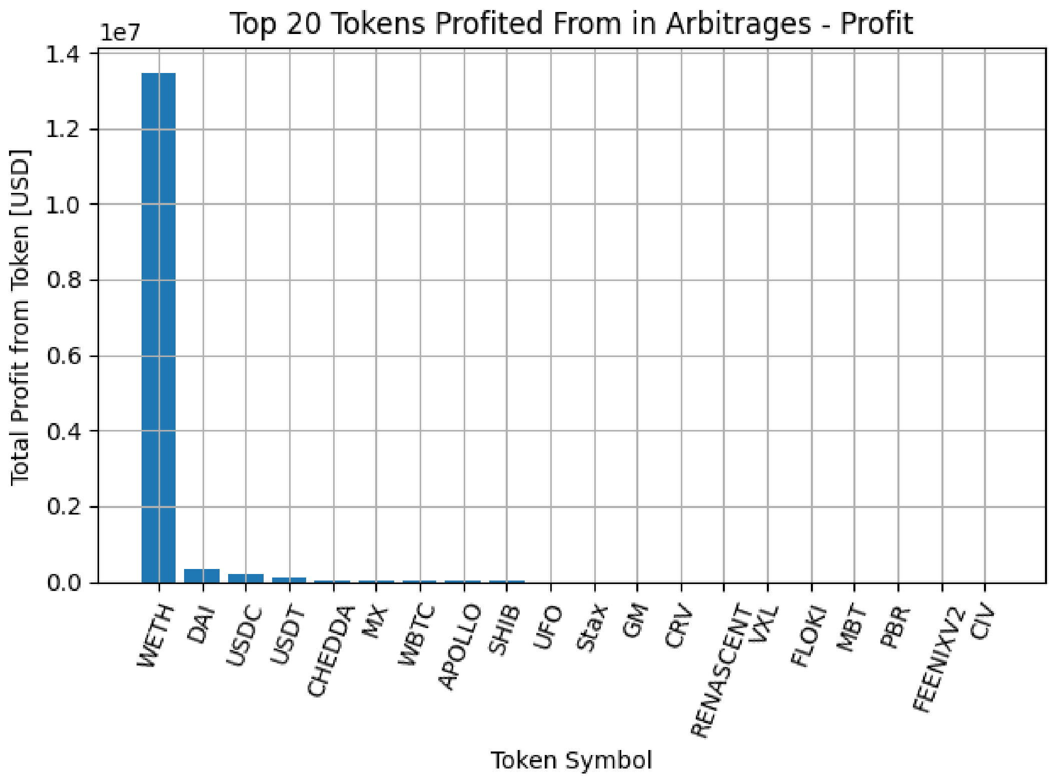
We can see that in terms of total profit, the UniswapV2-UniswapV3 combination is leading; the three pairs combinations are the three leading combinations; and the least used combination - the one with all the three exchanges is also with the smallest total profit.

Regarding the profits per-token, as we defined in section 3 there are precise and imprecise arbitrages. Out of the 13,590 arbitrages, 10,374 are precise and the rest aren’t. If we look at the main token as the main profit and all the rest in the same arbitrage as collateral, then the collateral profit is 0.1% of the total profit.

Looking at tokens that were profited from in arbitrage, we can look at the number of arbitrages that used each token and how much profit was made from each type. The following graph describes the top tokens that were profited from in arbitrage:



