Arbitrage in Crypto Markets: An Analysis of Primary Ethereum Blockchain Data

Magnus Hansson*

Department of Economics at University of Gothenburg

hansson.carl.magnus@gmail.com

Link to latest version

November 9, 2022

Abstract

The Ethereum blockchain is a decentralized computing platform providing peer-to-peer financial services. Decentralized exchanges, which run on the blockchain, enable matching of buyers and sellers without any central third party, and are distinct from the centralized "off-chain" cryptocurrency markets often studied in the literature. The decentralized markets facilitate trade in cryptocurrencies and other digital assets and have daily turnovers of several billion dollars. In this paper, I study how arbitrageurs on the blockchain contribute to price discovery and price efficiency in decentralized "on-chain" markets. I collect a transaction-level dataset of primary data from the Ethereum blockchain and cleanly identify a set of completed cross-exchange and triangular arbitrages. To investigate the speed at which arbitrage opportunities are eliminated, I study how sensitive arbitrage profits are to when the trades execute. I show that most arbitrage profits are made immediately after the occurrence of price anomalies, indicating that decentralized markets adjust fast after a shock to the no-arbitrage price.

JEL Classification: C55, C57, E42, G12, G14, G23

Keywords: Arbitrage, Price efficiency, Decentralized exchanges, Ethereum, Blockchain, Decentralized finance, Automated market maker, Cryptocurrency

*I thank Erik Hjalmarsson and Andreas Dzemski for valuable comments and suggestions. I am grateful for advice related to the data processing from Thomas Jay Rush at TrueBlocks. I also thank seminar participants at the University of Gothenburg.

1 Introduction

The Ethereum blockchain is a decentralized computing platform, and one of its main use cases is peer-to-peer financial services. *Decentralized exchanges* run on Ethereum and accommodate trading of digital assets, including derivatives of bitcoin and ether (Ethereum's native currency), and cryptocurrencies pegged to fiat currencies (stablecoins). The aggregated daily trading volume of these exchanges is several billion dollars, and trading is settled on the blockchain without intermediaries. As it is expensive to keep distributed order books, decentralized exchanges use algorithms to determine prices. Conditional on a trade, arbitrageurs can precisely predict changes to these prices due to the transparency of the blockchain. These arbitrageurs help in the price discovery process by trading away price imbalances between exchanges (cross-exchange arbitrage) and between relative exchange rates in multiple currencies (triangular arbitrage).

My paper answers the question: How do arbitrageurs on the blockchain contribute to price efficiency in decentralized markets? Arbitrageurs on the blockchain have been shown to act similar to high frequency traders in traditional markets (Daian et al., 2019), however it is unclear to what extent they contribute to the price discovery process. To investigate this, I collect a transaction-level dataset of primary data from the Ethereum blockchain, and track transactions individually. As Ethereum data are fully transparent, I am able to identify a clean set of pure arbitrage trades, and establish how prior shocks to exchange rates predict arbitrages. To investigate the speed at which arbitrage opportunities are eliminated, I study how sensitive arbitrage profits are to when the trades execute. I show that most arbitrage profits are made almost immediately after the occurrence of price anomalies, indicating that decentralized markets adjust fast after a shock to the no-arbitrage price.

This paper introduces primary decentralized exchange data from the Ethereum blockchain to the financial literature. The main source of the data is Ethereum's largest exchange *Uniswap*. Every trade that is executed on decentralized exchanges are stored on the blockchain. The transaction data contain detailed information about the trades, including what currencies and amounts are traded, and who made the trades. Multiple trades can be executed within one *atomic* Ethereum transaction. Arbitrageurs are expected to use this functionality and execute all trades related to an arbitrage as a bundle by coding them into one Ethereum transaction. This makes arbitrage on Ethereum as close to risk free as possible, with the caveat that transaction fees still need to be paid for failed arbitrages. Due to the unique transparency of the blockchain data, I am able to classify bundles of trades as arbitrages by identifying two or more trades that form a closed loop,

such that the output amount and currency of one trade is equal to the input amount and currency of the next trade. For each arbitrage, metadata such as the number of trades, costs, and profits are calculated. The transparency of the decentralized exchanges also provide data reliability as no manipulation of the data is possible besides actual trading.[1] Previous economics literature studying cryptocurrency arbitrage have observed price differences on traditional trading exchanges operating outside of the blockchain ecosystem, also called *centralized exchanges*. This literature suggests that there have been large inefficiencies in the cross-border cryptocurrency markets (Makarov and Schoar, 2020), but that the price deviations have become less pronounced since 2018 (Shynkevich, 2021). My paper extends these studies by analyzing price efficiency on decentralized exchanges operating on the blockchain itself and by studying realized arbitrages. A related but distinctly different literature in computer science has studied cryptocurrency arbitrage with primary blockchain data. Wang et al. (2021b) give an overview of triangular arbitrages on Uniswap, and most related to my paper is Berg et al. (2022), who show that triangular arbitrages are more common in times of high market volatility and that many trades execute at unfavourable prices.

I find that arbitrages are very sensitive to the order of execution and that arbitrage windows close fast. Transactions on the Ethereum blockchain are executed sequentially in batches called *blocks*. These blocks hold on average 200 Ethereum transactions and are executed on average every 14 seconds. Arbitrageurs eliminate price anomalies fast, often as soon as they arise and often within the blocks. Exchange rate changes from trading immediately prior to the arbitrage are the strongest predictor of arbitrage profits. This implies that prices at the end of the blocks are likely to reflect market prices. Decentralized exchanges use the end-of-block prices as starting prices in the next block (Adams, Zinsmeister, and Robinson, 2020), and traders on decentralized exchanges use these to initiate their orders. Furthermore, as the blockchain is a closed system, not easily connected to the "outside world", the stored end-of-block prices are used by other decentralized finance applications as on-chain reference prices. My findings indicate that these reference prices observed on decentralized exchanges and used by traders on the blockchain are likely to be arbitrage-free. The automated arbitrage activity leads to improved price efficiency on

[1] Alexander and Dakos (2020) show that many empirical published papers studying cryptocurrency use faulty data. A study filed to the U.S. Securities and Exchange Commission by Bitwise Asset Management (SEC, 2019) found that 95% of all reported trading volume on *off-chain* cryptocurrency exchanges is non-economic "wash" trading. Similarly, Cong et al. (2019) show that many unregulated cryptocurrency exchanges engage in significant wash trading that affects exchange ranking, cryptocurrency exchange rates, and volatility. Victor and Weintraud (2021) find that 30% of all traded tokens on the order book exchanges EtherDelta and IDXE have been subject to wash trading.

the blockchain, similar to how algorithmic trading observed in traditional markets reduces the frequency of arbitrage (Chaboud et al., 2014).

To study the sensitivity of arbitrage timing, I design a counterfactual simulation where bundles of arbitrage trades are re-executed with different timing compared to their original counterparts. By creating an alternative order of the full transaction history, I can evaluate if arbitrage transactions would have been profitable in different states of the world, and accordingly measure how sensitive arbitrages are to timing. By re-executing arbitrage transactions prior to when they initially occurred, I can observe how far back arbitrage transactions are profitable and thus for how long the arbitrage opportunities existed. This gives an indication to how fast (efficient) the market is to correct price imbalances. The simulation is made possible because Ethereum transactions are programmatically defined and the transactions' code can be observed on the blockchain. Therefore, it is possible to re-execute transactions with changed parameters ex-post and thus simulate an alternative reality. This is a unique feature of transparent blockchains, like Ethereum, and used in this study to form a detailed view about the price efficiency on the decentralized exchanges.

The counterfactual simulation shows that arbitrages are very time sensitive. 69% of the realized arbitrage transactions are no longer profitable if they are re-executed as the first transaction in their own block. As it is more expensive to execute transactions early in the block, this number is adjusted to 15% if the transaction costs are also adjusted to what they would have been in the first position. In these cases, the arbitrage opportunities occur within the blocks and the arbitrage transactions capitalizing on the opportunities shortly afterwards. Practically, this means that arbitrageurs monitor pending transactions and place their own trades to immediately neutralize price imbalances. If prices are observed on a block-level instead of on a transaction-level, these price anomalies are never seen. This means that the majority of the arbitrage profits are made within a window of 14 seconds. As traders on decentralized exchanges observe prices from the previous block, they are therefore likely to observe and trade based on arbitrage-free prices. The analysis thus indicates that most of the price discrepancies are arbitraged away with high efficiency within the blocks, leaving the end-of-block price close to the "equilibrium" price. When re-executing the arbitrage transactions at the beginning of previous blocks, only 10% of the arbitrage opportunities exist 5 blocks back (approximately 1 minute in calendar time).

To empirically investigates how far back exchange rate changes affect arbitrages, I conduct a predictive study in which price imbalances are used to estimate whether or not an arbitrage transaction is likely to occur. Arbitrages are solely created by trading that off-sets the no-arbitrage price. Large changes to the exchange rate should therefore signal future arbitrage transactions. To discriminate arbitrage transactions from regular

trading, a random control group is constructed, consisting of randomly sampled non-arbitrage transactions. As the decentralized exchange data are fully transparent on a transaction-level basis, the exact price impacts of prior trading to these transactions are calculated. The predictors are defined by changes to the exchange rates caused by trading up to 10 blocks (approximately 2 minutes) prior to the studied transactions.

The predictive exercise shows that price changes from prior trading in the same block and trading up to 4 blocks back significantly predict if a transaction is a bundle of arbitrage trades or not. Thus, trading up to one minute prior to a transaction helps to predict whether an arbitrage transaction will occur. These results hold for the full sample, as well as for 5 month subsamples constructed to pick up dynamics in the data.

To evaluate how prior trading trigger arbitrages, arbitrage net profits are regressed on exchange rate changes caused by the trading up to 10 blocks prior to the arbitrage transactions. The analysis indicates to what degree arbitrageurs are able to profit from prior imbalances in the exchange rate and at what speed these opportunities are arbitraged away. This speaks to the efficiency of the decentralized markets, and to what extent these markets are able to track the no-arbitrage price.

When regressing realized arbitrage net profits on previous exchange rate changes, within-block trading is far more important for arbitrage profits than price changes from previous blocks. When using the full sample, trading in the same block as the arbitrage transactions and trading in the previous block are both significantly affecting net profits. However, price changes from within the block have a significantly stronger effect. Arbitrage windows therefore tend to close, on average, in 14 seconds. Furthermore, when the analysis is run on 5-month subsamples, the results are stronger for the later part of the sample. Between May 2021 to September 2021 and October 2021 to February 2022, only price changes from within the arbitrage block affect arbitrage net profits. The markets are more efficient and prices adjust faster to the no-arbitrage price after a shock, possibly explained by increased arbitrage competition in the later part of the sample. These results are consistent with the observations from the counterfactual simulation, and again indicate that end-of-block reference prices are likely to be arbitrage-free.

The remainder of the paper is organized as follows: Section 2 gives a background to the Ethereum ecosystem, its native currency ether, and decentralized exchanges operating on the blockchain; Section 3 describes the data collection and arbitrage classification strategy; Section 4 outline the empirical analysis and the results; and Section 5 presents concluding remarks.

2 The Ethereum Blockchain and Decentralized Exchanges

Section 2 provides the necessary background for the rest of the paper. Section 2.1 covers the Ethereum blockchain and its native currency ether, Section 2.2 introduces decentralized exchanges, and Section 2.3 demonstrates how arbitrage is conducted on decentralized exchanges. Readers familiar with the Ethereum blockchain and decentralized exchanges can comfortably skip to Section 2.3.

2.1 Ether and Ethereum

Bitcoin is a distributed ledger allowing users to transact the cryptocurrency bitcoin without any third party, and records the transactions in its *blockchain* database. Harvey et al. (2022) categorize the Ethereum blockchain, together with its main competitors (Solana, Avalanche, Cardano, and Algorand), differently than Bitcoin, as Ethereum extends Bitcoin's distributed ledger to a universal decentralized computing system. Ethereum (Buterin, 2013; Wood, 2014), building on Dwork and Naor (1992), Back (2002), and Nakamoto (2008) amongst others, is in essence a network of computer nodes sharing the same database and global state. The global state is a large data structure that describes the state of the world at a specific time, containing, for instance, account balances for all Ethereum users. Formally, Ethereum is a so-called peer-to-peer replicated state machine capable of executing user transactions without a central agency.[2] The network nodes stay in sync with each other as the global state is updated discretely by an execution model called the Ethereum Virtual Machine.[3] Nodes are operated by network participants that run an Ethereum client software that follows the rules of the Ethereum protocol described in Wood (2014).

Figure 1 gives a general overview of the route of Ethereum transactions. Users create new transactions and send them to the Ethereum network for validation. The transactions can either be sent privately or publicly. *Miners* batch the transactions into *blocks* and spend computing resources to guarantee that the transactions are valid.[4] Once the block is validated, it is linked to the previous blocks of transactions in a *blockchain*, and the global state is updated. The rest of Section 2.1 explains each part of Figure 1 in more detail, and gives an institutional overview of the Ethereum blockchain.

[2]For an overview of cryptocurrencies and decentralized finance see Härdle, Harvey, and Reule (2020), Makarov and Schoar (2022), and Harvey et al. (2022).

[3]Not to be confused with the *Ethereum Virtual Mavericks*.

[4]On the 15th of September 2022, Ethereum changed the consensus mechanism from proof-of-work to proof-of-stake in a software upgrade called the *Merge*. In the current version of this paper only transactions from before the merge are analyzed and the term *miner* is used for this distinction and for simplicity.

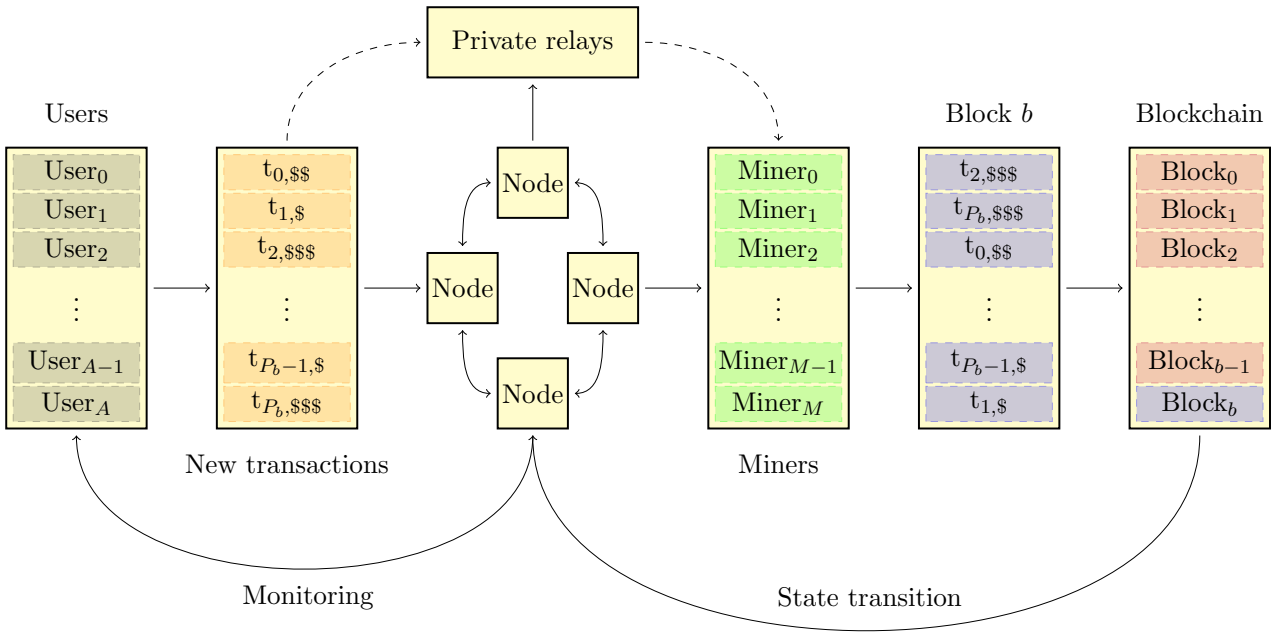


Figure 1: The Ethereum blockchain.

2.1.1 Ethereum transactions

In traditional markets, transactions are *completed* agreements between a buyer and a seller for some asset. In the blockchain ecosystem the term transaction is defined more broadly, and refers to a computer instruction that is not completed until it is validated by the network. A financial "transaction" on a blockchain is therefore more similar to an *order* in a traditional market.

