Suboptimality in DeFi

AVIV YAISH, The Hebrew University, Israel
MAYA DOTAN, The Hebrew University, Israel
KAIHUA QIN, Imperial College London and Berkeley Center for Responsible, Decentralized Intelligence (RDI)
AVIV ZOHAR, The Hebrew University, Israel
ARTHUR GERVAIS, University College London and Berkeley Center for Responsible, Decentralized Intelligence (RDI)

The decentralized finance (DeFi) ecosystem has proven to be immensely popular in facilitating financial operations such as lending and exchanging assets, with Ethereum-based platforms holding a combined amount in excess of 30 billion USD. The public availability of these platforms' code, together with real-time data on all user interactions with them has given rise to complex automatic tools that recognize and seize profit opportunities on behalf of users.

In this work, we formalize core DeFi primitives which are responsible for a daily volume of over 100 million USD in Ethereum-based platforms alone: (1) using *flash*swaps to close arbitrage opportunities between cryptocurrency exchanges, (2) lending and borrowing funds, (3) liquidation of insolvent loans using flashswaps. The profit which can be made from each primitive is then cast as an optimization problem that can be solved.

We use our formalization to analyze several case studies for these primitive, showing that popular platforms and tools which promise to automatically optimize profits for users, actually fall short. In specific instances, the profits can be increased by more than 100%, with the highest amount of "missed" revenue by a single suboptimal action equal to 428.14 ETH, or roughly 517K USD.

Finally, we show that many missed opportunities to make a profit do not go unnoticed by other users. Indeed, suboptimal transactions are sometimes immediately followed by "trailing" back-running transactions which extract additional profits using similar actions. By analyzing a subset of these events, we uncover that users who frequently create such trailing transactions are heavily tied to specific miners, meaning that all of their transactions appear only in blocks mined by one miner in particular. As a portion of the backrun non-optimal transactions are private, we hypothesize that the users who create them are, in fact, miners (or users collaborating with miners) who use inside information known only to them to make a profit, thus gaining an unfair advantage. This is essentially an instance of miner-extractable value (MEV) "in action".

CCS Concepts: • **Applied computing** → **Digital cash**; • **General and reference** → *Empirical studies*; • **Security and privacy** → *Economics of security and privacy*.

Additional Key Words and Phrases: Cryptocurrency, Blockchain, DeFi, Miner Extractable Value

This research was supported by the Ministry of Science & Technology, Israel.
Authors' addresses: Aviv Yaish, aviv.yaish@mail.huji.ac.il, The Hebrew University, Jerusalem, Israel; Maya Dotan, mayadotan@mail.huji.ac.il, The Hebrew University, Jerusalem, Israel; Kaihua Qin, kaihua.qin@imperial.ac.uk, Imperial College London and Berkeley Center for Responsible, Decentralized Intelligence (RDI); Aviv Zohar, avivz@cs.huji.ac.il, The Hebrew University, Jerusalem, Israel; Arthur Gervais, arthur@gervais.cc, University College London and Berkeley Center for Responsible, Decentralized Intelligence (RDI).

1 INTRODUCTION

DeFi is an umbrella term for financial platforms that run as smart contracts on decentralized cryptocurrencies. DeFi platforms for example allow users to exchange tokens, to lend and borrow funds [68]. Popular DeFi platforms incentivize users to provide liquidity using lucrative interest-rate schemes [46]. A daily average of 66.7 billion US dollars worth of tokens was held by various DeFi platforms between the beginning of May and the end of July 2021, with interest rates offered by certain platforms exceeding 20%, both for lending and borrowing assets [63].

Users have readily exploited security holes and market inefficiencies to profit from DeFi [93], for example by finding arbitrage between exchanges, producing per-annum profits which are estimated to be over 1 Billion USD [93]. Unfortunately, miners, who are supposed to maintain the system's integrity, can use such transactions to increase their revenue by manipulating the consensus layer [20]; the potential value earned by these exploits is termed MEV.

While on-chain DeFi services appear similar in functionality to their off-chain equivalents, the optimal usage of each differs as a consequence of design considerations which are required for the secure and efficient implementation of financial services in the decentralized and pseudonymous setting which DeFi inhabits.

In this work, we take the first step towards characterizing optimal user behaviours in the DeFi setting. We focus on three primitives underlying common DeFi use-cases: (1) the usage of "flash" token swaps to profit off of arbitrage opportunities in decentralized exchanges (DEXs), (2) collateralized lending, and (3) the liquidation of insolvent loans using flashswaps.

(1) Flashswap-based arbitrage. If a single asset has different prices in two or more markets, an actor can take advantage of the price difference to make a profit by simultaneously buying and selling the asset in these markets. This practice is also known as *arbitrage*. The prevalence of various asset pricing mechanisms across different blockchain DEXs means that such price discrepancies exist between DEXs, too. Furthermore, some DEXs such as Uniswap allow users to perform *flashswaps*, meaning that users can swap an x amount of a token X for a y amount of another token Y *without* having the required liquidity of an x amount of X tokens up-front, as long as the exchange is repaid within the same transaction in which the flashswap was performed. Thus, users can perform arbitrage without requiring large amounts of initial funds, using the profits of the arbitrage instead.

The search space for arbitrage opportunities on the Ethereum blockchain is huge, owing to the large number of DEXs currently active on Ethereum, and the even larger amount of fungible assets which can be exchanged on them. Furthermore, this space should be traversed within a limited time-frame, as a new Ethereum block is mined every 12 seconds, meaning that arbitrage opportunities may be short-lived. We define an optimization problem for the task of finding and capitalizing on arbitrage opportunities using flashswaps, and show that blockchain users have performed suboptimally in the past. In particular, we find instances where missed profits reach more than 517K USD.

To prevent profitable transactions from being front-run, would-be arbitrageurs use *private relays* to submit transactions to miners. These are communication channels which claim to transmit transactions in a private manner, without disclosing or acting upon the contents of relayed transactions to any party besides the recipient. Popular relay services offer their services free-of-charge, piquing the community's interest with regards to their sources of revenue. Surprisingly, by analyzing our data we found circumstantial evidence suggesting that the relay service operated by the Ethermine mining pool has been using "inside information" gained from private transactions to trim the search space for finding profitable arbitrage opportunities, thereby gaining an advantage over others.

Table 1. A summary of our case studies.

Case	"Lost" Profits	Action Type
Case Study 2: $0x5e1\ldots 6c5$	More than 94%	Flashswap
Case Study 3: $0x0f4\ldots 74b$	More than 7000%	Flashswap
Case Study 5: Justin Sun	More than 8.7%	Lending
Case Study 6: $0x7a1\ldots 428$	More than 700%	Lending

(2) Collateralized lending. While in the off-chain setting a user with free funds and access to a positive-interest bearing instrument may be incentivized to invest them in their entirety, this can be suboptimal in the on-chain DeFi setting. In particular, popular collateralized lending platforms such as Aave and Compound set their interest-rates curves to be monotonically increasing in the *utilization* of the platforms' liquidity, meaning in the ratio between the amount of funds which are loaned-out and the funds which are deposited. Thus, users who withdraw a partial amount of deposited funds may *increase* their revenue. The increase in interest-rate stemming from such a withdrawal can be so large, that the interest accrued on the remaining sum can become even greater than the interest were the user was to leave its funds unchanged.