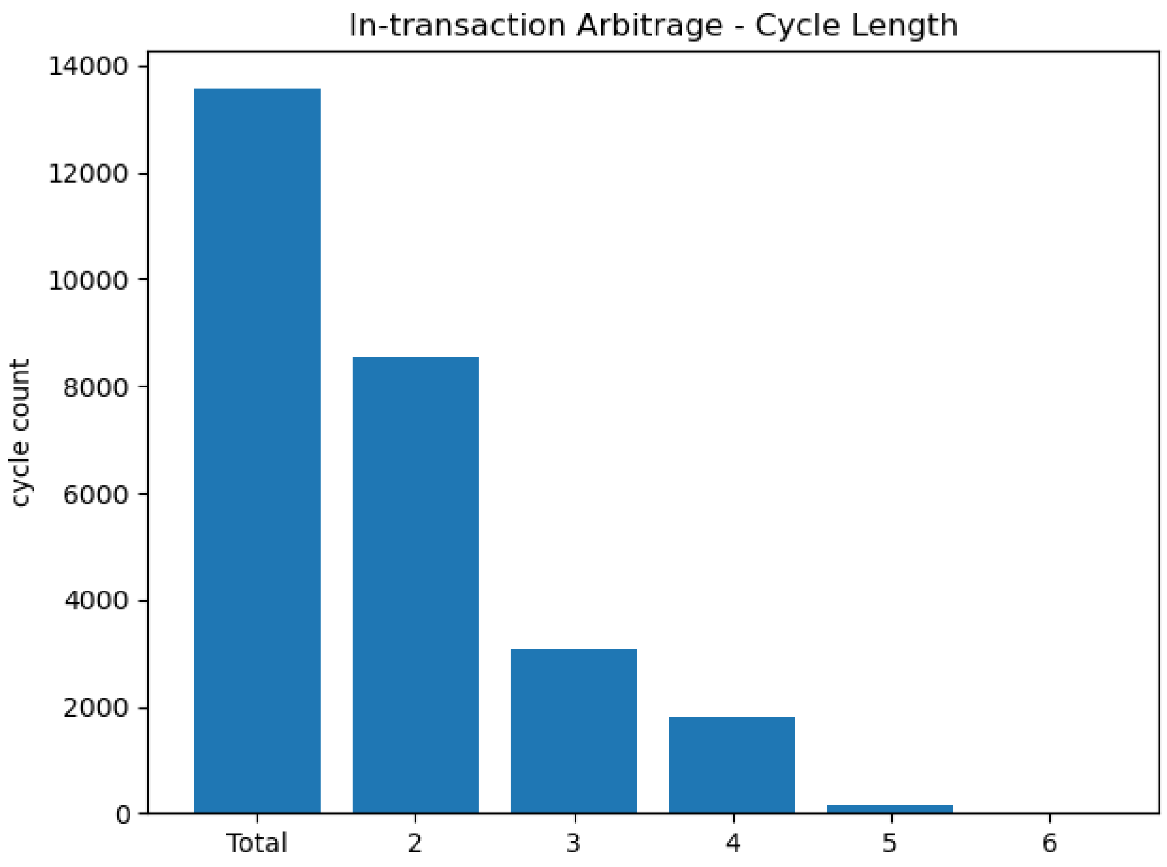
We can see that WETH is the most profited from, followed by the usual stable-coins. Looking at the profit per-token, the following graph describes it:



As expected, the largest profit is in WETH (13.4 Million USD), followed by DAI,USDC and USDT with 326,497, 217,006 and 102,162 respectively.

5.7. Cycle Length

An arbitrage is a cycle of two or more swaps. The following graph shows the distribution of the cycle's length:



We can see that the leading cycle length is two swaps. The maximum length is 6 swaps, but it is very scarce. An example with 6 swaps can be seen in this transaction[23].

5.8. Multi-Transaction Placement in Block

As explained before, a multi-transaction arbitrage can only be created by a miner since it requires careful placement of the 3 transactions. If we look at the index of transactions in the block, then about half (1,999/4,414) multi-transaction arbitrages start as the first transactions of the block, and the rest are also concentrated at the start.