Ethereum transactions are computer code that defines a set of instructions for how the Ethereum Virtual Machine should update the global state. In the Ethereum ecosystem transactions are the sole way users interact with the Ethereum network, and thus with each other. These transactions are cryptographically signed by their users and represent one or several of the following actions: Send cryptocurrency to another user, create a *smart contract* (Szabo, 1997), and interact with an already existing smart contract. One Ethereum transaction can hold several instructions at once, for example interact with multiple smart contracts within a single transaction.

A regular transaction, transacting currency from one user to another, functions in a similar way to transactions in traditional banks or on the Bitcoin network, where one user can send currency to another. A *smart contract creating transaction* is an Ethereum transaction with embedded computer code that defines a software application. By sending this code to the Ethereum network, it is uploaded and stored in the blockchain's "database", and is accessible to every user. All *decentralized applications* on Ethereum are smart

contracts. These include *decentralized finance* (DeFi) applications such as decentralized exchanges (DEXes), which facilitate trading of digital assets on the blockchain. The last type of Ethereum transactions are interactions with existing smart contracts. For example, when users trade on decentralized exchanges they send transactions to smart contracts with instructions on how to execute the trades.

Users pay a fee, denominated in Ethereum's native currency ether, to the network to execute their transactions. The size of the fee depends on the complexity of the transaction. Ethereum transactions can take up different amounts of computational work in terms of processing, memory, and network usage. This computational work is called *gas*. The ex post transaction fee is equal to the amount of gas used times the price for each gas unit denominated in ether.[5]

2.1.2 Atomicity

Ethereum transactions are *atomic*: Either every operation in the transaction is successfully executed or the whole transaction is cancelled. There are three ways an Ethereum transaction can fail: The transaction fee is set too low, a condition in the transaction code is not met, or the transaction is conflicting with the Ethereum state (Zhou et al., 2021a). Users can, therefore, condition transactions such that they only execute if certain conditions are met.

2.1.3 Transaction validation and blocks

Once a transaction is created and cryptographically signed by its user, it is not executed right away. The transaction is sent to a node that shares it with other nodes, and thus throughout the network. The transaction is now public and placed in a queue called the *mempool* (memory pool) to be executed. Special nodes in the network called *miners* compete with each other using processing power to validate transactions in exchange for transactions fees. During the sample period covered by the current analysis, the validation is done by a process called *proof-of-work*. Miners monitor the mempool and batch pending transactions into *blocks*. Miners are incentivized to include transactions that pay the highest transaction fees, and sort these transactions into the block based on their fees. This means that users can pay more to get their transactions executed faster (Zhou et al., 2021b). A block is considered full of transactions when the computational threshold, denominated in gas, is met. Thus, the number of transactions in a block varies depending on how complex they are. When a miner has selected a set of transactions,

[5]The Ethereum Virtual Machine is a quasi-Turing-complete machine as gas costs bound the total amount of computation.

representing a full block, it tries to validate the block as fast as possible. Different miners can select different transactions from the mempool to try to validate. Transactions that pay a below-market transaction fee might be held in the mempool until network demand goes down and they become profitable for miners to validate. There is no guarantee that pending transactions in the mempool that pay low gas fees will ever be executed.

A new block is validated on average every 14 seconds, but block times over 1 minute are not uncommon. The expected block time is a constant and the average amount of computing power needed to mine a block is dynamically adjusted to match this. Approximately 15 transactions are processed per second, although the variation is high.[6] When a block is mined at a mining node, it is sent out to other nodes to be checked and shared. The nodes verify that the miner's proof is correct, execute the transactions in the new block, and share the block to the rest of the network. When the majority of the nodes in the network have done so the new global state is agreed upon. The blockchain is immutable and guarantees that the transaction history cannot be changed, as any historical alteration changes the entirety of the blockchain. Every historic transaction on the blockchain must be public and accessible, such that the current state can always be verified.

It is important to note that the Ethereum global state is updated sequentially for each transaction, and each transaction operates independently on the current state. Therefore the order of transactions in a block matters.[7] The financial markets on Ethereum are combinations of continuous and discrete markets. The mined transactions are executed discretely, sequentially and independently. However, which transactions are executed and in what order, is subject to the mining process. This process is economically equivalent to a continuous time auction that finishes when a block is mined, and a new auction begins (Daian et al., 2019).

2.1.4 Front-running

Since Ethereum transactions are executed sequentially and operate on the Ethereum state independently, the transaction ordering in a block is of high importance. This leads to rent-seeking behavior where users try to front-run lucrative transactions. On Ethereum all information is public and front-running is the practice of acting fast on complex public information, similar to how high frequency traders operate in traditional markets.[8] Most front-running activity on Ethereum takes place on decentralized exchanges (Torres, Camino, and State, 2021). As miners sort transactions depending on their transaction

[6] Researchers are working on scalability of blockchain systems as there is a trade-off between transaction throughput and decentralized security.

[7] For a technical overview see Appendix A.

[8] For an overview of front-running on blockchains see Eskandari, Moosavi, and Clark (2020).

costs, users can post transactions with higher transaction fees with the hope of getting their transactions executed before others' and capture potential profits. Daian et al. (2019) show how transaction cost auctions take place to front-run profitable opportunities and capture what they call *miner extractable value* (MEV).[9][10] Miner extractable value is the profits a miner would be able to extract from a block by re-ordering and inserting transactions. This includes transactions fees, but also non-standard extraction methods such as arbitrages. The total amount of present miner extractable value is unknown, although estimations can be made ex post. Qin, Zhou, and Gervais (2021) estimate that front-running profits of $541M USD were extracted under their sample period of 32 months.

2.1.5 Private transactions

Miners have the ability to choose which transactions to batch into blocks and attempt to validate. A transaction can be included in a block as long as it is signed by its user, regardless if the miner retrieved the transaction from the public mempool or in any other way. A feature of the Ethereum ecosystem is the ability to relay transactions privately to miners. This provides pre-trade privacy by bypassing the public mempool. At the time of writing, 86% of Ethereum's mining capacity is supplied by miners using private relay functionality that accepts private transactions.[11] Private relays allow the option to send transaction bundles directly to miners. Traders use this to obfuscate transactions to mitigate font-running and miners accept these transactions provided that they pay sufficient gas fees.

A transaction bundle is one or several Ethereum transactions that the miner guarantees will be executed in sequence. This is similar to the atomic functionality of the single Ethereum transaction, but extended over several transactions and not guaranteed on a protocol level. The private relay guarantees that either all transactions are executed or none at all. Private transaction bundles can target a specific block for which it is valid. Furthermore, as all nodes in the network need to execute and verify each transaction in a new block to update the global state, private transactions become public as soon as they are included in a verified block.

[9] Miner extractable value, is more often called *maximal extractable value*, and sometimes *blockchain extractable value* since not only miners participate in capturing MEV.

[10] This phenomenon has attracted attention from economists, one example is researchers at the Bank for International Settlements writing that "Miner extractable value is an intrinsic shortcoming of pseudo-anonymous blockchains. Addressing this form of market manipulation may call for new regulatory approaches to this new class of intermediaries." (Bank for International Settlements, 2022), referring to block validators as intermediaries.

[11] https://docs.flashbots.net/Flashbots-auction/searchers/faq/

2.2 Decentralized exchanges

Market makers, in traditional limit order book exchanges, connect buyers and sellers to accommodate trading, and use centralized technology to keep track of the order book. In contrast, decentralized exchanges are smart contracts on the blockchain, operating as autonomous and non-custodial market places without an order book. These exchanges have gained traction and is one of the fastest growing sectors within decentralized finance (Makarov and Schoar, 2022). Users send transactions with trading instructions to the exchange, trading is accomplished peer-to-peer and the transactions are settled on the blockchain without intermediaries.

A decentralized exchange consists of three parts: Liquidity providers that deposit liquidity to the exchange in return for trading fees, liquidity takers that trade one asset for another, and a price-discovery mechanism (Zhou et al., 2021b). The core difference distinguishing decentralized exchanges from centralized exchanges is the market making mechanism. It is expensive to keep a decentralized order book. Therefore, decentralized exchanges use market making algorithms to facilitate trading (Angeris et al., 2021).[12]

2.2.1 Automated market makers

Automated market makers (Savage, 1971; Hanson, 2003; Berg and Proebsting, 2009) are a set of algorithmic markets using scoring rules for market and decision making, and resemble general models of non-cooperative trading equilibriums, such as Shapley and Shubik (1977). Uniswap (Zhang, Chen, and Park, 2018; Adams, Zinsmeister, and Robinson, 2020; Adams et al., 2021) is the largest decentralized exchange and exists as a set of smart contracts uploaded to Ethereum. Uniswap is an automated market maker that uses a *constant product formula* to decide the exchange rate between two currencies, and has been formally shown to track a reference market price (Angeris et al., 2021).

Consider two assets X and Y, and their exchange pair X/Y. The exchange rate is calculated such that the product of the respective liquidity pools is kept constant,

$$k = x \cdot y. \tag{1}$$

Here k is called the *invariant* and represents the depth of the market, and x and y are the amounts of asset X and asset Y deposited by liquidity providers into the liquidity pools.[13]

[12] In addition to centralized and decentralized exchanges, *hybrid* exchanges exist. Hybrid exchanges keep a centralized order book off-chain, but settle trading on-chain.

[13] The assets in the liquidity pools are fairly constant, as liquidity providers do not frequently move their assets across pools (Heimbach, Wang, and Wattenhofer, 2021).

The value of k is initially determined by the liquidity added to the pools when the trading pair was initiated. The invariant, k, changes in 3 ways: Liquidity providers add or remove liquidity, trading fees after each trade are added to the pools, or by "donations". Any added liquidity to the liquidity pools such that the ratio of x and y is not kept constant is considered a donation and changes the value of k disproportionately.

Contrary to buyers and sellers on limit order book exchanges, traders on constant product markets do not provide a price for one asset in terms of another. Instead, liquidity providers deposit liquidity to both assets' liquidity pools, and it is up to the liquidity takers, i.e., the traders, to decide which assets to trade given the current exchange rate. If a liquidity taker wants to swap δ_y of asset Y for δ_x of asset X, the liquidity pools are altered such that,

$$k = (x - \delta_x) \cdot (y + \delta_y) \tag{2}$$

and they have to pay,

$$\delta_y = \frac{k}{x - \delta_x} - y \tag{3}$$

of asset Y at the exchange rate $\frac{\delta_x}{\delta_y}$, as the invariant needs to stay constant.[14][15] The exchange rate converges to the marginal exchange rate as the liquidity pools get sufficiently large,

$$\frac{\delta_x}{\delta_y} = \frac{\delta_x}{\frac{k}{x-\delta_x} - y} = \frac{x}{y} - \frac{\delta_x}{y} \to \frac{x}{y} \quad \text{as} \quad \delta_x \to 0. \tag{4}$$

The difference between the exchange rate and the marginal exchange rate is the *price slippage*. If the amount traded is significant in relation to the amount in the liquidity pools, price slippage occurs, and this is how arbitrage is created. Cartea, Drissi, and Monga (2022) show that it is sub-optimal to execute large orders in one trade as it leads to price slippage and unbalancing of the exchange rate.

As a numerical example of how trading affects the exchange rate, consider the exchange pair X/Y, and the liquidity pools $x = 1,000$ and $y = 90$. Here the invariant, k, is equal to

[14] On Uniswap there is also a transaction fee of 0.3% to incentivize liquidity provision that is omitted in Equations 2 and 6.

[15] The constant product function in Equation 2, $k = (x - \delta_x) \cdot (y + \delta_y)$, can be generalized to $f(k) = f(x - \delta_x, y + \delta_y)$, where $f(\cdot)$ is an arbitrary pricing function. For an overview of automated market makers and pricing functions see Xu et al. (2021).

$1,000 \cdot 90 = 90,000$ and the marginal exchange rate is $\frac{x}{y} = \frac{1,000}{90} = 11.11$. A trader wants to buy 100 of X, and according to the constant product algorithm they have to pay,

$$\delta_y = \frac{k}{x - \delta_x} - y = \frac{90,000}{1,000 - 100} - 90 = 10 \tag{5}$$

of Y at the exchange rate $\frac{\delta_x}{\delta_y} = \frac{100}{10} = 10$, which is less favourable than at the marginal exchange rate.[16] After the trade, the invariant, k, is still equal to $900 \cdot 100 = 90,000$, the liquidity pools have changed to $x = 900$ and $y = 100$, and the new marginal exchange rate is $\frac{x}{y} = \frac{900}{100} = 9$. If another trader, at this point, also wants to buy 100 of X, they have to pay,

$$\delta_y = \frac{k}{x - \delta_x} - y = \frac{90,000}{900 - 100} - 100 = 12.5 \tag{6}$$

of Y, which is more than the first trader, at the exchange rate $\frac{\delta_x}{\delta_y} = \frac{100}{12.5} = 8$. After the trade, the invariant, k, is still equal to $800 \cdot 112.5 = 90,000$, the liquidity pools have changed to $x = 800$ and $y = 112.5$, and the new marginal exchange rate is $\frac{x}{y} = \frac{800}{112.5} = 7.11$. In the example, both traders affect the exchange rate as they alter the liquidity pools, and the second trader pay a higher price, compared to the first trader, for the same amount of currency X.

2.2.2 Crypto assets traded on decentralized exchanges

Crypto assets are digital assets secured by blockchain technology. The most common crypto assets are so-called *cryptocurrencies*. Ether is Ethereum's native currency, but there are thousands of other currencies within the Ethereum ecosystem. These currencies are deployed as smart contracts on the blockchain. The *ERC-20 Token Standard* (Ethereum Request for Comment 20) (Vogelsteller and Buterin, 2015) is a technical standard that allow users to create smart contracts that are fungible, i.e., interchangeable, tokens on the Ethereum blockchain. The ERC-20 standard specify an interface for how to transfer and use the currency. The standard allows developers to create decentralized applications that can interact universally with every currency of that standard. The ERC-20 currencies can be traded on decentralized exchanges running on the Ethereum blockchain. Two examples of ERC-20 tokens are the two largest cryptocurrencies in market cap after bitcoin and ether, the stablecoins Tether (USDT) and USD Coin (USDC), which are both pegged to the US dollar.

[16] For simplicity the example disregards trading costs.

2.3 Arbitrage on decentralized exchanges

Arbitrageurs are part of the price-discovery process of automated market makers, helping adjust the liquidity pools to reflect the no-arbitrage price. Arbitrage opportunities arise when trading has sufficiently changed the price in one market (or asset) but not in another. This can happen through price slippage in one large trade, or several consecutive smaller trades. Deviations from the no-arbitrage price can occur between exchanges in the same currency pair as cross-exchange arbitrage, within an exchange in different currency pairs as triangular arbitrage, or any combination of these.

There are two ways arbitrageurs can find profitable opportunities: (i) Observe the exchange rates in the current global state defined by the previously mined block B_b and try to cut in front of all other market participants to capture this opportunity in block B_{b+1}. (ii) Monitor the mempool to detect unconfirmed pending transactions that will affect the market price such that arbitrage opportunities occur and try to cut in front of all other market participant in capturing these opportunities (Daian et al., 2019). Arbitrageurs can predict exchange rate changes from pending transactions as all information about these transactions is public. Both the former and the latter methods are done in a process called *back-running*, where the arbitrageur adjusts the transaction fee such that their arbitrage transaction is executed directly after the transaction causing the price impact.

If the arbitrage opportunity is already manifested in the current state, arbitrageurs aim to have as high a transaction position as possible in the next block. Thus, paying up to the arbitrage profit itself in transaction fee to acquire an early execution in the coming block. If the arbitrage opportunity is pending in the mempool there are two ways arbitrageurs can capture the potential profits. First, the arbitrageurs can back-run the transaction by carefully choosing the transaction cost such that the probability of the arbitrage transaction being executed directly after the price-changing transaction is high. Alternatively, the arbitrageurs can "take" the transaction in the mempool, sign their own arbitrage transaction, and send these together directly to miners using private relays. The miners are incentivized to accept these bundles if the fees are sufficiently high. This process guarantees that the transactions are executed in the right order, given that the miner wins the race to confirm the block. Arbitrageurs reputedly use private relays to increase the probability of having their transactions executed directly after a large transaction off-sets the exchange rate, although the exact frequency of this practice is unknown.

2.3.1 Arbitrage atomicity

One significant difference between arbitrage trading on decentralized exchanges and traditional exchanges concerns the risks involved. On traditional exchanges, an arbitrage opportunity can generally only be capitalized on by submitting multiple trades. In this situation, there is a risk that the market moves in the opposite direction and the trader has to sell the position at a loss. Noise traders have been shown to create unpredictable risks that reduce the attractiveness of arbitrage (DeLong et al., 1990).