In our work, we generalize this observation and formulate an optimization problem which when solved, provides the optimal lending strategy in a one-shot myopic setting. We then use it to analyze the performance of several large investors and show it can be suboptimal by up to 700%. Furthermore, we go over empirical evidence which suggests that currently, market forces are slow to react to liquidity shocks, such as when a large amount of deposits is withdrawn in a single transaction. Thus, even myopically optimizing deposits can produce long-lasting profits.

(3) Liquidation of insolvent loans using flashswaps. Collateralized lending platforms allow users to repay under-collateralized debt positions and receive the underlying collateral at a discount. Platforms like Aave and Compound limit the amount of debt which can be repaid by some *close factor*, i.e., a close factor of 50% permits repaying up to half of the debt position. Popular tools that automatically look and act upon potentially profitable liquidation opportunities commonly repay the maximal amount of debt possible. Using flashswaps to fund liquidations is prevalent among these tools.

We show that this liquidation strategy is suboptimal. In fact, we provide a better strategy which shows that in certain cases it is more profitable to liquidate substantially less than the close factor permits. This is due to the effects that large swaps may have on the exchange-rate between the asset of the underlying debt and the collateral.

Our Contributions. Our paper's contributions to the literature can be summarized like so:

- **Optimal Execution**. We formalize three core DeFi primitives: flashswaps, collateralized lending and liquidating loans using flashswaps. The optimal execution of each is cast as an optimization problem which can be solved. To the best of our knowledge, we are the first to optimize these primitives and to formalize flashswaps.
- **Evaluation of Suboptimal Cases**. Our optimization problems are used to find and empirically evaluate multiple case studies showing the currently widespread suboptimal behavior of DeFi platform actors. We show that some suboptimal cases could be improved by more than 700%, with one case which could be improved to earn an additional amount of 428.14 ETH, amounting to about 517K USD (at the time the case happened). A summary of our findings is given in Table 1.

- **Longitudinal Study of Suboptimal Arbitrage Flashswaps**. We devised a heuristic to detect suboptimal arbitrage transactions which rely on flashswaps and found over 10K such instances, which combined have lost more than 4 Million USD due to suboptimality. In our study, we found circumstantial evidence which ties the Ethermine mining pool to a specific Ethereum address which capitalized on suboptimal transactions, suggesting that miners might be using or sharing "inside information" to find profitable trades, in spite of publicly committing not to do so. This is the *first* evidence of such behavior, to the best of our knowledge.
- **Impact of Suboptimality**. We show that tools and platforms which are used to manage over 400 Million USD, such as Yearn Finance [23], are suboptimal. Surprisingly, such suboptimality can benefit certain users. For example, borrowers benefit from the suboptimality of liquidity providers (LPs).

Paper structure. We go over required background on mining and cryptocurrencies in Section 2 and define the model used throughout the paper in Section 3. We present our formal analysis and case-studies on using flashswaps for arbitrage in Section 4, on collateralized lending in Section 5, and on using flashswaps for the liquidation of insolvent loans in Section 6. We conclude with a review of related work in Section 7 and a discussion on the implication of our results in Section 8.

2 BACKGROUND

We now go over the preliminaries required for our work.

2.1 Cryptocurrencies

Bitcoin [57] and Ethereum [12, 83] enable financial *transactions* between users. Transactions are collected in a ledger comprised of *blocks*, with each block specifying an order on transactions contained within it. In addition, each block points to a preceding block, thus the resulting data-structure, which is also called a *blockchain*, maintains order between transactions.

Most cryptocurrencies operate in a decentralized manner and rely on pseudonymous users called *miners* to create blocks, a process also called *mining*. Miners are compensated for their work with a *block-reward* which is offered to the miner of each valid block [88]. As blocks are limited in size, users compete for block-space and can offer *fees* to prioritize their transactions [44].

Computer programs can be stored on a blockchain and executed in a decentralized manner by cryptocurrency miners. Such programs are also called *smart contracts*, and the process for executing them in Ethereum is defined by a formal virtual-machine [45] execution model called the Ethereum virtual machine (EVM) [31, 47, 83].

2.2 Decentralized Finance

Smart contracts can be used to facilitate financial operations such as exchanging assets and taking and giving loans. Decentralized platforms that enable such financial operations and which are implemented as smart contracts are commonly referred to as DeFi platforms. As such platforms aspire to be decentralized, without relying on central authorities for financial backing, many platforms rely on user-provided liquidity for their operation, with funds collected in *liquidity pools*.

Lending Platforms. Certain DeFi platforms, like Aave and Compound, let users take and give loans [52, 84, 87]. Due to the pseudonymous nature of cryptocurrencies, it is impossible to assure borrowers will repay their debt without a form of collateral. Thus, platforms offer collateralized loans which are secured by up-front deposits that are at least equal in value to the loan taken. If the collateral loses in value relative to the loan, it is offered for liquidation [64].

Currently, DeFi lending platforms do not offer unsecured loans for periods longer than the span of a single transaction. If loans are taken and returned in such a manner, they can be verified to ensure that the loan taker has indeed returned all funds and paid any relevant fee. Transactions that do not satisfy these requirements are reverted [10], meaning that all changes it made to the EVM's state [48] are rolled back. To prevent spamming by users, reverted transactions still have to pay transaction fees. Thus, these loans can be taken without collateral, and are termed *flashloans* [20].

Decentralized Exchanges. DEXs are platforms that enable the exchange, or *swap*, of tokens [85]. Platforms like Uniswap [77] do not require that users interact to perform an exchange. Instead, swaps between a pair of tokens are performed using user-provided liquidity which is collected in *liquidity pools* by the platform, with swaps usually costing a proportional fee. Platforms which use automated mechanisms to determine exchange-rates between tokens are also known as automated market makers (AMMs).

It is common for platforms to collect liquidity for each token pair in a different pool. If a user wishes to swap using a specific platform between pairs which do not have a designated pool, it can do so by performing multiple swaps, also called *multi-hop* swaps, with certain platforms offering tools to facilitate such actions [74, 76].

ERC20 Tokens. Ethereum specifies a standard API that allows smart contracts to create and interact with fungible tokens. This standard is called Ethereum request for comments (ERC)20, and tokens that use it are called ERC20 tokens [30, 79]. These tokens are in use by various DeFi platforms, and are quite popular [15, 78].