6. Conclusion

In this project we gathered and analyzed the data of 30,000 blocks. We learned that arbitrages are a common practice which is sometimes very profitable. We distinguished between different types of arbitrages: single-transaction vs. multiple-transaction arbitrages, and between exchange combinations. It turned out that the single-transaction arbitrages are more profitable than the multi-transaction arbitrages on average. We also learned about common properties of the arbitrage: Ether and the stablecoins are the most popular tokens used for arbitrage; the leading cycle length is short - 2 or 3; and the common exchange combinations are those with two exchanges. There are open questions worth exploring, such as: What other transactions do the arbitrageurs and what do

[23] https://etherscan.io/tx/0x8dc19c697e4e5311825f9fe903114d7ed148fb92b3d07415800d87df8532bc492

they do with their crypto capital? What makes an arbitrageur a successful one? Can we find common properties of very profiting arbitrages?

Appendix A - Most Profitable Arbitrages

Profit [USD]	Transaction(s) Hash
2,499,102	0xc21a480140008dfa0af0864fbde9a69825b37effbc53fab954224e903fa9999e
448,917	0xd0c5a4cc95f537e4a92489cdc2770bacf062ae4ab7358acadeedfca6a1680c1a
408,750	0xfd1aea29aa64ca964bec64cda7b4fe313b69d5a1aa8ea97f200e886db72d4288
394,335	0x89c23bbb805958e8b3b4c23970a2ff5a964b85a791669f2173dace197eb895cc
368,951	0xe4298bf375fa640d8dea5a3dcab44b6dd1d756ce8e4bbc8e51b8763595ff77c9
368,499	0x0c34704bc167a97dced6f0c1e042e0aa7aff1ac6ea6544bdf0f7b7c01e382a7e
353,945	0xaf6961f55e5a9d4391881c50b870ca0791322096792d934f854e3d15218e246f
279,195	0x9180ec5330361b1ead3926392cab32b8ddfd8fcb070c6fbc8ba325c84643fbb6
261,475	0x0e14fd8885eb1f376341dabf6b6aa18a2e8ff8b1fd4e3d45e1ad503f9b7c2c79
210,757	0xd80aa5ab20552fdf1125f005604b1218d43666de2cc6d24e2973de683bf26da6
174,833	0x373c0cc6ec327a614c2e504d1882f24ecaf7ff346fbcb57f7bfdd6546223bcad
146,154	0xcae1f0e4e47872fcf98993b976ad61dc924054e12d8447d6ad0bc6478af95789
140,897	0x16eec4997fc89e8e75df66e5e4b394b0dcbe7b3dc1a4d084a9055a3f2963cdbc
108,764	0x982cbceba7b0f13e589819a5b82397a94c94b156be39164c78dd6ba30d5ea79d
89,888	0xfeee19313a538d02d06557adfd23d7e860a05568ef31e3ef18a0760050f603d3
78,416	0xbe8837ec7e2b7cb49f02d55900938cd79f6a47e5697e066881e4ed7e6401070f
67,222	0x7fa4d9007ea91cdd35bd3e529ab4516fea7e3e144886677464fc9fee700a68c9
66,010	0xb2a5ddc7bb4b27b1008cc649ccc5d74e34b41093860ffc4077c84ce3dcb8efc0
65,068	0x8a9dd7f1f89c411cc4f5f349f62280f520e8cc1d907e158c6e46011f89517a0d
64,701	0x6cab2621c543918d2c61683d08755f491d72750653c71bd1a650d120d72e6e6c
60,354	0x7f9c6aabc1cde3aa57ba48f1616072b9e36972e7095946eec20fae273fc0f768
59,847	0x20414aaf35a91d05146a116bfe2a36aeccd4f6d021a72d27ea7ea1b8192dcfc9
56,027	0xba5baebb9f7023567ec89baf9a10f22d2e5f03c513fae5598e6ab8a1cd2ec1d6
53,923	0x01f1027b6f62f378eb3ad5add1f7109ff9c8b0f5bbe191ef9732acac78d15213 - 0xb09283df66f5c7b68450cced95f2954932f5fb594227a1afe7fc66b886267154

Appendix B - Used Code

The code we used for extracting data, finding swaps and creating all graphs is available at the GitHub repository: https://github.com/ron-marcus/crypto_arbitrage