On the Ethereum blockchain, all legs of an arbitrage strategy can be included in the same transaction in which case no other transaction can interrupt the chain of trades. Furthermore, since Ethereum transactions are programmable, arbitrageurs can condition their transactions such that the transactions are cancelled if they are not profitable.[17] In principle, arbitrage on Ethereum is truly risk-free with the caveat that arbitrageurs still have to pay gas fees for cancelled transactions.

2.3.2 Arbitrageurs' profits and costs

The arbitrageurs have to pay two types of fees: A trading fee to the decentralized exchange, and a fee to the Ethereum network for processing the transaction. The trading fee is typically 0.03% on decentralized exchanges and goes to the liquidity providers for providing assets to the liquidity pools. Additionally, arbitrageurs have to pay gas fees to the Ethereum network to incentivize miners to process the transactions. Arbitrageurs using private relays can pay network fees to the miners either by direct payments or through regular gas payments.[18]

3 Primary transaction data

I set up and sync an Ethereum archive node (Erigon Team, 2022), which downloads the entire history of the blockchain. The node contains all transactions on the Ethereum network since its inception on the 30th of June 2015 and onwards. Trueblocks (TrueBlocks Team, 2022) is used to build an index of all transactions such that they can be filtered on interactions with decentralized exchanges.

A transaction-level data set is sourced directly from the archive node, consisting of every transaction interacting with the decentralized exchange Uniswap between 29th of July 2020 and 17th of February 2022. 37,856,529 transactions across 63,168 trading pairs,

[17] This can be done with a simple condition asserting that the end balance need to be greater than or equal to the start balance.

[18] More about this in Section 3.1.3.

and 84 decentralized exchanges are collected.[19] Thus, each transaction in the dataset has one or more interactions with Uniswap, without restricting any further interactions with additional smart contracts, such as other decentralized exchanges. There are three different software versions of Uniswap. In this paper Uniswap version 2 is used since, during the sample period, it is by far the largest decentralized exchange on the Ethereum blockchain.

3.1 Transaction classification

3.1.1 Overview of the data

Uniswap allows for three main user operations: Trading, adding liquidity, and removing liquidity. From the 37,856,529 transactions interacting with the Uniswap smart contracts, 71% concern regular trading where at least one currency is exchanged for another. Of these, 43% are *simple* trades, where only one currency is exchanged for another. Furthermore, 23% of the transactions are liquidity provisions, depositing assets to liquidity pools, and 1% of the transactions are liquidity withdrawals. This is consistent with previous research, finding that liquidity providers do not frequently move their assets between liquidity pools (Heimbach, Wang, and Wattenhofer, 2021). Lastly, 1% of the transactions give the Uniswap smart contracts approval for spending user funds, which is an industry standard for decentralized exchanges and necessary prior to any trading. Thus, approximately 96% of the transactions in the dataset perform standard operations of a decentralized exchange. 4% of the transactions are uncategorized and are possibly combinations of the above operations, or interactions with additional smart contracts on the Ethereum blockchain.

3.1.2 Detecting completed arbitrage transactions

To study price efficiency, I focus on two types of pure atomic arbitrages: Cross-exchange arbitrage and triangular arbitrage. In addition, combinations of the strategies are also included in the analysis. Cross-exchange arbitrages trade on price differences between two or more exchanges. Triangular arbitrage capitalize on price deviations between three or more exchange pairs within the same exchange.[20] The cross-exchange arbitrages all include Uniswap as one of the exchanges, and the triangular arbitrages all occur on Uniswap.

From the decentralized exchange dataset 231,645 completed cross-exchange and triangular arbitrage transactions with associated metadata are extracted. The arbitrage

[19] The transactions are collected from the Ethereum Mainnet network and the dataset includes all decentralized exchanges that use the same *application binary interface* as Uniswap version 2.

[20] Triangular arbitrage with more than 3 legs is sometimes called cyclic arbitrage.

transactions span the same time period as the full dataset, 29th of July 2020 through 17th of February 2022, trade on 82 decentralized exchanges, and across 4,663 cryptocurrency trading pairs. The transactions are publicly recorded on the Ethereum blockchain and for each arbitrage observation the following metadata are collected: Execution time, block, position in block, arbitrageurs' address, trading cost, number of trades, volume of first trade, profit, currency pairs, and decentralized exchanges used. The arbitrage transactions must satisfy the following criteria:

1. Two or more trades need to form a closed loop, such that the output amount and currency of one trade is equal to the input amount and currency of the next trade.
2. All trades must occur within one atomic Ethereum transaction.
3. The transaction cannot perform or be connected to any other operation on the blockchain.
4. The transaction must yield a positive profit.
5. The base currency must be Ether.
6. A fee must be paid to the miner.

The classification criteria capture a set of clean arbitrage transactions, where the arbitrageurs' risk is bounded by the transaction cost payed to validators for trying to execute the transaction. Criterion 1 assures that the transaction is an arbitrage trade. Any number of trades can take place between any number of exchanges. This covers, for example, cross-exchange arbitrage transactions with two or more trades between Uniswap and any other decentralized exchanges, as well as triangular arbitrages on Uniswap. Criterion 2 ensures that all trades in the transaction are done by the same agent and that the arbitrageurs act rationally by minimizing their risks and costs by executing all trades in one atomic transaction. Criterion 3 ensures that the transaction is a pure arbitrage transaction, meaning that it is not part of any other trading strategy. This criterion removes transactions included in sandwich bundles, flash loans, or any other blockchain operation not part of a pure arbitrage operation.[21] Criterion 4 removes arbitrage transactions with negative profits. Since it is possible for an arbitrage transaction to be programmed such that it is cancelled if it is not profitable, transactions with negative profits are most likely due to operational mistakes. Or, the transactions are not aimed at capturing arbitrages, but have some other use case. Wang et al. (2021b) find that approximately 1% of cyclic

[21] For a detailed description see Appendix D.

Table 1: Descriptive statistics of arbitrage transactions. N = 231,645. All currency units are in USD.

	Exchanges	Trades	Position	Volume	Cost	Profit	Net profit
mean	1.51	2.90	77.46	6,573.19	71.88	129.31	57.43
median	1.00	3.00	66.00	2,461.29	39.25	55.16	7.72
std	0.53	0.70	71.32	26,037.69	377.53	789.99	663.89
min	1.00	2.00	0.00	0.00	0.00	0.00	-3,936.86
25%	1.00	2.00	8.00	993.04	20.04	27.12	2.13
50%	1.00	3.00	66.00	2,461.29	39.25	55.16	7.72
75%	2.00	3.00	125.00	5,906.07	73.02	109.90	27.33
max	4.00	8.00	790.00	3,311,271.21	49,655.44	167,860.28	167,818.47
sum	358,720.00	689,714.00	18,439,829.00	1,564,878,324.27	17,113,326.74	30,784,514.82	13,671,188.08

arbitrage transactions have negative profits plausibly due to participation in other decentralized finance projects, such as increasing trading volume. Criterion 5 does not restrict the sample in any essential way, but enables straightforward comparison and profit calculations across transactions. The vast majority of arbitrage transaction fulfill this criterion as arbitrageurs pay transactions costs in ether. Lastly, Criterion 6 ensures that the arbitrageur paid a fee to the miner, either by regular gas payment, or a direct transfer. This removes arbitrages that rely on other transactions for gas payments.

Table 1 shows descriptive statistics for the arbitrage transactions. Approximately half of the transactions are triangular arbitrages indicated by the number of exchanges used and the number of trades per transaction. Cross-exchange arbitrages occur on 2 to 4 exchanges. The maximum number of trades for an arbitrage is 8, but few transactions have more than 3. One reason for this is likely the computational burden to find more complicated arbitrage opportunities.[22] Another reason is the larger transaction costs, which increase in two ways: A larger network fee due to a more complex Ethereum transaction, and higher exchange fees due to more trades. The volume of the arbitrage transactions is calculated by the volume of the first trade in each arbitrage, not adding the volume of later trades in the transaction. In this way it is possible to think about the opportunity cost and required capital for the arbitrageur. Most arbitrages require a capital up to $6,000, although the maximum is over $3 million. The distribution of the transaction costs reveal that the mean is substantial relative to the profits and the standard deviation is high. Further, the

[22] In order for an arbitrageur to find an opportunity, they must identify it within the average Ethereum block time of 14 seconds. Zhou et al. (2021a) estimate that given a 3 seconds network delay their algorithm must detect arbitrage within a 10.5 seconds window. The authors show that their algorithm exceeds the time limit when trying to exploit more than 6 arbitrage cycles.

minimum transaction cost is rounded to 0, although it is non-zero.

After Uniswap, Sushiswap is the largest decentralized exchange. Approximately 95% of the sampled arbitrages use these two exchanges (all of them use Uniswap). Further transactions, using CRO Defi Swap, Shibaswap, and Linkswap amounts to 4% of the arbitrageurs' trading. The stablecoins USDT (Tether), USDC, and DAI traded against Ether are the most traded currency pairs on Uniswap, and also generally the most liquid trading pairs.

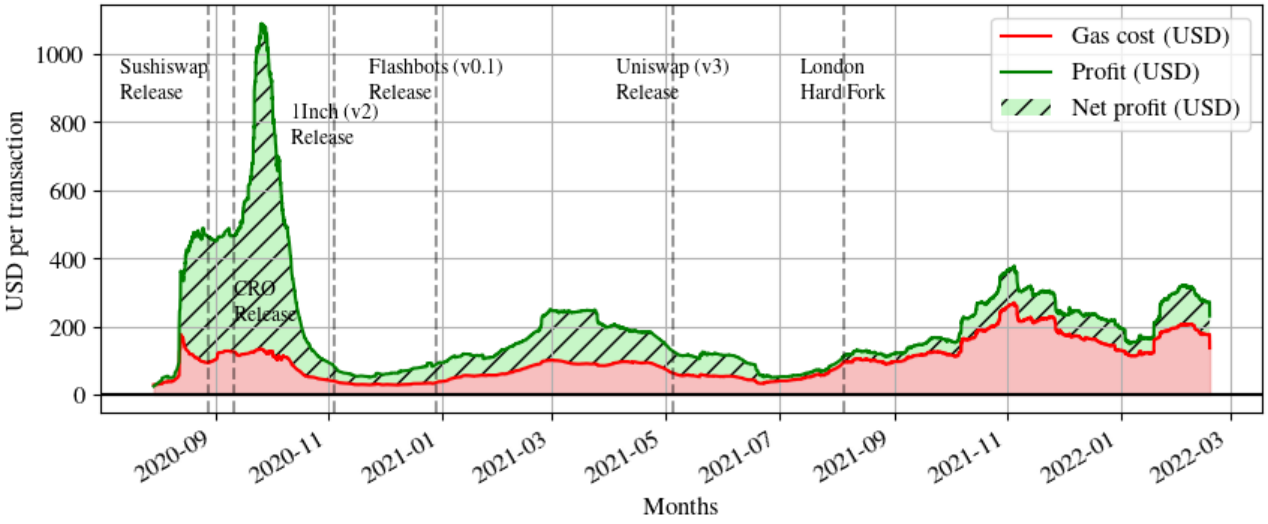


Figure 2: 30-day moving average of gas costs and profits from successful arbitrages.

Unlike other financial markets, where high frequency traders compete primarily with speed, arbitrageurs on decentralized markets also compete with willingness to pay for the arbitrage opportunity, as miners prioritize high-paying transactions. Figure 2 shows the 30-day moving average of successful arbitrage transactions' costs and profits. The dashed area, above the cost curve and below the profits curve, is the average net profits. The net profits are fairly uniformly distributed over the first half of sample period, with the exception of the beginning where Sushiswap and CRO Swap were released. In the second half of the sample, competition seems to have increased as costs increase and net profits looks slightly lower. Important events such as the introduction of *Flashbots'* private relay and the decentralized exchange routing protocol *1Inch* do not, visually, seem to have any significant effect on arbitrage profits.

3.1.3 Approximating arbitrage transaction costs

Transactions that are sent to miners through the mempool pay transaction costs by a regular gas payment. Arbitrage transactions that use private relays have the option to pay miners directly in addition to the standard gas payment. When estimating arbitrageurs'

total costs it is important to consider both alternatives. Regular gas payments are easy to observe as they are logged in the transaction data on the blockchain.[23] However, direct payments to the miner need to be considered separately. Specifically, any transfer to the miner's address needs to be considered. Miners treat direct payments and regular gas payments in the same way, and position transactions that pay the most first in the blocks.

The trading fees that the arbitrageurs pay to the decentralized exchanges do not need to be estimated as they are automatically accounted for in the trade. The total cost for each arbitrage transaction is calculated by adding the regular gas cost and any direct payment. 31 arbitrage transactions are removed from the sample data as they have 0 total cost after the calculation. These transactions probably pay the miners in some other way, and are therefore not considered simple arbitrages in this paper.

The total cost for successful arbitrages can be precisely calculated. However, there is a hidden cost as arbitrageurs do not always succeed in capturing arbitrage. Failed arbitrage transactions still pay a transaction fee to the miner for the attempt to execute the transaction. Assuming that failed transactions from arbitrageurs are attempted arbitrages, Wang et al. (2021b) find that most arbitrageurs have a success rate of over 90%.

3.1.4 Arbitrage transactions' position in blocks

The number of transactions in an Ethereum block varies with the complexity of the transactions and current network demand. The average number of transactions per block in the arbitrage dataset is 204. The distribution of the arbitrage transactions' positions in the blocks is visualized in Figure 3.

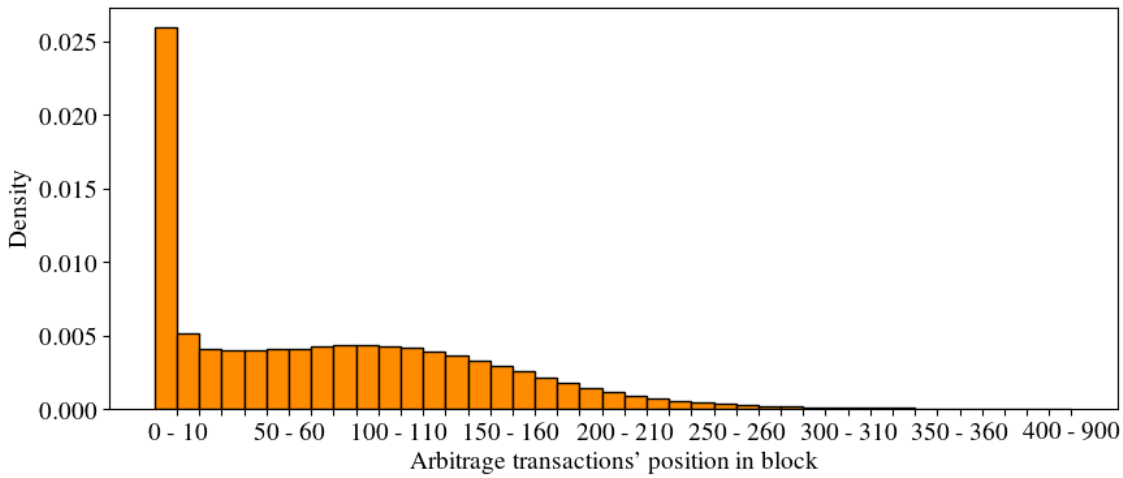


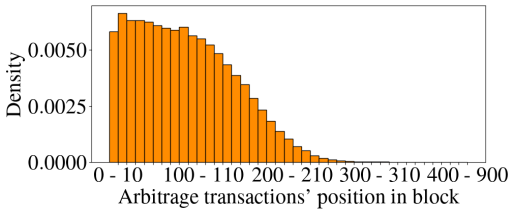
Figure 3: Distribution of arbitrage transactions' block position for the full sample, July 2020 to February 2022. Transactions with block position 0 is executed first in each block.

[23] See Appendix C for a detailed description.