In our work, we go over case studies which use such tokens, specifically the USDT and USDC tokens (which strive to maintain a one-to-one peg with the USD [56]), Wrapped Ethereum (WETH), a "wrapped" version of Ethereum which was designed to enable easier interaction between smart-contracts and the Ethereum token [13], Wrapped Bitcoin (WBTC), and the CRV token [19].

Many DeFi platforms create new ERC20 tokens for protocol-specific versions of each ERC20 token they support. For example, Compound created *cUSDT*, which is a platform-integrated version of the USDT token [17]. By interacting with cUSDT's smart contract, users can deposit and withdraw USDT to and from Compound.

3 MODEL

Notations are introduced as necessary, and all are summarized in Appendix B.

System Model. Our system consists of a blockchain which supports smart contracts, such as Ethereum. On top of this blockchain, there are at least two types of DeFi platforms implemented as smart contracts: (1) lending platforms, where users can lend or borrow funds, and (2) DEXs, which let users trade one currency for another. Together, these form an important part of the DeFi ecosystem. We assume that the lending platforms' interest-rates and the DEXs' exchange-rates are publicly available to all users.

User Model. We assume the existence of utility-maximizing users who can interact with these platforms. The action space of users consists of the following actions, which we define throughout the various sections: (1) Swapping one currency for another, (2) Taking and repaying loans and flashloans, and (3) Liquidating an insolvent position. All actions can be executed using single transactions, or be implemented in a smart contract that executes multiple actions in sequence. When users issue transactions, we assume that by paying a higher transaction fee these can be strategically placed between other transactions in a future block in a deterministic manner.

4 FLASHSWAP SUBOPTIMALITY

DEXs such as Uniswap let users exchange, or *swap* tokens in a decentralized manner. Currently, the leading DEX platforms on the Ethereum blockchain are Uniswap v2 [4], Uniswap v3 [5], Sushiswap [71] and Balancer [55], which together comprise roughly 75% of the daily volume of all token exchanges in Ethereum [21]. We note that Uniswap v1 is considered as a proof-of-concept by its creators [73], and thus was not included in our work.

Surprisingly, even sophisticated users such as cryptocurrency *swappers*, who are savvy enough to find extremely short-lived opportunities to make a profit by swapping multiple assets across different DEXs, do not always do so optimally.

4.1 Arbitrage in Decentralized Exchanges

DEXs often rely on different mechanisms and sources of information to determine the exchange-rates between tokens, thereby the same asset might have different prices in different DEXs. This can create *arbitrage* opportunities [70], where users can generate a net profit by trading across platforms with different exchange-rates [81, 91], these users are oftentimes referred to as *arbitrageurs*. The act of profiting from arbitrage is sometimes referred to as *closing* the arbitrage. We crystallize the notion of arbitrage in Definition 1.

Definition 1 (Arbitrage). *Profiting from the buying and selling of an asset across multiple markets which list different prices for it.*

Indeed, arbitrage opportunities are prevalent in today's DeFi ecosystem [20], with arbitrageurs making an average profit of more than 12 Million USD per month [81].

4.2 Flashswaps

Flashswaps [75] are a DeFi instrument implemented in most major DEXs in Ethereum. In fact, in Uniswap v2 and v3, as well as in Sushiswap, all swaps are implemented as flashswaps [4, 39, 73].

Flashswaps let users swap between two currencies without having the initial liquidity requirements upfront, and work similarly to flashloans. Generally, given that a user wants to obtain a c amount of token C to perform an action (e.g., perform arbitrage), it has to:

(1) Write and deploy a smart contract which has a function that can receive this amount and execute the wanted action. This function is also known as a *callback* function.
(2) Call the "flashswap" function of a DEX, providing c and the callback function as arguments.

Then, given that the DEX has more than a c amount of token C, it will transfer the requested token to the smart contract, execute the user's code, and afterwards verify that the code either returned the tokens in full, or repaid an equivalent amount of another token $\mathcal{D}$ according to the DEX's $C \leftrightarrow \mathcal{D}$ exchange-rate. Platforms can also ask for a predetermined fee which is equal to a fraction of the given tokens, for example 0.3% in Uniswap v2 [75]. If the user's callback function did not satisfy these conditions, the transaction is reverted, similarly to a flashloan (see Section 2).

To further illustrate how flashswaps work, we provide a real-world example of a flashswap which was used by an arbitrageur in Example 1, with a graphical depiction given in Fig. 1.

Example 1. *In transaction* $0x9ae\ldots bc4$*, the user with Ethereum address* $0x5e1\ldots 6c5$ *exploited an arbitrage opportunity between the Uniswap v2 and Uniswap v3 platforms, arising from different ETH to USDC exchange rates on the two platforms. We now go over this transaction step-by-step.*

First, the user obtained $3.6K$ *ETH from Uniswap v3 against a debt of* 3.9 *Million USDC by using a flashswap; the user pays this debt in its entirety using profits made from the arbitrage. Then,* $3.1K$ *ETH were swapped for* 3.9 *Million USDC on Unsiwap v2. Finally, the flashswap was repaid by transferring* 3.9 *Million USDC to Uniswap v3, leaving the user with a net profit of* 454 *ETH, translating to about* $600K$ *USD, according to the rate at the time.*

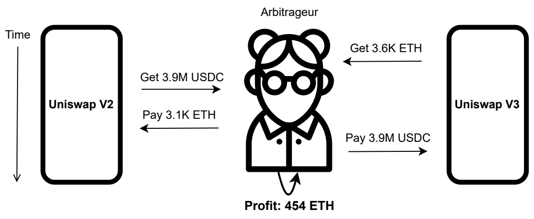


Fig. 1. A depiction of the arbitrage given in Example 1.

Notice that the arbitrageur does not need to have any amount of funds upfront, as the second swap was paid in full using the initial flashswap, while the initial flashswap was repaid using the profits obtained from the second swap, with all operations executed within the span of a single transaction.

4.3 Optimal Flashswap Arbitrage

We now formally define an optimization problem for the task of finding an arbitrage opportunity which can be closed using a single transaction that is comprised solely of flashloans.

Denote all tokens which exist in our economy by $C_1, \ldots, C_n$. For brevity, we may refer to tokens by their index, e.g., use i instead of C_i. For simplicity, we assume an arbitrageur wishes to maximize profits in some token i^*, and is willing to perform up to k_{end} swaps in a single transaction (decreasing k_{end} allows reducing the time required for finding a solution).

Denote by $c_{k,i}$ the amount of token i which is held (or owed) by the user at the end of the k-th step of the transaction made. Denote the set of all DEXs as S, and for each DEX $\mathbb{D} \in S$, denote the set of token-pairs which it allows to swap as $\text{TokenPairs}(\mathbb{D}) \overset{\text{def}}{=} \{(i, j) \mid i, j \text{ are valid token pair in } \mathbb{D}\}$. We notate the action of swapping a Δ amount of the i-th token in exchange for a δ amount of j tokens on DEX $\mathbb{D}$ with $\delta = \text{Swap}_{\mathbb{D}}^{i \to j}(\Delta)$, and any debt or fees incurred in the i-th token with $\phi_i = \text{Debt}\left(\text{Swap}_{\mathbb{D}}^{i \to j}(\Delta)\right)$. We note that a flashswap which is to be repaid with the received token can be denoted as a swap between token pair (i, i).