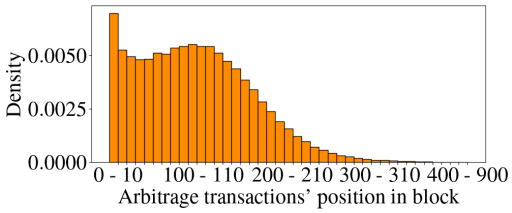
Figure 3 gives insight into what information the arbitrageurs use, what previous transaction might have created the arbitrage opportunity, and if the arbitrageurs might use private relays to send their transactions to the miners. The figure shows that many arbitrage transactions are placed in the beginning of the blocks, with transaction positions 0 to 10. 4.47% of the arbitrage transactions are at block position 0, and thus the first transactions to be executed in the blocks. This indicates that these transactions capitalize on arbitrage opportunities created in previous blocks, as no other transaction is executed before the arbitrage in the current block. Other arbitrage transactions have positions in the middle or end of the blocks, suggesting that these transactions profit from arbitrage opportunities within the same block. However, arbitrageurs that use private relays do not need to wait for an exchange rate changing transaction to be executed and try to back-run it. Instead, arbitrageurs can bundle pending exchange rate changing transactions together with their arbitrage transactions and pay a high amount of transaction fee to the miner to execute these transaction together. Thus, it is possible that privately relayed transactions tend to have a higher block position, such that the arbitrage opportunity is captured early in the block.

Figures 4a to 4d show the distributions of the arbitrage transactions positions in the blocks over 5-month subsamples. Interestingly, in the beginning of the sample period arbitrage transactions tend to be positioned in the middle of the blocks, and in the later part of the sample period the transactions tend to be positioned in the beginning of the blocks. One possible explanation for this is the increased amount of privately relayed arbitrage transactions in the end of the sample, which would be consistent with the seemingly increasing gas costs in the end of the sample in Figure 2.

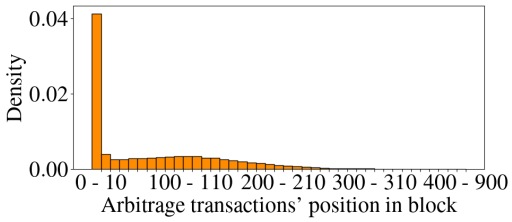
Figure 5 shows a lower bound of privately relayed arbitrage transactions over the sample period. The percentages are calculated based on the number of arbitrage transactions that use a direct payment to the miner, which is only possible through private relays. This however, creates a lower bound for the number of privately relayed transactions as the arbitrageurs do not have to pay miners directly, but can instead do so through regular gas payments. The first privately relayed arbitrage transactions can be observed around the launch of Flashbots' private relay client *MEV Geth* in December 2020, and from then on the percentage of private arbitrage transactions steadily increase. However, after July 2021, there is a declining pattern of arbitrage transactions that pay the miners directly. It is, however, unclear if the percentage of privately relayed arbitrage transactions also decline during this time period, or whether the arbitrageurs changed their primary payment method.



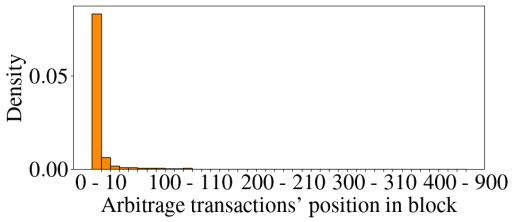
(a) July 2020 to November 2020.



(b) December 2020 to April 2021.



(c) May 2021 to September 2021.



(d) October 2021 to February 2022.

Figure 4: Distribution of arbitrage transactions' block position. Transactions with block position 0 is executed first in each block.

4 Arbitrage analysis

To investigate the occurrence and subsequent elimination of arbitrage opportunities the empirical methodology is designed in three parts. (i) A counterfactual simulation is designed, where arbitrage transactions are re-executed in different blocks and positions. By simulating another, hypothetical, state of the blockchain it is possible to analyze how the arbitrage transactions would have behaved under different circumstances, and draw conclusions from their dependence on previous trading. (ii) A predictive model is designed to estimate how previous trading predicts arbitrages, using the realized arbitrage transactions and a randomly sampled control group of non-arbitrage transactions. (iii) Arbitrage profits are regressed on previous exchange rate changes to understand how profits differ depending on how far in the past the arbitrage-triggering price changes occurred. However, before turning to the full formal analysis, a snapshot of within-block price differences between Uniswap and Sushiswap is visualized together with some completed arbitrage transactions.

4.1 Snapshot of arbitrages

The Ethereum state is updated with each Ethereum transaction and thus multiple times within each block. Therefore, exchange rates on decentralized exchanges change within each block and prices can be measured on a transaction-level basis. Figure 6 shows the difference in the ETH-USDT (ether and Tether stablecoin) exchange rate between Uniswap

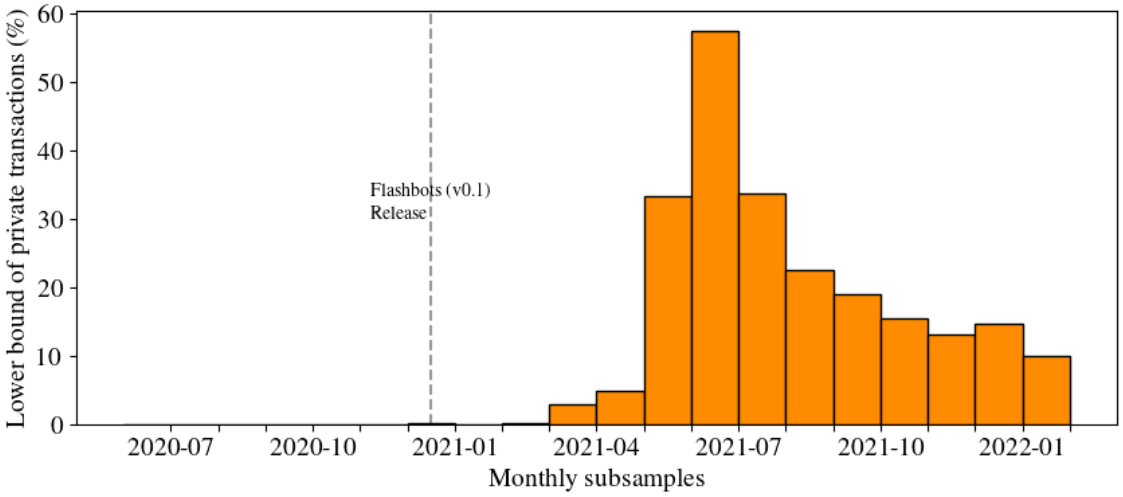


Figure 5: Lower bound of privately relayed arbitrage transactions for each month in the sample.

and Sushiswap. The dashed bars show the end-of-block exchange rate differences between the two exchanges and the solid bars show the maximum differences within the blocks. The snapshot shows approximately 1 hour of trading on the 23 of May 2021. The day was chosen arbitrarily, and the time was chosen to show multiple arbitrage transactions within a short time interval. The maximum within-block price differences include the end-of-block price difference: The bars have the same height if the end-of-block price difference is the largest price difference in that block. If the bars have different heights there has been at least one occasion within that block where a larger price discrepancy has occurred and then disappeared.

At several times during the snapshot, the end-of-block price difference is lower than the within-block price difference. That is, within the block, the price difference has been greater but subsequently corrected before the block ends. In the figure, pure arbitrage transactions are marked with stars and show how within-block price anomalies are regularly arbitraged away. Unsurprisingly, there are also large price differences without any observed classified arbitrage transaction correcting the price. The classification method in this paper focus on pure arbitrage transactions, and prices may adjust due to other types of trades as well.

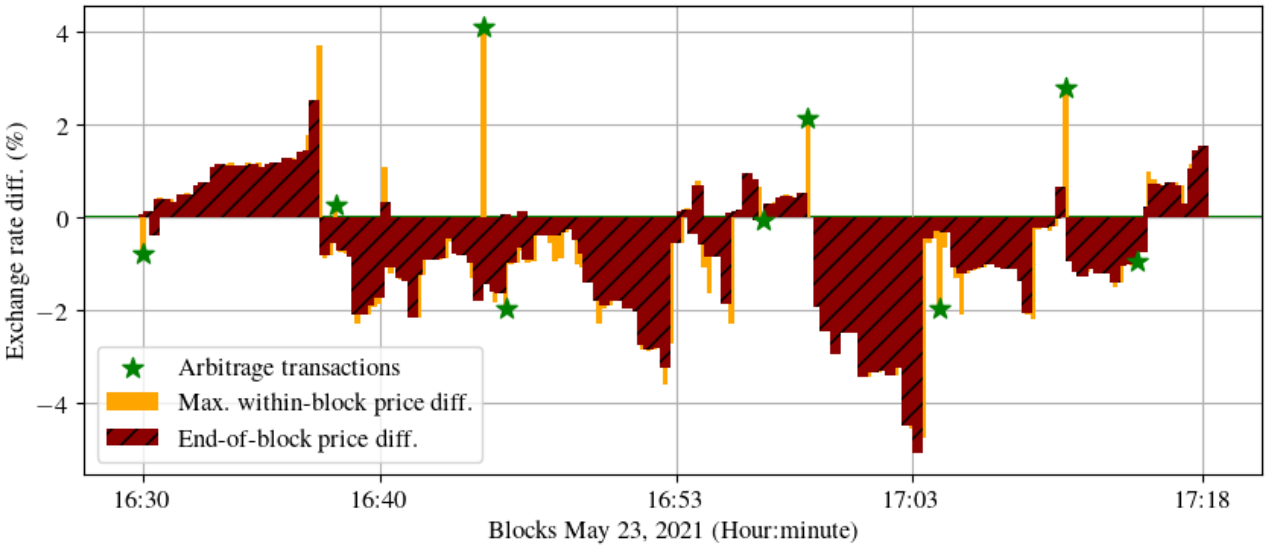


Figure 6: End-of-block price differences and maximum within-block price differences for the exchange pair ETH-USDT (ether and Tether stablecoin) on the decentralized exchanges Uniswap and Sushiswap. The data are taken from a snapshot of blocks from the 23 of May 2021.

Table 2: Descriptive statistics of arbitrage transactions between 2021-05-23 16:30:52 and 2021-05-23 17:14:57. All currency units are in USD. Volume is calcuated as the amount traded in the first leg of the arbitrage.

Time	Position	Arbitrageur	Volume	Cost	Net profit
2021-05-23 16:30:52	6	0x0000000000007f150bd6f54c40a34d7c3d5e9f56	174,988.73	149.51	476.96
2021-05-23 16:38:38	96	0x0000000089341e263b85d84a0eea39f47c37a9d2	441,000.00	188.03	8,315.18
2021-05-23 16:46:02	10	0x3700006fbcde59a8b3af2c134d00e9530000e379	318,253.28	417.57	1,275.97
2021-05-23 16:46:55	83	0x3700006fbcde59a8b3af2c134d00e9530000e379	161,427.18	199.12	239.77
2021-05-23 16:57:47	94	0x0000000000007f150bd6f54c40a34d7c3d5e9f56	11,034.15	158.67	32.03
2021-05-23 16:59:59	54	0xf5b4f13bdbe12709bd3ea280ebf4b936e99b20f2	253,778.07	233.99	1,114.21
2021-05-23 17:05:39	14	0x3700006fbcde59a8b3af2c134d00e9530000e379	337,209.53	397.11	1,605.89
2021-05-23 17:11:07	153	0x3700006fbcde59a8b3af2c134d00e9530000e379	143,005.49	180.07	179.44
2021-05-23 17:14:57	148	0x3700006fbcde59a8b3af2c134d00e9530000e379	170,750.44	187.94	327.63

Table 2 shows some statistics for the arbitrage transactions in Figure 6.[24] The block

[24]One arbitrageur address, `0x...e379`, executes 5 out of the 9 arbitrages in the snapshot. In fact, this arbitrageur captures approximately 20% of the profits in the full arbitrage dataset. All arbitrageur addresses in Table 2, except `0x...20f2`, are present on Etherscan's list of 260 addresses that are heavily involved with miner extractable value such as arbitrage trading (https://etherscan.io/accounts/label/mev-bot). The competition for arbitrages is said to be increasing and solo arbitrageurs are getting outcompeted by teams, both from traditional finance and the cryptocurrency industry. Rumor has it that around 20 teams are dominating the industry (Worsley, 2022). This claim is in line with the data in this paper, where 840 unique arbitrageur addresses are identified, but 76% of the net profits are captured by 20 arbitrage addresses.

positions of the arbitrage transactions reveal that they most likely have back-run pending transactions and not captured arbitrage from previous blocks. The reason being that in order to compete in capturing arbitrage from the previous block the arbitrageurs would like to execute their transactions as early as possible in the block. Looking closer at the last arbitrage in Table 2, with block position 148, there is a previous transaction, with position 146 in the same block (now shown it Table 2), that trade 51 Ether for 100,000 USDT on Sushiswap, off-setting the no-arbitrage exchange rate between the exchanges. The arbitrage transaction capitalizes on this price discrepancy and brings the market back to the no-arbitrage price. This scenario showcase how an arbitrage opportunity arise and how an arbitrageur profit from the price discrepancy.

From Figure 6, it is clear that arbitrageurs act fast to capitalize on arbitrage opportunities and that price anomalies are often corrected within a block. This suggests that arbitrage opportunities are highly time sensitive, appearing for brief moments within blocks. Prices measured at the end of blocks, rather than at the transaction level, therefore seems more likely to be arbitrage-free.

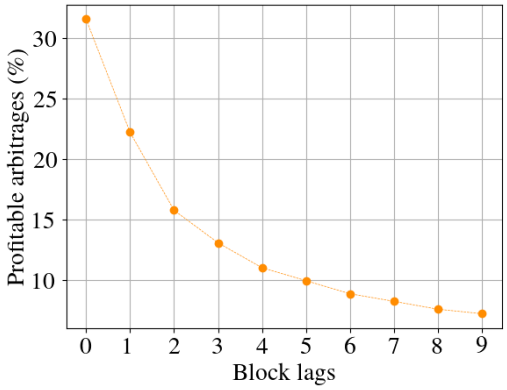
4.2 Counterfactual simulation

The computer code for each Ethereum transaction is fully transparent as it is necessary for node operators to be able to replay every transaction on the network up until the current state. This feature makes it possible to simulate alternative versions of the blockchain. Transactions are defined by their accompanying transaction code and any detail in the code can be changed, such as transaction cost and block position. This gives a unique opportunity to study counterfactual states of the world, in a way not possible in any traditional market. Transactions can be altered and re-ordered in any way possible and the counterfactual results can be analyzed.[25] By re-executing the arbitrage transactions in a different order in the blockchain, it is possible to analyze if the transactions would have been profitable under different circumstances. For example, if an arbitrage transaction is re-executed where the price anomaly does not exist, the transaction will either be cancelled or show negative profits.

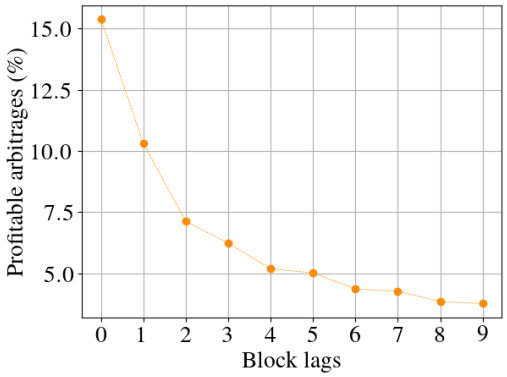
A counterfactual simulation is designed to re-execute each arbitrage transaction as the first transaction in their own block or in previous blocks to determine at which point it is no longer profitable. The hypothesis is that if the arbitrage transactions are primarily capitalizing on exchange rate differences created in their own blocks, then the transactions would no longer be profitable if executed as the first transaction in their own blocks, i.e., the arbitrage transactions are placed before the price anomalies occur. Similarly, if an

[25] Appendix C gives an overview of the data of an Ethereum transaction.

arbitrage transaction profits from exchange rate changes in the previous block, it would no longer be profitable if it were executed as the first transaction in the previous block.



(a) Net profits with original transaction costs.



(b) Net profits with updated transaction costs.

Figure 7: Counter factual simulation with 9 lags.

Figures 7a and 7b shows the percentage of arbitrages that are still profitable when re-executed as the first transaction in their current and previous blocks.[26] Figure 7a keeps the original transaction costs of the arbitrage transactions, whereas Figure 7b uses the transaction cost the arbitrage transaction would have payed to be in the simulated position. By re-executing each arbitrage transaction at the beginning of its own block, only 31% of the transactions are profitable. This suggests that a vast majority (69%) of the arbitrage transactions profit from exchange rate changes within the same block. The further away from its original position that the arbitrage transaction is re-executed, the less likely it is to be profitable. When the transactions are re-executed as the first transaction in the block that was mined 9 blocks from its original position, only 7% are profitable. Put differently, most arbitrage opportunities are eliminated within 9 blocks. Since a block is mined approximately every 14 seconds, it takes roughly 2.5 minutes for all arbitrage to disappear. However, after an average of 14 seconds the majority of the arbitrage profits are made.

The counterfactual simulation shows that arbitrage transactions are very sensitive to the order of executions. Arbitrageurs need to act fast and precise when back-running pending transactions from the mempool. The arbitrageurs have to identify the arbitrage opportunity within the average block window of 14 seconds, and carefully place their transaction after the exchange rate changing trade, either by adjusting the transaction cost to pay marginally less than the pending trade or by submitting the two transactions

[26] As a robustness check, the simulations are also run excluding the arbitrage transactions at position 0. See Appendix B.

in a private transaction bundle.

4.3 Predicting arbitrages by prior trading

On decentralized exchanges the marginal exchange rate between two assets, X and Y, is determined by the fraction of their liquidity pools, $\frac{x}{y}$. Arbitrages are solely created by agents trading against the liquidity pools such that the no-arbitrage exchange rate is off-set. Large exchange rate changes should therefore signal future arbitrage transactions. The counterfactual simulation shows that arbitrage transaction are sensitive to timing, but that some arbitrage opportunities exist as far back as 9 blocks from the original arbitrage. To empirically evaluate how far back exchange rate changes affect arbitrage transactions, I conduct a predictive study in which imbalances in the liquidity pools are used to estimate whether or not an arbitrage transaction is likely to occur. The exercise also investigates the dynamics of arbitrages in the sense that it indicates how long-lived arbitrages are by answering: How far back into the past do I have to look for price imbalances to predict current arbitrage?

In the predictive exercise, $n = 231,645$ arbitrage transactions are studied. To discriminate these arbitrage transactions from regular trading, a random control group is constructed from the original dataset. Specifically, the control group consists of n randomly sampled non-arbitrage transactions. These transactions are sampled uniformly without replacement, under the criteria that they perform one trade on Uniswap. The total sample size is thus equal to $2n = 463,290$ transactions. A transaction t_{b_i,p_i} is assigned $A_i = 1$ if it is an arbitrage transaction, and $A_i = 0$ if the transaction belongs to the control group. Transaction t_{b_i,p_i}, $i = 1, \ldots, 2n$, exists in block b_i at position p_i. As the number of positions differ in each block, block b has P_b positions.

As price changes are fully deterministic on automated market makers, there is a direct mapping from trading to changes in price. This makes it possible to calculate the exact price impact of each transaction. The price impact of a transaction $t_{b,p}$ on the exchange rate $\frac{x}{y}$, is defined by the log difference in the liquidity pools x and y,

$$\Delta_{b,p}(x, y) = \log\left(\frac{x_{b,p-1}}{y_{b,p-1}}\right) - \log\left(\frac{x_{b,p}}{y_{b,p}}\right). \tag{7}$$

Here $\frac{x_{b,p-1}}{y_{b,p-1}}$ is the exchange rate before transaction $t_{b,p}$ is executed and $\frac{x_{b,p}}{y_{b,p}}$ it the exchange rate after transaction $t_{b,p}$ is executed. To study how a previous transaction $t_{b,p}$ affects t_{b_i,p_i}, the maximum absolute price impact of $t_{b,p}$ related to t_{b_i,p_i} is defined as,

$$\phi_{b,p}(i) = \max_{(x,\,y) \text{ such that } x,\, y \text{ are traded in } t_{b_i,p_i}} \left|\Delta_{b,p}(x,y)\right|. \tag{8}$$

Thus, Equation 8 describes the maximum impact on the exchange rates traded in t_{b_i,p_i} by a prior transaction $t_{b,p}$. Although, 43% of $t_{b,p}$ is only trading in one currency pair (see Section 3.1.1), transaction t_{b_i,p_i} can trade in several exchange rates, often across several exchanges. Here, only the maximum absolute exchange rate change is captured by $\phi_{b,p}(i)$. This is a somewhat conservative measure of how trading in transaction $t_{b,p}$ affects transaction i, and as a robustness check the *sum* is also calculated.

In order to predict if transaction t_{b_i,p_i} is an arbitrage or not, the price impacts from transactions executed prior to t_{b_i,p_i} are calculated. The maximum price impact of the transaction immediately prior to t_{b_i,p_i}, i.e., t_{b_i,p_i-1}, is defined as,

$$PrevTrans_i = \phi_{b_i,p_i-1}(i). \tag{9}$$

The relationship between $PrevTrans_i$ and transaction t_{b_i,p_i} indicates to what degree the transaction immediately prior to t_{b_i,p_i} helps to predict if it is an arbitrage. Arbitrageurs can profit from a large $\phi_{b_i,p_i-1}(i)$ by identifying t_{b_i,p_i-1} when it is pending in the mempool, calculate its exact price impact given the current state, and either back-run it through a private transaction bundle or a regular public transaction. If done successfully, this is the absolute fastest way price anomalies can be arbitraged away, as there are no other transaction between t_{b_i,p_i-1} and t_{b_i,p_i}. In this situation the exchange rate is back at the no-arbitrage price within the next transaction.

Furthermore, to measure the magnitude of the changes in the exchange rates in the same block as transaction t_{b_i,p_i}, the maximum price impact of all prior transactions in the block, excluding transaction t_{b_i,p_i-1}, is defined as,

$$SameBlock_i = \max\{\phi_{b_i,0}(i), \ldots, \phi_{b_i,p_i-2}(i)\}. \tag{10}$$

The price impact $SameBlock_i$ describes how the exchange rate changed prior to transaction t_{b_i,p_i} in block b_i. This relationship further describes to what extent prior trading in the same block predicts arbitrage. For arbitrageurs to profit from a large $SameBlock_i$, the arbitrageur need to identify a pending transaction that will significantly affect the exchange rate, but is not able to place the arbitrage transaction immediately after it. For some reason there is at least one other transaction in between the exchange rate changing transaction and the arbitrage transaction. However, in terms of price efficiency, the exchange rate will still be arbitrage-free within the block.

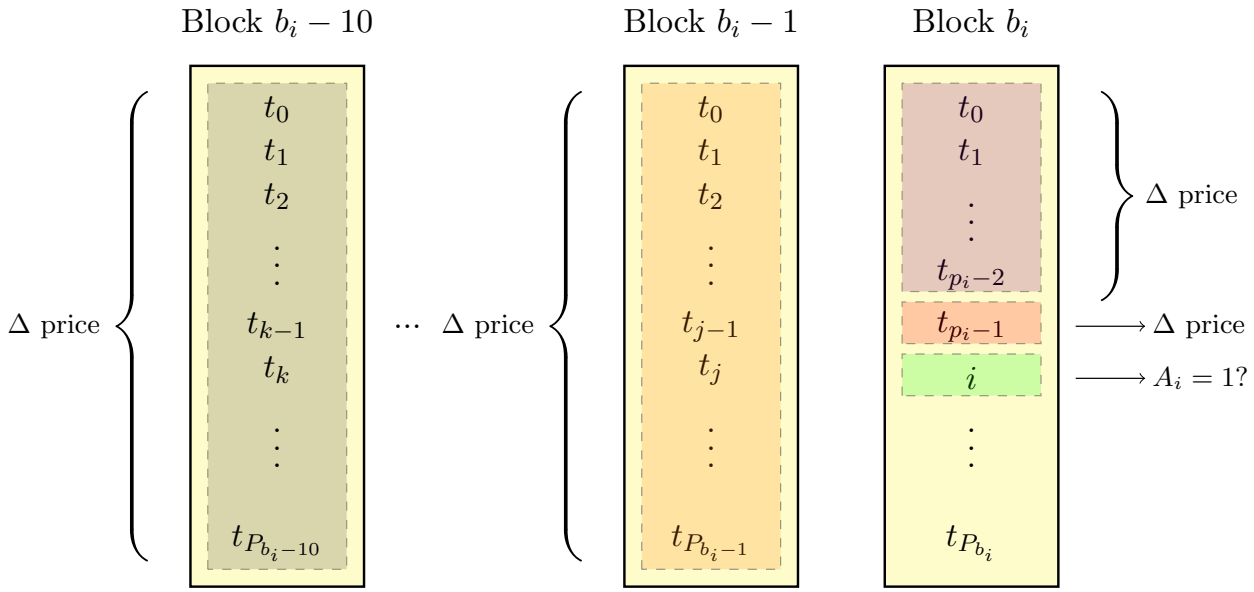


Figure 8: For each transaction in the sample data, the maximum price impact of previous transactions is calculated in groups. The notation in the figure is simplified as the blocks are labeled in the headings instead of as subscripts.

Although much of the arbitrage action happens within the block, the counterfactual simulation (Section 4.2) shows that some arbitrages live across blocks. By analyzing how far back transactions affect arbitrages, it is possible to get a complete measure of how fast arbitrageurs are able to correct exchange rate deviations. To investigate how far back exchange rate changes affect t_{b_i,p_i}, the maximum change in the exchange rates from transactions in previous blocks $b_i - s$, $s = 1, \ldots, 10$ is measured as,

$$PrevBlock_{i,s} = \max\{\phi_{b_i-s,0}(i), \ldots, \phi_{b_i-s,P_{b_s}}(i)\}. \tag{11}$$

The relation between $PrevBlock_{i,s}$ and transaction t_{b_i,p_i}, describes how far back exchange rate changes affect arbitrages. One reason for why arbitrageurs are not always able to back-run arbitrage creating transactions is that some blocks are mined faster than average. Although, arbitrageurs have on average 14 seconds to observe pending transactions, calculate their price impacts, and take action, this is not always the case as some blocks are mined as fast as within 1 second.

Figure 8 visually illustrates how $PrevTrans_i$, $SameBlock_i$, and $PrevBlock_{i,1}, \ldots, PrevBlock_{i,10}$ are calculated. Tables 3 and 4 show the distributions of $PrevTrans_i$, $SameBlock_i$, and $PrevBlock_{i,1}, \ldots, PrevBlock_{i,4}$ for the arbitrage transactions, $A_i = 1$, and the non-arbitrage control group, $A_i = 0$, respectively. The transactions just prior to the arbitrage transactions affect the exchange rate the most on average, with a 3% impact. 27% of the $PrevTrans_i$ observations are non-zero

Table 3: Descriptives statistics of the maximum price changes by trading prior to the arbitrage transactions. All units are in log differences multiplied by 100 to be interpreted as percentages.

	$PrevTrans_i$	$SameBlock_i$	$PrevBlock_{i,1}$	$PrevBlock_{i,2}$	$PrevBlock_{i,3}$	$PrevBlock_{i,4}$
mean	3.16	0.57	2.11	1.46	0.91	0.66
median	0.00	0.00	0.00	0.00	0.00	0.00
std	56.24	7.90	37.87	37.85	30.82	20.13
min	0.00	0.00	0.00	0.00	0.00	0.00
25%	0.00	0.00	0.00	0.00	0.00	0.00
50%	0.00	0.00	0.00	0.00	0.00	0.00
75%	0.45	0.00	0.49	0.05	0.01	0.01
max	6,914.57	1,927.32	7,525.22	6,936.49	7,828.48	7,201.81
sum	735,546.11	132,226.65	491,477.84	340,514.61	211,374.69	153,207.66
% nonzero	27.01	20.37	48.38	42.26	37.35	35.43

for the arbitrages. These statistics are significantly lower for the control group, where $PrevTrans_i$ is 0.22% on average, and barely 5% are non-zero. Similarly, $SameBlock_i$, and $PrevBlock_{i,1}, \ldots, PrevBlock_{i,10}$ are all significantly higher for the arbitrages compared to the control group.

To predict whether a transaction is an arbitrage, i.e., $A_i = 1$, or a regular transaction from the non-arbitrage control group, i.e., $A_i = 0$, I fit a probit regression with predictors $PrevTrans_i$, $SameBlock_i$, and $PrevBlock_{i,1}, \ldots, PrevBlock_{i,10}$ on the full sample and on 5-month subsamples. The subsamples are constructed to investigate any dynamics in the data.

The results are presented in Table 5 and show that exchange rate changes in transactions up to 4 blocks prior to transaction t_{b_i,p_i}, significantly predict if the transaction is an arbitrage.[27] Thus, trading up to 1 minute before a transaction helps to predict if an arbitrage transaction will occur. One explanation for these results are that price imbalances build up. The price could be different between two exchanges without arbitrage being profitable. Arbitrageurs would arbitrage away price imbalances only when it is profitable to do so. However, the probability of a profitable price imbalance increases as trading is pushing the exchange rate in the off-setting direction. These findings are consistent with those of the counterfactual simulation in Section 4.2, in which some arbitrages are profitable over blocks. Approximately 12% (5% with updated transaction costs) of the arbitrage transactions are profitable 4 blocks back, and after which there is clear levelling

[27] A probit model using the summation of previous price changes without taking the absolute value is used as a robustness check and shows similar results, see Table 7 in Section B.

Table 4: Descriptives statistics of the maximum price changes by trading prior to the control transactions. All units are in log differences multiplied by 100 to be interpreted as percentages.

	$PrevTrans_i$	$SameBlock_i$	$PrevBlock_{i,1}$	$PrevBlock_{i,2}$	$PrevBlock_{i,3}$	$PrevBlock_{i,4}$
mean	0.22	0.38	0.50	0.48	0.45	0.43
median	0.00	0.00	0.00	0.00	0.00	0.00
std	5.42	8.70	11.84	11.00	10.37	6.25
min	0.00	0.00	0.00	0.00	0.00	0.00
25%	0.00	0.00	0.00	0.00	0.00	0.00
50%	0.00	0.00	0.00	0.00	0.00	0.00
75%	0.00	0.00	0.00	0.00	0.00	0.00
max	1,749.97	4,143.30	3,425.05	2,901.43	4,715.84	2,462.47
sum	68,607.19	116,404.43	151,211.56	146,227.55	138,145.99	131,055.41
% nonzero	4.91	18.57	26.40	25.35	24.86	24.78

off in the fraction of profitable arbitrages.

4.4 The effect of prior trading on arbitrage profits

Due to the unique features of the Ethereum data, costs and profits can be precisely calculated for the successful arbitrage transactions. As a further step in understanding arbitrage trading on decentralized exchanges, net profits are regressed on $PrevTrans_i$, $SameBlock_i$, and $PrevBlock_{i,1}, \ldots, PrevBlock_{i,10}$.[28] In the regression, the block times of the blocks b_i, b_{i-1}, and b_{i-2} are used as control variables. The reason is that if the block b_{i-1} is mined fast, arbitrageurs might not have enough time to capture the arbitrage opportunity in the same block. If 2 consecutive blocks are mined fast, arbitrageurs might not be able to capture the profits for 2 blocks. The estimation is conducted on the sample of $n = 231,645$ arbitrage transactions, as well as on 5-month subsamples.

Table 6 presents the results from the regressions. The first column displays the results from the regression using the full sample. It shows that previous exchange rate changes up to $PrevBlock_{i,1}$ significantly affect net arbitrage profits. These results are different from the predictive study, and indicate that arbitrageurs' profits are generated by trading much closer to the arbitrage than the trading that predicts the arbitrage. Furthermore, the results show that changes in the exchange rates from transactions within the arbitrage transaction's block, $PrevTrans_i$ and $SameBlock_i$, strongly affect arbitrage net profits. This holds true across all subsamples, but changes somewhat over time. Looking at the

[28] As a robustness check, a regression is run using the summation of the previous price changes without taking the absolute value, see Table 9 in Appendix B.

Table 5: Probit: Arbitrage trades and non-arbitrage trades regressed on maximum of previous price changes.