Finally, the arbitrageur should solve the problem in Eq. (1).

$$\begin{aligned}
\text{maximize} \quad & c_{k_{end},i^*} \\
\text{subject to} \quad & \forall k \in [k_{end}] : \mathbb{D}_k \in S \\
& \forall k \in [k_{end}] : (i_k, j_k) \in \text{TokenPair}\left(\mathbb{D}_k\right) \\
& \forall k \in [k_{end}] : \Delta_k > 0 \\
& \forall k \in [k_{end}] : \delta_k = \text{Swap}_{\mathbb{D}_k}^{i_k \to j_k}(\Delta_k) \\
& \forall k \in [k_{end}] : c_{k,j_k} = c_{k-1,j_k} + \delta_k \\
& \forall k \in [k_{end}] : \phi_{k,i} = \text{Debt}(\text{Swap}_{\mathbb{D}_k}^{i_k \to j_k}(\Delta_k)) \\
& \forall i \in [n] : c_{k_{end},i} - \sum_{k=0}^{k_{end}} \phi_{k,i} \geq 0
\end{aligned} \tag{1}$$

Remark 1. *"Complex" swaps may require multiple function calls. For example, a cross-platform swap requires calling the* Swap *function of different DEXs. Additionally, performing multiple function calls* *is required when swapping more than two assets in the same or different platforms, an operation also known as a "multi-hop" swap.*

Due to the EVM's specification, a single transaction can only call one function. This function can, in turn, call multiple other ones. To create a single transaction which performs multiple function calls, one must either use an existing smart contract which supports such logic, or create a new one [76] which incorporates all logic within a single function.

4.4 Arbitrage Suboptimality

As the DeFi ecosystem is flourishing, there are many tokens and DEXs which can be used to capitalize on arbitrage opportunities, thus the search space for the problem is huge.

On top of this, the optimal solution also depends on the current state of the blockchain, which changes at a fast pace – the current block time in Ethereum is 12 seconds [29, 89]. The time needed by nodes to broadcast messages over the network and validate incoming messages means that one cannot wait until the end of the allocated 12 seconds to create a block, with 4 seconds being the recommendation by the developers of Teku [28, 72], a popular Ethereum consensus client [6]. Therefore, exhaustively searching for an optimal arbitrage transaction and executing it while it is still valid becomes quite tricky.

Case Study 1 (Arbitrage Tools)**.** *Popular DeFi platforms usually make their code publicly available, publish tools that enable users to obtain data and interact with platforms in real-time, and even write tutorials to guide users in the process [9, 35, 53, 77]. Still, finding and acting on arbitrage opportunities requires considerable technical skill, giving rise to tools which attempt to automate much of the actions required to do so. As the DeFi ecosystem is rapidly evolving, with DEXs in particular, not all tools cover the entire spectrum of platforms and tokens which can be used in order to obtain profits.*

Zhou et al. *[91] have formulated and implemented an algorithm that detects profitable arbitrage opportunities across a large amount of DEXs and tokens. The authors noted that their tool obtains results within an average of* 5.39 *seconds, even though their tool does not perform an exhaustive search and thus is not optimal. The tool's run-time was achieved by (1) applying heuristics to trim the search-space (thereby potentially losing in profits), and (2) running on considerably powerful hardware (AMD Ryzen Threadripper* 3990*X processor with* 64 *cores running at* 4.3*GHz, a RAID*0 *array of four* 2*TB NVMe SSDs, and* 256*GB of RAM).*

Popular open-source tools further trim the search space by limiting themselves to single-hop swaps between two tokens [2, 61], which might under-perform relative to multi-hop swaps.

It is reasonable to assume that in contrast to Case Study 1, arbitrageurs investing resources in developing good tools for finding and capitalizing on arbitrage opportunities would avoid sharing their tools, thereby preserving potential advantages they have.

Longitudinal Study of Arbitrage Suboptimality

To further make the case that even sophisticated arbitrageurs fall short of being optimal, we implement a simple yet effective heuristic to uncover suboptimal swaps from blockchain data and use this heuristic to perform a longitudinal study of arbitrage suboptimality.

At each block, our heuristic searches for a sequence of consecutive flashswap-based arbitrage transactions, where the profit made by each transaction could have been made by the transaction which preceded it, thereby meaning that the preceding transaction was suboptimal. This heuristic is crystallized in Definition 2.

Definition 2 (Suboptimal arbitrage transaction). *Let tx_1, tx_2 be two consecutive transactions contained within the same block, sorted by their order in the block. We call tx_1 a* suboptimal arbitrage

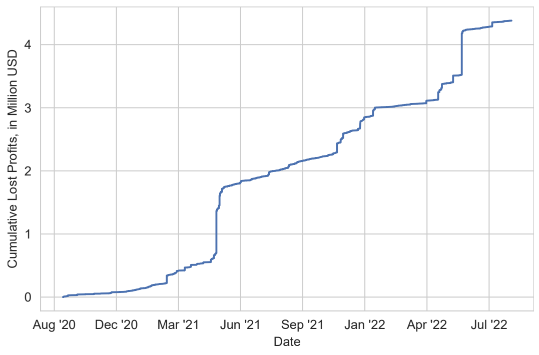


Fig. 2. Cumulative lost profits by suboptimal flashswaps in Ethereum, over time.

transaction *if both* tx_1, tx_2 *rely on swaps to profit off of arbitrage in DEXs, and transaction* tx_2 *solely relies on* flash*swaps to do so.*

Note that in Definition 2, as tx_2 only utilized flashswaps to perform the arbitrage, then transaction tx_1 could have simply replicated the actions taken by tx_2, thereby making the same profit. Thus, tx_1 was suboptimal. We generalize this notion in Definition 3.

Definition 3 (Suboptimal arbitrage sequence). *Let* $tx_1, \ldots, tx_n$ *be a sequence of consecutive transactions contained within the same block, sorted by their order in the block. We call this a* suboptimal arbitrage sequence *if* $\forall i < n$, *transaction* tx_i *is suboptimal.*

Results

We now provide evidence that indeed, suboptimal behavior is prevalent among arbitrageurs. By applying our heuristics to the time period starting at block 10749295 and ending at 15450669 (e.g., since Uniswap v2 launched, until the submission of this paper), we discovered over 9875 suboptimal transactions which fit Definition 2, with a total lost profit of over 4.38 Million USD. The cumulative lost profits over time are depicted in Fig. 2.

Over the same time period, on average 2.91 ETH were lost per day by suboptimal transactions. About 0.2% of all blocks contained at least one suboptimal sequence of arbitrage transactions, where the longest suboptimal sequence was 7 transactions long (all contained in block 15243809, starting at the transaction located at index 209).