	(1) All	(2) July 2020 - Nov 2020	(3) Dec 2020 - April 2021	(4) May 2021 - Sep 2021	(5) Oct 2021 - Feb 2022
$PrevTrans_i$	0.0468***	1.607***	0.555***	0.0719***	0.0260***
	(0.00188)	(0.0270)	(0.0165)	(0.00472)	(0.00204)
$SameBlock_i$	0.0526***	0.309***	0.552***	0.132***	-0.0148
	(0.00890)	(0.0387)	(0.0385)	(0.0194)	(0.0106)
$PrevBlock_{i,1}$	0.0489***	0.833***	0.0971***	0.0751***	0.0219***
	(0.00263)	(0.0324)	(0.00671)	(0.00620)	(0.00319)
$PrevBlock_{i,2}$	0.0288***	0.0196***	0.240***	0.0776***	0.0144***
	(0.00275)	(0.00488)	(0.0150)	(0.00838)	(0.00360)
$PrevBlock_{i,3}$	0.0222***	0.0130*	0.0215***	0.121***	0.0154**
	(0.00336)	(0.00604)	(0.00613)	(0.0152)	(0.00543)
$PrevBlock_{i,4}$	0.0214***	0.00887	0.201***	0.0166*	0.0688***
	(0.00489)	(0.00647)	(0.0334)	(0.00757)	(0.0190)
$PrevBlock_{i,5}$	0.00751	-0.130***	0.0581	0.0000517	0.0105
	(0.00766)	(0.0341)	(0.0339)	(0.00870)	(0.0197)
$PrevBlock_{i,6}$	0.0185*	-0.145***	0.00781	0.117***	0.0600***
	(0.00823)	(0.0362)	(0.0101)	(0.0287)	(0.0182)
$PrevBlock_{i,7}$	-0.00395	0.00422	0.130***	-0.000220	-0.00637
	(0.00411)	(0.0292)	(0.0372)	(0.0104)	(0.00453)
$PrevBlock_{i,8}$	-0.00940***	-0.0516**	-0.108***	0.0615*	-0.00753**
	(0.00272)	(0.0178)	(0.0296)	(0.0276)	(0.00275)
$PrevBlock_{i,9}$	-0.00592	-0.284***	0.0473	0.0332	0.000425
	(0.00761)	(0.0452)	(0.0307)	(0.0244)	(0.00868)
$PrevBlock_{i,10}$	0.00332	0.00221	-0.0914**	0.0501*	0.00559
	(0.00637)	(0.00686)	(0.0328)	(0.0222)	(0.0198)
Constants	0.495***	0.349***	0.551***	0.607***	0.430***
	(0.000740)	(0.00146)	(0.00129)	(0.00135)	(0.00228)
N	463290	114701	154408	133478	47987

Standard errors in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

later part of the sample, May 2021 to September 2021 and October 2021 to February 2022, only price changes in the current block, $PrevTrans_i$ and $SameBlock_i$, affect net profits. This indicates that the decentralized exchanges have become more efficient over time and arbitrage competition seems to have increased. In the later part of the sample it is, on average, no longer possible to use information from the last state (block) of the blockchain and previous states (blocks) for profitable arbitrage trading. Arbitrageurs need to observe pending transactions in order to profit. One explanation for these results are the increased

Table 6: OLS estimation: Arbitrage profits and net profits regressed on maximum of previous price changes.

	(1) All	(2) July 2020 - Nov 2020	(3) Dec 2020 - April 2021	(4) May 2021 - Sep 2021	(5) Oct 2021 - Feb 2022
$PrevTrans_i$	19.94***	261.3***	1060.5***	16.98*	6.191*
	(2.409)	(20.09)	(23.44)	(7.102)	(2.869)
$SameBlock_i$	390.7***	539.0***	591.2***	446.0***	140.4**
	(20.56)	(42.44)	(68.14)	(29.72)	(44.81)
$PrevBlock_{i,1}$	10.01**	515.3***	20.95*	14.56	2.952
	(3.583)	(30.90)	(9.105)	(9.309)	(4.850)
$PrevBlock_{i,2}$	4.265	-1.533	92.91***	14.54	-0.865
	(3.576)	(3.456)	(21.02)	(11.77)	(5.150)
$PrevBlock_{i,3}$	0.475	0.952	3.033	-3.902	-1.669
	(4.504)	(4.316)	(8.317)	(23.04)	(8.230)
$PrevBlock_{i,4}$	5.217	2.407	10.86	4.066	9.475
	(6.758)	(4.613)	(51.09)	(12.45)	(31.57)
$PrevBlock_{i,5}$	105.1***	24.17	756.5***	-23.15	42.94
	(20.29)	(43.36)	(63.19)	(34.17)	(37.59)
$PrevBlock_{i,6}$	13.75	12.12	-35.77	8.605	23.27
	(15.19)	(34.28)	(22.69)	(47.57)	(30.71)
$PrevBlock_{i,7}$	8.867	-31.62	83.56	-9.408	-7.213
	(15.69)	(25.86)	(54.89)	(33.86)	(24.49)
$PrevBlock_{i,8}$	53.28**	46.15	265.5***	23.88	11.72
	(18.21)	(53.66)	(67.82)	(43.58)	(24.60)
$PrevBlock_{i,9}$	10.69	106.4*	-81.49	61.18	2.110
	(10.82)	(52.35)	(70.12)	(44.51)	(13.10)
$PrevBlock_{i,10}$	5.085	-0.0608	-85.25	-0.235	4.328
	(8.725)	(4.866)	(67.25)	(33.73)	(40.58)
$SameBlock_i$ Block time	0.196	0.147	0.373*	-0.0590	-0.133
	(0.108)	(0.133)	(0.184)	(0.198)	(0.355)
$PrevBlock_{i,1}$ Block time	0.234*	0.0810	-0.220	0.769***	-0.0689
	(0.109)	(0.131)	(0.181)	(0.202)	(0.394)
$PrevBlock_{i,2}$ Block time	0.0544	0.0526	0.00909	-0.130	0.678
	(0.109)	(0.130)	(0.180)	(0.203)	(0.409)
Constant	45.98***	24.58***	56.55***	20.15***	57.61***
	(2.832)	(3.513)	(4.839)	(5.239)	(9.313)
N	231639	42211	86830	81754	22278

Standard errors in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

usage of private relays in the later part of the sample, which allows arbitrageurs to capture arbitrage opportunities with a higher precision.

The atomicity of the Ethereum transaction ensures that the arbitrage risk is reduced. On the blockchain, compared to traditional markets, arbitrageurs have the advantage to be able to calculate the exact price changes of pending transactions, and are thus able to precisely forecast arbitrage opportunities. Although speed is of importance, arbitrageurs on the blockchain have an average of 14 seconds to do their calculations before the next block is mined. These features are reflected in the empirical results that suggest that in most cases, these 14 seconds are sufficient for the arbitrageurs to act and thus eliminate the arbitrage before the end of the block.

Furthermore, the results have some implications for the use of exchange rates from decentralized exchanges as reference prices on the blockchain. As Ethereum is an isolated system, it is unable to receive external data from the "outside" world. Therefore, decentralized exchanges are used for reference pricing, and smart contracts can query market information from the exchanges on-chain. As deviations from the no-arbitrage price are prone to be arbitraged away within the block, the end-of-block prices are likely to be arbitrage-free and suitable as reference prices.

5 Conclusion

In this paper, I show that arbitrageurs contribute to price efficiency on decentralized exchanges by neutralizing price anomalies. This happens very fast, and most of the arbitrage opportunities are created and capitalized on within the Ethereum block. These effects are stonger in the later part of the sample, where only trading in the same block as the arbitrage transaction affects its profits. Arbitrageurs in the later part of the sample have to monitor pending transactions in order to profit. The results speaks to an increased arbitrage competition over time. The speed at which price anomalies are arbitraged away implies that end-of-block prices are likely to be arbitrage-free. This is important as traders place orders based on the price from the previous block, and other on-chain applications use these prices as reference prices.

The results show that arbitrages are created by trading that off-sets the no-arbitrage price. A natural question arises: Do arbitrage opportunities need to occur in the first place? One way around a large price impact is to split an order into multiple smaller orders across multiple exchanges within the same transaction. The trades could be routed over several exchanges and exchange pairs such that no arbitrage opportunity arises. The trade-off for this kind of order routing would be between the expected price slippage and

the increased transaction costs for splitting the trades.

References

Adams, Hayden, Noah Zinsmeister, and Dan Robinson (2020). *Uniswap v2 core*. Tech. rep. Tech. rep., Uniswap (cit. on pp. 3, 11).

Adams, Hayden, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson (2021). *Uniswap v3 core*. Tech. rep. Tech. rep., Uniswap (cit. on p. 11).

Alexander, C. and M. Dakos (2020). "A critical investigation of cryptocurrency data and analysis". In: *Quantitative Finance* 20.2, pp. 173–188. DOI: 10.1080/14697688.2019.1641347. eprint: https://doi.org/10.1080/14697688.2019.1641347. URL: https://doi.org/10.1080/14697688.2019.1641347 (cit. on p. 3).

Angeris, Guillermo, Hsien-Tang Kao, Rei Chiang, Charlie Noyes, and Tarun Chitra (2021). *An analysis of Uniswap markets*. arXiv: 1911.03380 [q-fin.TR] (cit. on p. 11).

Back, Adam (2002). *Hashcash - A Denial of Service Counter-Measure*. White paper. URL: http://www.hashcash.org/papers/hashcash.pdf (cit. on p. 6).

Bank for International Settlements (June 2022). *Miners as intermediaries: extractable value and market manipulation in crypto and DeFi*. Tech. rep. URL: https://www.bis.org/publ/bisbull58.htm (cit. on p. 10).

Berg, Henry and Todd A Proebsting (2009). "Hanson's automated market maker". In: *The Journal of Prediction Markets* 3.1, pp. 45–59 (cit. on p. 11).

Berg, Jan Arvid, Robin Fritsch, Lioba Heimbach, and Roger Wattenhofer (2022). *An Empirical Study of Market Inefficiencies in Uniswap and SushiSwap*. DOI: 10.48550/ARXIV.2203.07774. URL: https://arxiv.org/abs/2203.07774 (cit. on p. 3).

Buterin, Vitalik (2013). *A Next-Generation Smart Contract and Decentralized Application Platform*. White paper. URL: https://ethereum.org/en/whitepaper/ (cit. on p. 6).

Cartea, Álvaro, Fayçal Drissi, and Marcello Monga (June 2022). "Decentralised Finance and Automated Market Making: Execution and Speculation". In: *SSRN Electronic Journal*. DOI: https://dx.doi.org/10.2139/ssrn.4144743 (cit. on p. 12).

Chaboud, Alain P., Benjamin Chiquoine, Erik Hjalmarsson, and Clara Vega (2014). "Rise of the Machines: Algorithmic Trading in the Foreign Exchange Market". In: *The Journal of Finance* 69.5, pp. 2045–2084. ISSN: 00221082, 15406261. URL: http://www.jstor.org/stable/43612951 (visited on 10/07/2022) (cit. on p. 4).

Cong, Lin, Xi Li, Ke Tang, and Yang Yang (2019). "Crypto Wash Trading". In: *SSRN Electronic Journal*. ISSN: 1556-5068. DOI: 10.2139/ssrn.3530220. URL: http://dx.doi.org/10.2139/ssrn.3530220 (cit. on p. 3).

Daian, Philip, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels (2019). *Flash Boys 2.0: Frontrunning, Transaction Re-*

ordering, and Consensus Instability in Decentralized Exchanges. arXiv: 1904.05234 [cs.CR] (cit. on pp. 2, 9, 10, 14).

DeLong, J. Bradford, Andrei Shleifer, Lawrence H. Summers, and Robert J. Waldmann (1990). "Noise Trader Risk in Financial Markets". In: *Journal of Political Economy* 98.4, pp. 703–738. ISSN: 00223808, 1537534X. URL: http://www.jstor.org/stable/2937765 (visited on 10/12/2022) (cit. on p. 15).

Dwork, Cynthia and Moni Naor (1992). "Pricing via Processing or Combatting Junk Mail". In: *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO '92. Berlin, Heidelberg: Springer-Verlag, pp. 139–147. ISBN: 3540573402 (cit. on p. 6).

Erigon Team (Feb. 2022). *Erigon*. URL: https://github.com/ledgerwatch/erigon (cit. on pp. 15, 46).

Eskandari, Shayan, Seyedehmahsa Moosavi, and Jeremy Clark (2020). "SoK: Transparent Dishonesty: Front-Running Attacks on Blockchain". eng. In: *Financial Cryptography and Data Security*. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 170–189. ISBN: 9783030437244 (cit. on p. 9).

Hanson, Robin (2003). "Combinatorial information market design". In: *Information Systems Frontiers* 5.1, pp. 107–119 (cit. on p. 11).

Härdle, Wolfgang Karl, Campbell R Harvey, and Raphael C G Reule (Feb. 2020). "Understanding Cryptocurrencies*". In: *Journal of Financial Econometrics* 18.2, pp. 181–208. ISSN: 1479-8409. DOI: 10.1093/jjfinec/nbz033. eprint: https://academic.oup.com/jfec/article-pdf/18/2/181/33218309/nbz033.pdf. URL: https://doi.org/10.1093/jjfinec/nbz033 (cit. on p. 6).

Harvey, Campbell R., Tarek Abou Zeid, Teun Draaisma, Martin Luk, Henry Neville, Andre Rzym, and Otto van Hemert (May 2022). "An Investor's Guide to Crypto". In: *SSRN Electronic Journal*. DOI: http://dx.doi.org/10.2139/ssrn.4124576 (cit. on p. 6).

Heimbach, Lioba, Ye Wang, and Roger Wattenhofer (2021). *Behavior of Liquidity Providers in Decentralized Exchanges*. DOI: 10.48550/ARXIV.2105.13822. URL: https://arxiv.org/abs/2105.13822 (cit. on pp. 11, 16).

Makarov, Igor and Antoinette Schoar (2020). "Trading and arbitrage in cryptocurrency markets". In: *Journal of Financial Economics* 135.2, pp. 293–319. ISSN: 0304-405X. DOI: https://doi.org/10.1016/j.jfineco.2019.07.001. URL: https://www.sciencedirect.com/science/article/pii/S0304405X19301746 (cit. on p. 3).

— (Apr. 2022). *Cryptocurrencies and Decentralized Finance (DeFi)*. Working Paper 30006. National Bureau of Economic Research. DOI: 10.3386/w30006. URL: http://www.nber.org/papers/w30006 (cit. on pp. 6, 11).

Nakamoto, Satoshi (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. White paper. URL: https://bitcoin.org/bitcoin.pdf (cit. on p. 6).

Qin, Kaihua, Liyi Zhou, and Arthur Gervais (2021). *Quantifying Blockchain Extractable Value: How dark is the forest?* arXiv: 2101.05511 [cs.CR] (cit. on pp. 10, 55).

Savage, Leonard J. (1971). "Elicitation of Personal Probabilities and Expectations". In: *Journal of the American Statistical Association* 66.336, pp. 783–801. ISSN: 01621459. URL: http://www.jstor.org/stable/2284229 (visited on 08/03/2022) (cit. on p. 11).

SEC (Mar. 2019). *Memorandum: Meeting with Bitwise Asset Management, Inc., NYSE Arca, Inc., and Vedder Price P.C.* Tech. rep. URL: https://www.sec.gov/comments/sr-nysearca-2019-01/srnysearca201901-5164833-183434.pdf (cit. on p. 3).

Shapley, Lloyd and Martin Shubik (1977). "Trade Using One Commodity as a Means of Payment". In: *Journal of Political Economy* 85.5, pp. 937–968. ISSN: 00223808, 1537534X. URL: http://www.jstor.org/stable/1830340 (visited on 07/26/2022) (cit. on p. 11).

Shynkevich, Andrei (2021). "Bitcoin arbitrage". In: *Finance Research Letters* 40, p. 101698. ISSN: 1544-6123. DOI: https://doi.org/10.1016/j.frl.2020.101698. URL: https://www.sciencedirect.com/science/article/pii/S1544612320308886 (cit. on p. 3).

Szabo, Nick (Sept. 1997). "Formalizing and Securing Relationships on Public Networks". In: *First Monday* 2.9. DOI: 10.5210/fm.v2i9.548. URL: https://journals.uic.edu/ojs/index.php/fm/article/view/548 (cit. on p. 7).

Torres, Christof Ferreira, Ramiro Camino, and Radu State (Aug. 2021). "Frontrunner Jones and the Raiders of the Dark Forest: An Empirical Study of Frontrunning on the Ethereum Blockchain". In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, pp. 1343–1359. ISBN: 978-1-939133-24-3. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/torres (cit. on p. 9).

TrueBlocks Team (Feb. 2022). *TrueBlocks: Lightweight indexing for any EVM-based blockchain*. URL: https://trueblocks.io/ (cit. on p. 15).

Victor, Friedhelm and Andrea Marie Weintraud (Apr. 2021). "Detecting and Quantifying Wash Trading on Decentralized Cryptocurrency Exchanges". In: *Proceedings of the Web Conference 2021*. DOI: 10.1145/3442381.3449824. URL: http://dx.doi.org/10.1145/3442381.3449824 (cit. on p. 3).

Vogelsteller, Fabian and Vitalik Buterin (Nov. 2015). *EIP-20: Token Standard*. Ethereum Improvement Proposal. URL: https://eips.ethereum.org/EIPS/eip-20 (cit. on p. 13).

Wang, Dabao, Siwei Wu, Ziling Lin, Lei Wu, Xingliang Yuan, Yajin Zhou, Haoyu Wang, and Kui Ren (May 2021a). "Towards A First Step to Understand Flash Loan and Its Applications in DeFi Ecosystem". In: *Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing*. ACM. DOI: 10.1145/3457977.3460301. URL: https://doi.org/10.1145%2F3457977.3460301 (cit. on p. 55).

Wang, Ye, Yan Chen, Shuiguang Deng, and Roger Wattenhofer (2021b). *Cyclic Arbitrage in Decentralized Exchange Markets*. arXiv: 2105.02784 [q-fin.TR] (cit. on pp. 3, 17, 20).

Wood, Gavin (2014). *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. Yellow paper. URL: http://gavwood.com/Paper.pdf (cit. on pp. 6, 40, 46).

Worsley, Nathan (2022). "MEV as an inner experience". MEV.Day at Devconnect Amsterdam. URL: https://www.youtube.com/watch?v=9iHlyaRsgYI (cit. on p. 24).

Xu, Jiahua, Krzysztof Paruch, Simon Cousaert, and Yebo Feng (2021). *SoK: Decentralized Exchanges (DEX) with Automated Market Maker (AMM) Protocols*. DOI: 10.48550/ARXIV.2103.12732. URL: https://arxiv.org/abs/2103.12732 (cit. on p. 12).

Zhang, Yi, Xiaohong Chen, and Daejun Park (2018). "Formal specification of constant product (xy=k) market maker model and implementation". In: *White paper* (cit. on p. 11).

Zhou, Liyi, Kaihua Qin, Antoine Cully, Benjamin Livshits, and Arthur Gervais (2021a). "On the Just-In-Time Discovery of Profit-Generating Transactions in DeFi Protocols". eng. In: *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, pp. 919–936. ISBN: 1728189349 (cit. on pp. 8, 18).

Zhou, Liyi, Kaihua Qin, Christof Ferreira Torres, Duc V Le, and Arthur Gervais (May 2021b). "High-Frequency Trading on Decentralized On-Chain Exchanges". In: *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 428–445. DOI: 10.1109/SP40001.2021.00027 (cit. on pp. 8, 11).

A Ethereum state transition

Formally, the transition to the next global state can be described with the following set of equations (Wood, 2014),

$$\sigma_{t+1} \equiv \Upsilon(\sigma_t, T_i) \tag{12}$$

$$\Sigma_{b+1} \equiv \Pi(\Sigma_b, B_{b+1}) \tag{13}$$

$$B_{b+1} \equiv (B_H, B_T, B_U) \tag{14}$$

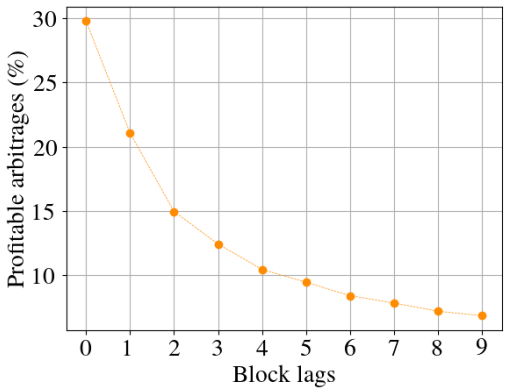
$$B_T \equiv (T_0, T_1, \ldots, T_I) \tag{15}$$

where Υ is the Ethereum state transition function operating on a transaction-level basis. T_i, $i = 0, \ldots, I$, is a valid transaction and σ_{t+1} is the state at transaction time $t + 1$. Π is the block level state transition function, Σ_{b+1} is the global block state and B_{b+1} is a block in block time $b + 1$. B_{b+1} contains valid transactions B_T and the block information B_H and B_U, called headers, containing important metadata about the current and previous blocks.[29] The global state is a mapping between addresses and account states and is updated each time a new block is added to the blockchain. Importantly, Equation 12 shows that each valid transaction affect the Ethereum Virtual Machine state sequentially, indicating that the blockchain state changes several times within each block.

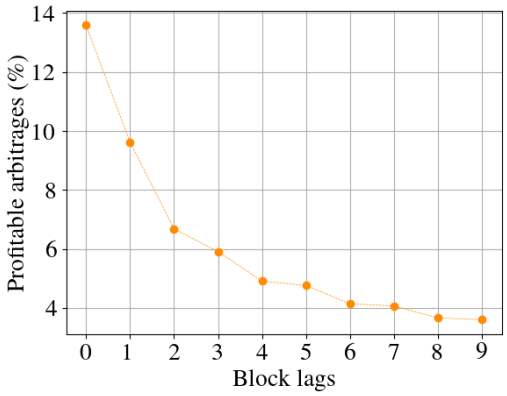
[29] See Appendix C for a full description of the block data.

B Robustness analysis

4.47% of the arbitrage transactions are executed at block position 0. These transactions are removed in the simulations presented in Figures 9a and 9b.



(a) Net profits with original transaction costs, where the arbitrage transactions at position 0 are removed.



(b) Net profits with updated transaction costs, where the arbitrage transaction at position 0 are removed.

Figure 9: Counter factual simulation with 9 lags.

Table 7: Probit estimation: Arbitrage trades and non-arbitrage trades regressed on sum of previous pirce changes.

	(1) All	(2) July 2020 - Nov 2020	(3) Dec 2020 - April 2021	(4) May 2021 - Sep 2021	(5) Oct 2021 - Feb 2022
$PrevTrans_i$	0.0470***	1.608***	0.551***	0.0718***	0.0261***
	(0.00188)	(0.0270)	(0.0165)	(0.00472)	(0.00204)
$SameBlock_i$	0.000283	0.162***	0.592***	0.315***	-0.0499***
	(0.0100)	(0.0385)	(0.0428)	(0.0385)	(0.0111)
$PrevBlock_{i,1}$	0.0699***	0.801***	0.345***	0.0794***	0.0246***
	(0.00352)	(0.0332)	(0.0132)	(0.00701)	(0.00423)
$PrevBlock_{i,2}$	0.0259***	0.0195***	0.217***	0.0797***	0.0123***
	(0.00288)	(0.00526)	(0.0152)	(0.0102)	(0.00363)
$PrevBlock_{i,3}$	0.0250***	0.0127*	0.0524***	0.181***	0.0179***
	(0.00381)	(0.00606)	(0.0112)	(0.0216)	(0.00538)
$PrevBlock_{i,4}$	0.0115**	0.0103	0.110***	0.00922	-0.00629
	(0.00436)	(0.00649)	(0.0314)	(0.00585)	(0.0213)
$PrevBlock_{i,5}$	-0.0124	-0.0484	0.0128	-0.00203	-0.0948***
	(0.00800)	(0.0389)	(0.0290)	(0.00892)	(0.0243)
$PrevBlock_{i,6}$	-0.00993	-0.174***	0.00940	0.0856**	-0.0370
	(0.0105)	(0.0393)	(0.0138)	(0.0272)	(0.0239)
$PrevBlock_{i,7}$	-0.00938*	-0.00541	0.0839*	-0.00259	-0.00894
	(0.00429)	(0.0249)	(0.0360)	(0.0116)	(0.00470)
$PrevBlock_{i,8}$	-0.0112***	-0.142***	0.0554	0.0151	-0.00867**
	(0.00276)	(0.0301)	(0.0436)	(0.0295)	(0.00277)
$PrevBlock_{i,9}$	-0.0722***	-0.341***	0.135***	0.0243	-0.0627***
	(0.0131)	(0.0464)	(0.0386)	(0.0321)	(0.0172)
$PrevBlock_{i,10}$	-0.00510	0.00170	-0.0688*	0.0799*	-0.0372
	(0.00786)	(0.00837)	(0.0348)	(0.0372)	(0.0240)
Constants	0.495***	0.351***	0.548***	0.608***	0.433***
	(0.000741)	(0.00146)	(0.00129)	(0.00135)	(0.00229)
N	463290	114701	154408	133478	47987

Standard errors in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Table 8: OLS estimation: Arbitrage net profits regressed on max of previous price changes without controls.

	(1) All	(2) July 2020 - Nov 2020	(3) Dec 2020 - April 2021	(4) May 2021 - Sep 2021	(5) Oct 2021 - Feb 2022
$PrevTrans_i$	19.94***	262.1***	1060.8***	17.25*	6.153*
	(2.409)	(20.08)	(23.44)	(7.102)	(2.869)
$SameBlock_i$	391.0***	539.8***	593.0***	446.5***	140.3**
	(20.56)	(42.44)	(68.14)	(29.72)	(44.81)
$PrevBlock_{i,1}$	10.12**	517.6***	21.08*	14.59	2.996
	(3.583)	(30.84)	(9.105)	(9.308)	(4.849)
$PrevBlock_{i,2}$	4.323	-1.540	92.48***	15.33	-0.872
	(3.576)	(3.456)	(21.02)	(11.77)	(5.148)
$PrevBlock_{i,3}$	0.475	0.978	3.061	-3.770	-1.357
	(4.503)	(4.315)	(8.317)	(23.03)	(8.228)
$PrevBlock_{i,4}$	5.257	2.430	10.43	4.096	9.341
	(6.758)	(4.613)	(51.09)	(12.45)	(31.57)
$PrevBlock_{i,5}$	105.0***	24.06	756.6***	-23.61	43.26
	(20.29)	(43.36)	(63.19)	(34.17)	(37.59)
$PrevBlock_{i,6}$	13.78	11.71	-35.50	8.235	24.33
	(15.19)	(34.28)	(22.69)	(47.57)	(30.70)
$PrevBlock_{i,7}$	9.073	-31.48	84.17	-9.340	-7.173
	(15.69)	(25.86)	(54.89)	(33.86)	(24.49)
$PrevBlock_{i,8}$	53.37**	46.59	265.0***	23.40	11.75
	(18.21)	(53.66)	(67.82)	(43.58)	(24.60)
$PrevBlock_{i,9}$	10.70	106.1*	-81.01	61.53	1.840
	(10.82)	(52.35)	(70.12)	(44.51)	(13.10)
$PrevBlock_{i,10}$	5.112	-0.0583	-85.51	-0.374	4.619
	(8.725)	(4.866)	(67.25)	(33.74)	(40.58)
Constant	52.32***	28.23***	58.69***	27.73***	63.22***
	(1.375)	(1.762)	(2.366)	(2.574)	(4.845)
N	231639	42211	86830	81754	22278

Standard errors in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Table 9: OLS estimation: Arbitrage net profits regressed on sum of previous price changes with control variables.

	(1) All	(2) July 2020 - Nov 2020	(3) Dec 2020 - April 2021	(4) May 2021 - Sep 2021	(5) Oct 2021 - Feb 2022
$PrevTrans_i$	19.95***	264.8***	1059.9***	16.94*	6.229*
	(2.407)	(20.06)	(23.39)	(7.096)	(2.869)
$SameBlock_i$	811.8***	753.9***	741.2***	1611.0***	168.3**
	(32.37)	(48.26)	(77.20)	(63.29)	(58.00)
$PrevBlock_{i,1}$	13.61**	571.3***	27.39	18.90	4.524
	(4.727)	(32.62)	(18.08)	(10.49)	(6.298)
$PrevBlock_{i,2}$	10.96**	-0.295	257.8***	19.94	-0.765
	(3.909)	(3.720)	(21.36)	(13.93)	(5.609)
$PrevBlock_{i,3}$	-4.082	1.098	3.992	-24.85	-3.255
	(5.134)	(4.314)	(15.36)	(33.12)	(8.223)
$PrevBlock_{i,4}$	1.575	2.137	53.98	1.342	-4.383
	(5.948)	(4.603)	(47.46)	(9.174)	(37.84)
$PrevBlock_{i,5}$	129.0***	43.59	415.6***	-48.89	67.39
	(25.03)	(42.18)	(49.33)	(41.54)	(71.21)
$PrevBlock_{i,6}$	21.10	38.26	-44.56	-53.11	72.38
	(25.00)	(37.99)	(73.95)	(43.88)	(51.11)
$PrevBlock_{i,7}$	1.598	-13.27	23.41	-11.09	-47.37
	(21.59)	(20.38)	(52.17)	(60.46)	(46.53)
$PrevBlock_{i,8}$	24.43	7.055	157.8*	-82.81	11.96
	(19.80)	(61.12)	(66.42)	(47.90)	(26.80)
$PrevBlock_{i,9}$	46.49*	106.4	30.63	47.08	6.137
	(22.37)	(54.54)	(57.76)	(64.46)	(31.80)
$PrevBlock_{i,10}$	5.383	-0.249	-34.76	-7.858	6.919
	(10.64)	(5.943)	(62.36)	(57.30)	(39.19)
$SameBlock_i$ Block time	0.187	0.146	0.375*	-0.0781	-0.144
	(0.108)	(0.133)	(0.184)	(0.198)	(0.355)
$PrevBlock_{i,1}$ Block time	0.229*	0.0662	-0.238	0.776***	-0.0716
	(0.109)	(0.130)	(0.181)	(0.201)	(0.394)
$PrevBlock_{i,2}$ Block time	0.0558	0.0542	0.000558	-0.113	0.681
	(0.109)	(0.130)	(0.180)	(0.203)	(0.409)
Constant	44.57***	23.72***	56.18***	17.48***	57.92***
	(2.832)	(3.508)	(4.842)	(5.230)	(9.310)
N	231639	42211	86830	81754	22278

Standard errors in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Table 10: OLS estimation: Arbitrage net profits regressed on sum of previous price changes without controls.

	(1) All	(2) July 2020 - Nov 2020	(3) Dec 2020 - April 2021	(4) May 2021 - Sep 2021	(5) Oct 2021 - Feb 2022
$PrevTrans_i$	19.95***	265.6***	1060.2***	17.22*	6.192*
	(2.407)	(20.04)	(23.38)	(7.096)	(2.869)
$SameBlock_i$	812.5***	754.8***	743.1***	1611.2***	168.2**
	(32.37)	(48.25)	(77.20)	(63.29)	(58.00)
$PrevBlock_{i,1}$	13.76**	573.5***	27.83	18.99	4.567
	(4.727)	(32.56)	(18.08)	(10.49)	(6.296)
$PrevBlock_{i,2}$	11.01**	-0.305	257.4***	20.72	-0.763
	(3.909)	(3.719)	(21.36)	(13.93)	(5.607)
$PrevBlock_{i,3}$	-4.089	1.125	4.035	-25.83	-2.997
	(5.134)	(4.314)	(15.36)	(33.11)	(8.222)
$PrevBlock_{i,4}$	1.613	2.158	53.19	1.352	-4.599
	(5.948)	(4.603)	(47.46)	(9.175)	(37.84)
$PrevBlock_{i,5}$	128.7***	43.57	415.6***	-49.77	67.79
	(25.03)	(42.18)	(49.33)	(41.54)	(71.21)
$PrevBlock_{i,6}$	21.23	37.85	-44.56	-53.12	72.76
	(25.00)	(37.99)	(73.95)	(43.89)	(51.10)
$PrevBlock_{i,7}$	1.839	-13.14	23.81	-11.03	-46.30
	(21.59)	(20.38)	(52.17)	(60.47)	(46.52)
$PrevBlock_{i,8}$	24.54	7.423	157.3*	-82.74	12.09
	(19.80)	(61.12)	(66.42)	(47.90)	(26.80)
$PrevBlock_{i,9}$	46.60*	106.3	31.05	47.33	5.735
	(22.37)	(54.53)	(57.76)	(64.47)	(31.80)
$PrevBlock_{i,10}$	5.439	-0.254	-34.89	-8.218	7.060
	(10.64)	(5.943)	(62.36)	(57.30)	(39.19)
Constant	50.75***	27.19***	57.99***	25.13***	63.40***
	(1.379)	(1.760)	(2.380)	(2.575)	(4.841)
N	231639	42211	86830	81754	22278

Standard errors in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

C Data

This appendix describes the on-chain Ethereum data used in this paper in detail. Ethereum on-chain data is structured into four tables `blocks`, `transactions`, `receipts` and `traces`, and can be accessed through a JSON RPC API. The Ethereum protocol specify a number of necessary fields for blocks, transactions and receipts (Wood, 2014). In addition, trace logs are outputs from the Ethereum Virtual Machine that consist of additional information about the transactions. Depending on what Ethereum client software is used to query the data, the output can vary slightly. The sections C.1, C.2, C.3 and C.4 show block, transaction, receipt and trace call responses from an Erigon (Erigon Team, 2022) archive node.[30][31] The output descriptions have been compiled from the official Ethereum documentation, OpenEthereum's documentation, and Wood (2014). Unnecessary verbose outputs are, at places, replaced with ....