Our heuristic discovered interesting case studies, such as Case Study 2, which shows that even arbitrageurs who are proficient enough to make large profits are not always optimal.

Case Study 2 (Account $0x9ae\ldots bc4$)**.** *Recall the transaction which was covered in Example 1. Although it produced a net profit of* 454 *ETH (equal at the time to* 600*K USD), it is, in fact, not optimal.*

Immediately following this transaction, came another one with the hash $0x580\ldots eff$. *This transaction executed another flashswap, utilizing a similar arbitrage opportunity between Sushiswap (note that Sushiswap uses the same code for swaps as Uniswap v2) and Uniswap v3. In this instance, this user was able to extract an extra* 428 *ETH in revenue, amounting to a profit of* 517*K USD, roughly* 94% *of the profit of the previous transaction.*

As the second transaction relied on flashswaps, the first arbitrageur could have performed the same actions in its first transaction, thereby obtaining at least the same amount of profits.

A more extreme case of suboptimality is given in Case Study 3.

Case Study 3 (Account $0x0f4\dots74b$). *In transaction* $0xb67\dots4a6$*, the user* $0x0f4\dots74b$ *captured a "two-hop" arbitrage opportunity between three platforms. The user swapped* 1.13 *ETH for* 4316 *USDT on Sushiswap, then swapped* 4316 *USDT for* 6058 *USDC on Uniswap v2 and finally swapped* 6058 *USDC for* 1.35 *ETH on Balancer, earning* 0.22 *ETH, or just under* 900 *USD. Although not to be sneezed at, it turns out this is* extremely *suboptimal.*

This was followed by transaction $0xc07\dots a2b$*. The creator of this transaction noticed that the previous one opened a new arbitrage opportunity, which involved the same order of operations and tokens, but on different DEXs – Uniswap v2 was used for swapping both ETH for USDT and USDT for USDC, and Sushiswap was used for swapping USD to ETH. This latter transaction was even more successful, extracting an extra* 170 *ETH in revenue (worth* 694*K USD at the time), making the initial transaction suboptimal by a factor of* ≈ 770*!*

4.5 Trimming the Search Space With Inside Information

A careful inspection of the results gathered by our heuristic revealed that certain arbitrageurs are more actively back-running suboptimal transactions, with the top 5 users, in order, being: $0x000\dots f56$ with 13K such transactions, while both $0x860\dots f66$ and $0xe33\dots c85$ are tied with 6K transactions each. Similarly, $0x911\dots116$ and $0x584\dots dba$ are tied, both with 3.6K transactions.

Some miners allow users to relay transactions directly to them instead of using the public peer to peer (p2p) network, while guaranteeing the privacy of transactions relayed in that manner [32, 43], with the intention of preventing transaction front-running or other potential attacks. Such channels, also known as *private relays*, have become widely adopted by arbitrageurs and miners alike [65].

In this aspect, arbitrageur $0x584\dots dba$ should be particularly noted – circumstantial evidence suggests $0x584\dots dba$ is possibly tied to the *Ethermine* mining pool [67], which was the largest before Ethereum's transition to proof-of-stake (PoS) [86]. Specifically:

(1) All of the arbitrageur's transactions have appeared solely in blocks mined by Ethermine.
(2) The arbitrageur's first and last transactions both correspond to the times at which Ethermine launched and shut-down its private communication channel [33, 34].

This is the *first* evidence of such miner behavior.

Thus, we raise the possibility that the success of $0x584\dots dba$ in identifying unexploited arbitrage is due to "inside information" it has from its ties to a private relay. This information can be used, for example, to trim the huge arbitrage search-space using transactions which were privately relayed to it. Indeed, in many of the results which were found by our heuristics, neighboring transactions use almost identical operations, usually only differing in the DEXs they use to carry out their arbitrage (e.g., Case Study 2 and Case Study 3).

4.6 Discussion

Suboptimal arbitrage transactions might leave arbitrage opportunities open. As arbitrageurs are actively seeking out such opportunities [65, 81], this means that unexploited arbitrage leave the door open to potentially other transactions which will attempt to capitalize on it. Thus, this suboptimality can result in increased congestion [54], leading to higher transaction fees for all users of the cryptocurrency.

5 LENDING SUBOPTIMALITY

Lending platforms such as Aave [84] and Compound [52] exist as Ethereum smart contracts, and let users take out collateralized loans. As of the time of writing, these two platforms hold a combined amount of 6.2 Billion USD [22], making them the most popular lending platforms on the

Ethereum blockchain. We present a general model for the aforementioned platforms, and present commonplace suboptimal behavior by users interacting with them.

5.1 Interest-Bearing Liquidity Pools

DeFi platforms like Compound and Aave rely on user-provided funds collected in so-called *liquidity-pools* for their operation, with such users called LPs. Platforms incentivize liquidity provisioning by giving interest on funds stored in their pools [46], with the annualized interest referred to as annual percentage yield (APY).

Usually, platforms define their interest-rate mechanisms to ensure that the *interest-rate spread* is positive, e.g., the difference between the interest paid on borrowing funds and the interest earned for supplying them is larger than 0. Another precautionary step taken by lending platforms is to reserve a fraction r of the interest paid by borrowers; this fraction is termed the *reserve-factor.* These reserves can be used by the platforms to safeguard against borrower insolvency and potential economic shocks.

Most platforms advertise their interest-rate schemes as designed to maximize available liquidity. Even so-called *yield-farming* services, designed to automatically invest user funds thereby maximizing profits for LPs [18], invest all funds at all time. As we show, this is not optimal.

Platforms might design interest mechanisms to pursue other goals, such as preventing interest-rate volatility [26, 27], and even solely for the sake of standardization [11]. Platforms differ in their approach to accruing interest, either on a per-block basis, or according to block timestamps [87]. Both can be manipulated, although such behavior was not observed in the wild, with manipulations (if existing) restricted to earning more mining-related rewards rather than gaming interest rate mechanisms [86, 87]. Thus, we make the reasonable assumption that block timestamps are honest and that interest is accrued solely according to them. Although interest-accretion is usually associated with lending, any platform that rewards liquidity provisioning is essentially paying interest. A notable example is Tornado Cash and other so-called “Privacy Protocols” [50].

5.2 Utilization-based Interest Rate Schemes

Compound Finance, a DeFi platform that runs on top of Ethereum’s blockchain, is a popular liquidity pool for a variety of ERC20 tokens. Compound’s mechanisms were designed to make sure that on the one hand enough funds are available for would-be loan-takers, and that existing liquidity is indeed used. To that end, Compound defines the following metric.