C.1 Block data

Listing 1 shows the response from the archive node when calling the function `eth_getBlockByHash`. Description of the output data,

- `baseFeePerGas`: A scalar value equal to the minimum fee per gas required to be included in the block.[32]
- `difficulty`: A scalar value corresponding to the difficulty level of this block. This can be calculated from the previous block's difficulty level and the timestamp.
- `extraData`: An arbitrary byte array containing data relevant to this block.
- `gasLimit`: A scalar value equal to the current limit of gas expenditure per block.
- `gasUsed`: A scalar value equal to the total gas used in transactions in this block.
- `hash`: The Keccak 256-bit hash of this block's header.
- `logsBloom`: The Bloom filter composed from indexable information (logger address and log topics) contained in each log entry from the receipt of each transaction in the transaction list.

[30] The archive node is running on a Debian 11.2 machine with AMD Ryzen 5 1600 (6-core, 3.2GHz), 64GB of RAM and 4TB SSD.

[31] Example transaction hash: 0x0e5e386a2e3a80f1843f6520ebe2f0f118fd1939b36d8a3c00e2e90d2c88df8e.

[32] This is only present in `type 2` transactions after the implementation of EIP-1559 in the London Hard Fork 2021-08-05.

- `miner`: The 160-bit address to which the fees collected from the successful mining of this block be transferred.
- `mixHash`: A 256-bit hash which proves combined with the nonce that a sufficient amount of computation has been carried out on this block.
- `nonce`: A 64-bit hash which proves combined with the mix-hash that a sufficient amount of computation has been carried out on this block.
- `number`: A scalar value equal to the number of ancestor blocks. The genesis block has a number of zero.
- `parentHash`: The Keccak 256-bit hash of the parent block's header.
- `receiptsRoot`: The Keccak 256-bit hash of the root node of the Merkle Patricia tree structure populated with the receipts of each transaction in the transaction list portion of the block.
- `sha3Uncles`: The Keccak 256-bit hash of the uncles list portion of this block.
- `size`: A scalar value equal to the size of the block.
- `stateRoot`: The Keccak 256-bit hash of the root node of the state Merkle Patricia tree, after all transactions are executed and finalized.
- `timestamp`: A scalar value equal to the reasonable output of Unix's time() at this block's inception.
- `totalDifficulty`: A scalar value equal to the total difficulty of the chain until this block.
- `transactions`: A list of Keccak 256-bit hashes of the transactions included in this block.
- `transactionsRoot`: The Keccak 256-bit hash of the root node of the Merkle Patricia tree structure populated with each transaction in the transaction list portion of this block.
- `uncles`: A list of Keccak 256-bit hashes of the uncle blocks.

```
1 {
2   "difficulty": "0xc4bbf8674df01",
3   "extraData": "0x307834383639373636353666366532303530366636663663",
4   "gasLimit": "0xbe150c",
```

```
  "gasUsed": "0xbbd420",
  "hash": "0xd85f9b3690a8aca172d096a408024c12da45eb4621e08982eaf886f1d12f5d
     49",
  "logsBloom": "0xdfe041d475201950871933f0a87d5da05a28b2980014c3ec829dd10a7
     aa24a1454803c5d660542a6c22133663906cdd546d934080aa698f2ab981a70db2a4
     dad11131500c7ce6303f82c04bd18214ce15ad2095f23480d5458cdd4ea9175d10176
     1408849a0ec5b88031830c02268e3dcfe414221e648dc6032d5c92f6a8fc627e04b31
     787792426df52f560a8a38e0bc003d4816ffd9cfbf911f5ef065dc8d7831e1640707c
     61da0df797ac0528b183d3a100018ac06a61a1170c009c2ad28140d8e86ae1b406303
     e846a688f6d85dc04088ec1c0fa443009327343a606b00da098359ca2218540352567
     8cd5911a9d66758715b0da2954193d2707ba360a84",
  "miner": "0x1ad91ee08f21be3de0ba2ba6918e714da6b45836",
  "mixHash": "0xf58be2dacfb26108447da3d7809e44829fe35d9ac0bdba9115d3a41364
     bfa29c",
  "nonce": "0x21f0257c209b32dd",
  "number": "0xacee03",
  "parentHash": "0xe424fb2b560b5c7d405dacf2b92cee2dfc89726b365e10b034d776d7
     b1a16365",
  "receiptsRoot": "0xcf2a24c67500957ccf0faeff4dc0d3b268062d898e084d1f062867
     442cb887e5",
  "sha3Uncles": "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd
     40d49347",
  "size": "0xa839",
  "stateRoot": "0x13649aca35de0a40e1b1f53c50eab6b2766013877c8474a7bfd8b1b81
     350f53b",
  "timestamp": "0x5fbf71b6",
  "totalDifficulty": "0x4045661d0d677d859d1",
  "transactions": [
    "0xa988d729ecb71c6402fbb893cb696e35f32b9a257eba0fc4be77adad443832bd",
    "0x0af408473617105f24ed80117a267315eeadc65048fce857cf52529419629e3d",
    "0x578d168a72a9a054f89155a5c38d64401a517f5fb46a64fc77ee796873205541",
    ...
    "0xc4713088a14c4be8954d03083bae9a28280ef55b4001005b72df0eaf22ffc87c",
    "0x4eb4d31fdf54adab612ed64c0ab836bf0b9ab2f6d87a7e3b0f736e125f947133",
    "0x4c2531e6dccb65e3d64f9f41ebb1dd4f86bf9b6aceadeecef669df417899ada2"
  ],
  "transactionsRoot": "0xdbc7459cf1eb23471bdedf2cb02a9140d4e2e1956b4366316b
     1b83b82aeb4a8c",
  "uncles": []
}
```


Listing 1: Erigon archive node block response.

C.2 Transaction data

Listing 2 shows the response from the archive node when calling the function `eth_getTransactionByHash`. Description of the output data,

- `blockHash`: The Keccak 256-bit hash of the block's header this transaction is included in.
- `blockNumber`: A scalar value equal to the block's number this transaction is included in.
- `from`: The 160-bit address of the sender.
- `gas`: A scalar value equal to the maximum amount of gas units that can be consumed by the transaction.
- `gasPrice`: A scalar value specifying the gas price provided by the sender in wei.
- `maxPriorityFeePerGas`: A scalar value equal to the maximum amount of gas to be included as a tip to the miner.[33]
- `maxFeePerGas`: A scalar value equal to the maximum amount of gas to be paid.[34]
- `hash`: The Keccak 256-bit hash of this transaction.
- `input`: An unlimited size byte array specifying the EVM-code for the account initialisation procedure.
- `nonce`: A scalar value equal to the number of transactions sent by the sender.
- `to`: The 160-bit address of the message call's recipient.
- `transactionIndex`: A scalar value equal to this transactions' position in the block.
- `value`: A scalar value equal to the number of Wei to be transferred to the message call's recipient or, in the case of a contract creation, as an endowment to the newly created account.
- `type`: A scalar value indicating transaction type (0 for legacy transaction and 2 for transaction type after EIP-1559).
- `accessList`: Optional list of addresses and storage keys that the transaction plans to access.

[33] See footnote 32.
[34] See footnote 32.

- `chainId`: A scalar value indicating which chain this transaction is on (1 for Ethereum Mainnet).
- `v`, `r` and `s`: Values corresponding to the signature of the transaction and used to determine the sender of the transaction.


```
{
  "blockHash": "0xd85f9b3690a8aca172d096a408024c12da45eb4621e08982eaf886f1d
     12f5d49",
  "blockNumber": "0xacee03",
  "from": "0x000000007cb2bd00ae5eb839930bb7847ae5b039",
  "gas": "0x4c959",
  "gasPrice": "0x1ddc4aadade",
  "hash": "0x0e5e386a2e3a80f1843f6520ebe2f0f118fd1939b36d8a3c00e2e90d2c88df
     8e",
  "input":  "0x00000000000000000000000000000000000000000000000000000000000000
     0280000000000000000000000000000000000000000000000000000000000000d4c20
     000000000000000000000000000000b3f879cb30fe243b4dfee438691c...882f0
     000000000000000000000000000000000000000001662e93021bfb0ca856e1000000
     000000000000000000000000000000000c5beefbbfa5688",
  "nonce": "0x37b8",
  "to": "0x00000000000080c886232e9b7ebbfb942b5987aa",
  "transactionIndex": "0xd",
  "value": "0x0",
  "type": "0x0",
  "v": "0x26",
  "r": "0x83de603b9714fbd2b5446b9061bedd5bf8ba4868567595d82022378ae700054f"
     ,
  "s": "0x708e255f2a98120084267dbce82c739c6c1275a03c10f4c6cd1d7e4c5d361730"
}
```


Listing 2: Erigon archive node transaction response.

C.3 Receipt data

Listing 3 shows the response from the archive node when calling the function `eth_getTransactionReceipt`. Description of the output data,

- `blockHash`: The Keccak 256-bit hash of the block's header this transaction is included in.
- `blockNumber`: A scalar value equal to the block's number this transaction is included in.

- `contractAddress`: The Keccak 256-bit hash of the address if a contract was created, otherwise `null`.
- `cumulativeGasUsed`: A scalar value equal to the total amount of gas used when this transaction was executed in the block.
- `effectiveGasPrice`: A scalar value equal to the gas price used by the transaction.
- `from`: The 160-bit address of the sender.
- `gasUsed`: A scalar value equal to the gas used by the transaction.
- `logs`: A list of log objects.
  - `address`: The 160-bit address to the contract emitting the event.
  - `topics`: An array of Keccak 256-bit hashes of contract functions including arguments.
  - `data`: Byte array specifying the arguments for the contract function called.
  - `blockNumber`: A scalar value equal to the block's number this transaction is included in.
  - `transactionIndex`: A scalar value equal to this transactions' position in the block.
  - `blockHash`: The Keccak 256-bit hash of the block's header this transaction is included in.
  - `logIndex`: A scalar value equal to this logs' position in logs.
  - `removed`: Boolean indicating if the log was removed in a reorg.
- `logsBloom`: The Bloom filter composed from indexing information.
- `status`: A scalar value indicating if the transaction was successfully mined.
- `to`: The 160-bit address of the message call's recipient.
- `transactionHash`: The Keccak 256-bit hash of this transaction.
- `transactionIndex`: A scalar value equal to this transactions' position in the block.
- `type`: A scalar value indicating transaction type (0 for legacy transaction and 2 for transaction type after EIP-1559).

```
{
  "blockHash": "0xd85f9b3690a8aca172d096a408024c12da45eb4621e08982eaf886f1d
     12f5d49",
  "blockNumber": "0xacee03",
  "contractAddress": null,
  "cumulativeGasUsed": "0x172b16",
  "effectiveGasPrice": "0x1ddc4aadade",
  "from": "0x000000007cb2bd00ae5eb839930bb7847ae5b039",
  "gasUsed": "0x197a4",
  "logs": [
    {
      "address": "0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2",
      "topics": [
        "0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef
           ",
        "0x0000000000000000000000000a478c2975ab1ea89e8196811f51a7b7ade33eb11
           ",
        "0x000000000000000000000000fbc312fa3b5be4e7631db2901ae7e0e79a764c9b
           "
      ],
      "data": "0x00000000000000000000000000000000000000000000000d68fba3c3c11
         e53108",
      "blockNumber": "0xacee03",
      "transactionHash": "0x0e5e386a2e3a80f1843f6520ebe2f0f118fd1939b36d8a3
         c00e2e90d2c88df8e",
      "transactionIndex": "0xd",
      "blockHash": "0xd85f9b3690a8aca172d096a408024c12da45eb4621e08982eaf88
         6f1d12f5d49",
      "logIndex": "0x39",
      "removed": false
    },
    ...
    }
  ],
  "logsBloom": "0x00200000402000000000001080010000000000000000000000000000
     00000000000000000000000000000020000000020000008000000000000000000000
     00000000002000001000000080000002000000000000000000000000000000000100
     00000000800000000000000000000000000000000000000000010000000000000000
     00000200000000000000000000000000001000080000004100000000000000000000
     00000000000000000000000000000000000020000000000000000420004000000
     00000004000000000020000000100000000000000004000020000000000000000000
     40000000000000000008000000000000000000",
  "status": "0x1",
```

```
  "to": "0x00000000000080c886232e9b7ebbfb942b5987aa",
  "transactionHash": "0x0e5e386a2e3a80f1843f6520ebe2f0f118fd1939b36d8a3c00e
      2e90d2c88df8e",
  "transactionIndex": "0xd",
  "type": "0x0"
}
```

Listing 3: Erigon archive node receipt response.

C.4 Trace call data

Listing 4 shows the response from the archive node when calling the function `trace_replayTransaction`. Description of the output data,

- `output`: String.
- `stateDiff`: Array.
- `trace`:
  - `action`:
    - `from`: The 160-bit address of the trace initiator.
    - `callType`: String of the type of call.
    - `gas`: Maximum number of gas allowed for this trace.
    - `input`: Byte array of data specifying the EVM-code for the action.
    - `to`: The 160-bit address to the trace recipient.
    - `value`: A scalar value equal to the number of Wei to be transferred to the recipient.
  - `result`:
    - `gasUsed`: Gas used by the trace.
    - `output`: Byte array of the result of the call for this trace.
  - `subtraces`: The number of children traces.
  - `traceAddress`: Array of a particular trace address in the trace tree.
  - `type`: String of the type of trace.
- `vmTrace`: VmTrace object.

```
{
  "output": "0x0000000000000000000000000000000000000000000000000000000000000
      0004",
```

```
  "stateDiff": null,
  "trace": [
      ...,
      {
      "action": {
        "from": "0x00000000000080c886232e9b7ebbfb942b5987aa",
        "callType": "call",
        "gas": "0x35b60",
        "input": "0x022c0d9f0000000000000000000000000000000000000000000000000000
            0000000000000000000000000000000000000000000000000000000000000000
            d68fba3c3c11e5310800000000000000000000000fbc312fa3b5be4e7631db
            2901ae7e0e79a764c9b0000000000000000000000000000000000000000000000
            0000000000000000080000000000000000000000000000000000000000000000
            0000000000000000080000000000000000000000c02aaa39b223fe8d0a
            0e5c4f27ead9083c756cc2000000000000000000000000000000000000000000
            0000010d0ca8041bb5d3108000000000000000000000c3d03e4f041fd4cd
            388c549ee2a29a9e5075882f0000000000000000000000000000000000000000
            00001662e93021bfb0ca856e1",
        "to": "0xa478c2975ab1ea89e8196811f51a7b7ade33eb11",
        "value": "0x0"
      },
      "result": {
        "gasUsed": "0x209dc",
        "output": "0x"
      },
      "subtraces": 4,
      "traceAddress": [
        2
      ],
      "type": "call"
    }
    ...,
  ],
  "vmTrace": null
}
```

Listing 4: Erigon archive node trace response.

D Empirical classification

The arbitrage classification consists of three parts: Detecting arbitrage transactions, detecting sandwich arbitrage bundles, and detecting transactions using flash swaps. This appendix describes the empirical classification strategy with more technical details.

D.1 Arbitrage detection

To detect arbitrage transactions the following process is used,

1. Necessary swap actions:
   (a) At least two `Swap` events are emitted.
   (b) All `Swap` events must form a loop, the input asset and amount of any swap action must be the output asset and amount of the previous action.
   (c) The input asset of the first swap action and the output asset of the last swap action must be the same, closing the loop.
2. Atomic transaction: All swap actions must be included in a single transaction.
3. Pure arbitrage:
   (a) The transaction receipt log should only contain `Transfer`, `Sync` and `Swap` events, ensuring that nothing other than DEX trading takes place in the transaction.
   (b) The transaction need to be profitable.
   (c) The transaction need to pay a non-zero fee to the miner.
   (d) Flash swap transactions were classified and then removed using the following conditions (Wang et al., 2021a):
      i. The length of the parameter `data` in the transaction's `trace` is greater than zero.
      ii. The internal transaction triggered by `uniswapV2Call` must include the invocation of `transfer` or `transferFrom` function.
      iii. The receiver address of `transfer` or `transferFrom` function must be the pair contract.
   (e) Sandwich bundles were classified and then removed using the following criterion (Qin, Zhou, and Gervais, 2021):
      i. The transactions must be executed by the same address.

ii. The transactions must be in the same block and their transaction positions must be within one step from each other.

iii. The transactions' `swap` events must include the same tokens and trade in opposite directions.

iv. There must be one other transaction in between the transactions trading at least one currency pair of the transactions.

4. Simple arbitrage: A token pair should at most occur in two `Swap` events, ensuring that only one arbitrage trade is executed per transaction.