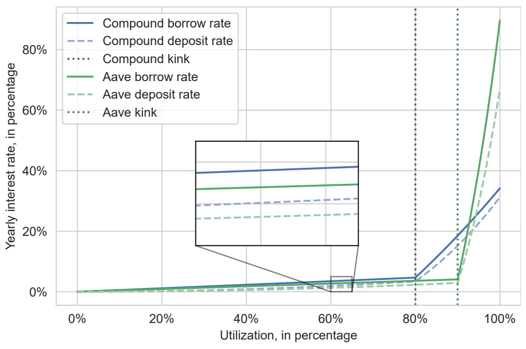
Definition 4 (Utilization). *If the total amount of pool liquidity is d, and $b \leq d$ is currently borrowed, then the available liquidity is $d - b$ and the utilization of the liquidity pool is: $u \overset{def}{=} \frac{b}{d}$.*

For each token, the interest-rates for deposits and borrows are determined according to the utilization of the token’s liquidity pool. Specifically, certain pools, such as the one for the USDT stablecoin [56], uses a *kinked*-curve interest-rate scheme [42, 46]. In this type of scheme, the rates for both deposits and borrows are increasing in the utilization of pool, with the slope of the curve increasing after a certain target utilization value, which we denote by u_{opt}. The resulting function looks like it has a “kink” at that point, as can be seen in Fig. 3a, hence the name.

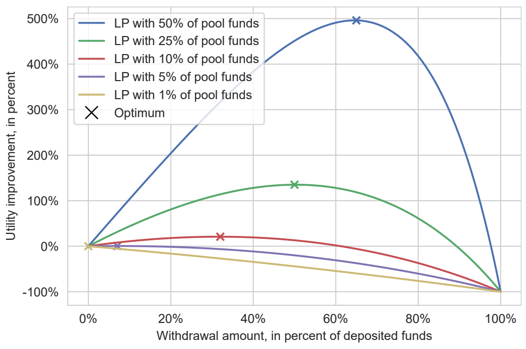
In Example 2, we go over Compound Finance’s actual interest-rate curve, and formally write it down using our notations.

Example 2. *If we denote the baseline interest-rate as I_0 and the slope before and after u_{opt} by I_1 and I_2 respectively, Compound’s per-block interest on USDT borrows can be written as follows [16, 36]:*

$$I_b\left(u\right) \overset{def}{=} I_0 + \min\left(u, u_{opt}\right) \cdot I_1 + \left(\max\left(u, u_{opt}\right) - u_{opt}\right) \cdot I_2 \qquad (2)$$



(a) Compound's and Aave's yearly interest rates on deposits and borrows, as functions of the utilization. Both have a "kink", at 80% and 90%, respectively.



(b) The increase in utility a LP can make by withdrawing funds, as a function of the percentage of withdrawn funds, for LPs of different sizes.

Fig. 3. The optimal liquidity provisioning strategy might not rely on depositing all available funds, as this can *lower* the interest-rate.

The interest for supplying liquidity is a function of the borrow rate:

$$I_d(u) \overset{def}{=} u \cdot (1-r) \cdot I_b(u) \tag{3}$$

Note that by definition, the interest-rate for deposits is quadratic in the utilization. This can be seen by substituting Eq. (2) *in Eq.* (3)*, thereby giving Eq.* (4)*:*

$$\begin{aligned} I_d(u) &= u \cdot (1-r) \cdot I_b(u) \\ &= (1-r)\left(uI_0 + \min\left(u^2, u_{opt}\right) I_1\right) \\ &+ (1-r)\left(\max\left(u^2, u_{opt}\right) - u_{opt}\right) I_2 \end{aligned} \tag{4}$$

These schemes are in use by other platforms besides Compound, for example Aave [46], both of which are depicted in Fig. 3a.

5.3 Optimal Liquidity Provisioning

The amount of yield on a given deposit is dependent on both the amount of funds deposited and the interest rate. The interest rate itself is *also* dependent on the amount of funds deposited, as the rate is a function of the utilization, which in turn is a function of the total amount of deposited funds.

We generalize this observation for any liquidity pool relying on mechanisms which are functions of the pool's utilization by formalizing a per-platform optimization problem in Eq. (5), to optimize a single user's deposit d' given a maximal amount of available funds d^*.

We denote the platform's interest rate for liquidity suppliers by $I_d(\cdot)$, the total amount of funds given out by the pool as loans by b, and the amount of funds supplied to the pool by other LPs as d.

$$\begin{aligned} \underset{d'}{\text{maximize}} \quad & d' \cdot I_d(u) \\ \text{subject to} \quad & d' \geq 0 \\ & d' \leq d^* \\ & u = \frac{b}{d+d'} \end{aligned} \tag{5}$$

Remark 2. *This optimization considers a one-shot setting. Thus, profits are optimized under the assumption that the utilization remains unchanged after the optimizing actor's action. We note that an actor can ensure that indeed the utilization will be unchanged for at least the span of a single block, by paying the appropriate amount of fees such that its transaction will be positioned to be the last one in the block [20], or by sending the transaction as part of a transaction bundle [7].*

In the case that the actor's amount of funds is large compared to others', Fig. 3b shows that the potential profits from even a single block can be substantial. As previous research shows, the majority of funds in most pools are held by a small number of independent actors [46], meaning that such considerations are applicable to the real world.

Furthermore, empirical data suggests that the market is slow to react to large liquidity shocks. Notably, Aave's CRV liquidity pool recently experienced a series of shocks, starting with a single withdrawal of more than 24% *of its liquidity performed at block* 16011068 *by a single actor. After this single action, the pool's utilization remained at roughly the same level for a period of* 1172 *blocks. The same actor performed more large withdrawals and deposits between the 6th and 27th of November, 2022. As no other user would "fill the void" left by the actor's withdrawals, AAVE passed proposals to (1) Temporarily disable new borrows and withdrawals from this pool, (2) Tweak the pool's interest-rate curve. After the restrictions were lifted, the actor continued performing actions of a similar magnitude.*

5.4 Suboptimal Lending

In Fig. 3b we show that the optimal action given by Eq. (5) for a LP which is in possession of large amounts of funds is not necessarily to deposit all of its money. For example, a LP who owns 25% of all funds deposited in a pool can withdraw almost half of its funds, thereby increasing per-block profits by more than 10%.

In the following, we share notable examples of suboptimal capital allocation to interest-bearing pools by large liquidity providers. We begin by showing in Case Study 4 that Yearn, the largest investment platform on the Ethereum blockchain, is suboptimal.

Case Study 4 (Yearn Finance). *Yearn Finance is an on-chain* yield aggregator *or* yield farming *service, which is essentially an investment platform that automates the distribution of funds across platforms for users, claiming to maximize the interest rate the user can yield out of its respective funds [18, 87]. Yearn is currently the largest on-chain yield aggregator on the Ethereum blockchain [24], with a total value locked (TVL) of* 400 *Million USD at the time of writing (and reaching over* 6.5 *Billion USD at its peak) [23].*

Yearn follows the "all-in" approach by comparing interest rates across platforms, and depositing all funds in the platform offering the best rates [38, 90]. As we have shown, this "all-in" approach is, in fact, suboptimal regarding utilization-based interest rate schemes.

In Case Study 5 we show a specific instance of a user performing suboptimally.

Case Study 5 (Justin Sun). *Compound's largest LP for the cUSDT token at block* 13632753 *was address* $0x3dd\ldots296$*, associated with Justin Sun [37], providing* 2.8 *Billion cUSDT. At the time, the supply APY was* 3.52% *owing to a utilization of* 80.09%*. Mr. Sun left all of his funds deposited, and earned* 40.55 *cUSDT by the next block.*

By acting rationally, Mr. Sun can increase these profits by withdrawing *funds. Specifically, by withdrawing* 0.7 *Billion cUSDT, Mr. Sun can increase the utilization to* 81.3%*, thus increasing the supply APY to* 5.15%*. His profit would therefore grow to* 44.1 *cUSDT per-block, an increase of* 8.7%*.*

If performed correctly, this per-block profit can be assured for the next block if the transaction which performs the withdrawal is placed as the last transaction of the block. Thus, this increased interest-rate will remain until other users interact with the liquidity pool to the degree that the utilization is changed.

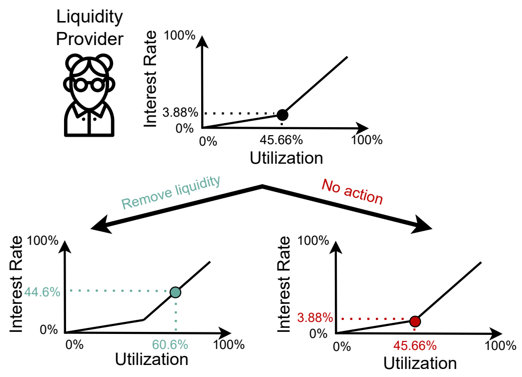


Fig. 4. The LP from Case Study 6 has a deposit of 132.9 Million CRV in Aave. Its profit can be increased by 700% by *withdrawing* 38 Million CRV.

Note that this suboptimality is two-fold: after withdrawing the aforementioned funds, Mr. Sun could invest them, for example by depositing them in another liquidity pool.

We provide a more dramatic case of suboptimality in Case Study 6, with a graphical depiction of the case given in Fig. 4.

Case Study 6 (Account 0x7a1 . . . 428)**.** *At block* 15529008*, address* 0x7a1 . . . 428 *had a deposit of* 132.9 *Million CRV in Aave's liquidity pool, making this address the largest CRV LP on Aave, providing almost* 85% *of the pool's liquidity.*

The pool's utilization at the time was equal to 45.66%*. According to the pool's interest-rate scheme, at this utilization the supply APY is* 3.88%*. As the pool's kink is precisely at* 45%*, the supply rate can be increased drastically by withdrawing even a small amount of funds. Indeed, our LP can increase the utilization to* 60.6% *by withdrawing* 38 *Million CRV. This modest bump in utilization increases the supply APY to* 44.6%*, thereby increasing our LP's profit by* 700%*.*

5.5 Discussion

We remark that while LPs miss out on potential revenue by acting suboptimally, any optimal action they may take will serve to increase both supply and borrow interest rates, due to the underlying mechanism's reliance on the utilization metric.

Thus, suboptimality by LPs leads to lower interest-rates for both lenders *and borrowers*, meaning that the latter are profiting off of the suboptimality of the former.

6 LIQUIDATION SUBOPTIMALITY

Liquidity providers bear the risk that borrowers might default (see Section 5). A debt default can happen if the collateral asset cannot cover the loan, for example due to a drop in the asset's exchange-rate. Typically, platforms define a *liquidation threshold*, below which other users can buy, or *liquidate*, the borrower's debt, while receiving the collateral at a discount. This discount is known as the liquidation *spread*.

6.1 Fixed Spread Liquidation

At the time of writing, the fixed spread liquidation (FSL) is the most prevalent method for securing the LPs' capital [64]. On a high level, FSL incentivizes entities, so-called liquidators, to repay the outstanding debt for a borrower. In return, the liquidator is allowed to acquire pro rata collateral from the borrower. By design, the financial value of the acquired collateral exceeds the debt repaid by a liquidator, which underpins the incentive compatibility of FSL. We proceed to formulate the FSL mechanism.

We assume a debt position collateralized by cryptocurrency C with an outstanding debt in cryptocurrency $\mathcal{D}$.[1] The amount of collateral and debt is c and d respectively. We apply the USD price of C and $\mathcal{D}$ to unify financial value calculation, denoted by p^C and $p^{\mathcal{D}}$ respectively. To measure the "health" of a debt position, meaning how close it is to insolvency, platforms use a metric called the *health factor*. The health factor is defined by Eq. (6), where lt is the liquidation threshold, s.t. $0 < lt < 1$.

$$hf \overset{\text{def}}{=} \frac{p^C \cdot c}{p^{\mathcal{D}} \cdot d} \cdot lt \tag{6}$$

The health factor measures the collateralization state of a borrowing position. When the collateral value decreases because of, for example, price decline (i.e., $\frac{p^C}{p^{\mathcal{D}}} \downarrow$), hf decreases, indicating that the borrower is more likely to default. Once hf falls below one, the borrowing position is available for liquidations. Any liquidator is then allowed to repay rd of debt to the liquidity pool, while receiving the collateral at a discount. The discount is called the liquidation *spread* and is defined to be a fixed ratio, which we denote as ls. The collateral acquired by the liquidator within this liquidation is denoted by ac and given in Eq. (7).

$$ac \overset{\text{def}}{=} \frac{p^{\mathcal{D}} \cdot (1 + ls)}{p^C} \cdot rd \tag{7}$$

For brevity, we define the following: $p^{\text{liq}} \overset{\text{def}}{=} \frac{p^{\mathcal{D}} \cdot (1+ls)}{p^C}$, thereby we can simplify Eq. (7) to $ac = p^{\text{liq}} \cdot rd$. Without considering transaction fees, the liquidator's theoretical profit is formulated as Eq. (8).

$$\text{profit}^t(rd) = p^C \cdot ac - p^{\mathcal{D}} \cdot rd = ls \cdot p^{\mathcal{D}} \cdot rd \tag{8}$$

Platforms often constrain the fraction of debt that a liquidator is allowed to repay within a single FSL by a *close factor* cf, e.g. Eq. (9).

$$cf \geq \frac{rd}{d} \tag{9}$$

6.2 Flash Loan Usage in Liquidation

FSL may require a liquidator to hold various assets upfront for debt repayment. This presents operational risks to liquidators due to the price volatility of most cryptocurrencies. Because of such risks, flash loans [66] have become a popular option for liquidators [51, 80].

A flash loan is granted and repaid within a single blockchain transaction. Contrary to other forms of DeFi lending, a flash loan borrower does not need to provide any collateral. The atomicity of blockchain transactions guarantees that, if a borrower cannot repay the loan by the end of the transaction, the transaction will be reverted, as if no loan was issued.

With a flash loan, a liquidator can borrow assets in $\mathcal{D}$ to repay liquidated debt and acquire collateral in C. To repay the flash loan, the liquidator needs to exchange the acquired C to $\mathcal{D}$. This process is completed in a single transaction, as required by the flash loan.

[1]In practice, a debt position can have multi-cryptocurreny collateral (debt). For simplicity, we assume that the collateral (debt) is in single cryptocurreny.

6.3 Liquidator Suboptimality

Given Eqs. (8) and (9), the profitability of FSL is formalized in Eq. (10).

$$\begin{aligned} \underset{rd}{\text{maximize}} \quad & ls \cdot p^{\mathcal{D}} \cdot rd \\ \text{subject to} \quad & rd \leq cf \cdot d \end{aligned} \tag{10}$$

Intuitively, to optimize liquidation profits, a liquidator should liquidate up to the close factor constraint, because $\text{profit}^{t}(rd)$ increases monotonically with regards to (w.r.t.) rd (see Eq. (8)). However, the theoretical profit does not hold when an asset exchange from C to $\mathcal{D}$ is involved in the liquidation process. We hence reformulate the liquidation profit as follows.

We assume that the liquidator completes the exchange from C to $\mathcal{D}$ over a DEX $\mathbb{D}$. At the time of liquidation, the spot price from C to $\mathcal{D}$ on $\mathbb{D}$ is $p^{C \to \mathcal{D}}$. The actual exchange rate from C to $\mathcal{D}$ on $\mathbb{D}$ may, however, diverge from the spot price due to the so-called slippage [92]. Therefore, $\text{Swap}^{C \to \mathcal{D}}(\Delta c)$, representing the amount of $\mathcal{D}$ that a trader receives when exchanging Δc units of C over $\mathbb{D}$, is formulated in Eq. (11).

$$\text{Swap}^{C \to \mathcal{D}}(\Delta c) = (1 - \text{Slippage}(\Delta \text{c})) \cdot p^{C \to \mathcal{D}} \cdot \Delta c \tag{11}$$

Notably, $\text{Slippage}(\Delta c)$ is typically a monotonically increasing function w.r.t. Δc, implying that the exchange rate becomes less favorable for a trader when the trading volume increases.

Technically, a liquidator can exchange an arbitrary amount of C to $\mathcal{D}$ as long as the flash loan can be covered. In our model, we assume that all acquired C is exchanged to $\mathcal{D}$ to ease understanding. The liquidation profit with a flash loan is hence outlined in Eq. (12).

$$\begin{aligned} \text{profit}^{f}(rd) &= p^{\mathcal{D}} \cdot \text{Swap}^{C \to \mathcal{D}}(ac) - p^{\mathcal{D}} \cdot rd \\ &= p^{\mathcal{D}} \cdot (1 - \text{Slippage}(ac)) \cdot p^{C \to \mathcal{D}} \cdot ac - p^{\mathcal{D}} \cdot rd \\ &= p^{\mathcal{D}} \cdot \left(p^{C \to \mathcal{D}} \cdot p^{\text{liq}} \cdot \left(1 - \text{Slippage}\left(p^{\text{liq}} \cdot rd\right)\right) - 1\right) \cdot rd \end{aligned} \tag{12}$$

Recall that $\text{Slippage}\left(p^{\text{liq}} \cdot rd\right)$ is a monotonically increasing function w.r.t. rd. Therefore, $\text{profit}^{f}(rd)$ is a concave function w.r.t. rd, indicating that liquidating up to the close factor might not be the optimal strategy for FSL.

We now derive the optimal FSL strategy by instantiating a DEX with the constant product rule, a prevalent DEX pricing formula [81, 92]. A DEX following the constant product rule is referred to as a constant product automated market maker (CPAMM), which is in essence a smart contract reserving a pair of cryptocurrencies. Any trader can trade against a CPAMM by sending one cryptocurrency to the contract and, in exchange, taking the other cryptocurrency from the contract. Without considering fees, an exchange between the two assets should not change the asset amount product, which is detailed in the following.

We assume a CPAMM with x_c of C and x_d of $\mathcal{D}$ reserved. The spot price from C to $\mathcal{D}$ is hence $\frac{x_d}{x_c}$. Fulfilling the constant product rule, we obtain Eqs. (13) and (14).

$$\text{Swap}^{C \to \mathcal{D}}(\Delta c) = x_d - \frac{x_c \cdot x_d}{x_c + \Delta c} \tag{13}$$

$$\text{Slippage}(\Delta c) = 1 - \frac{x_c}{\Delta c} + \frac{\frac{x_c}{\Delta c}}{1 + \frac{\Delta c}{x_c}} \tag{14}$$

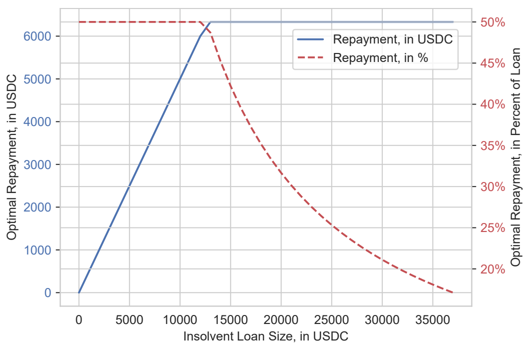


Fig. 5. Optimal repayment in a swap-based liquidation for Aave's USDC pool, that has a close factor of 50%. Due to exchange-rate slippage, it is suboptimal to repay large illiquid positions up to the close factor.

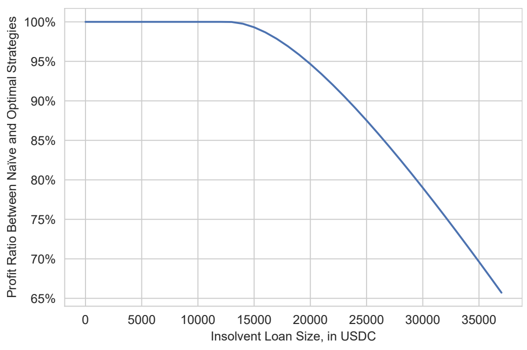


Fig. 6. The ratio between profits made by naïve and optimal repayments in swap-based liquidations for Aave's USDC pool. The ratio drops below 1 for large loans, indicating that the naïve strategy is suboptimal.

Thus, we obtain the liquidation profit with flash loans in Eq. (15).

$$\begin{aligned} \underset{rd}{\text{maximize}} \quad & x_d - \frac{x_c \cdot x_d}{x_c + p^{\text{liq}} \cdot rd} - rd \\ \text{subject to} \quad & rd \leq cf \cdot d \end{aligned} \tag{15}$$

By solving Eq. (15) analytically, we get that the optimal amount rd^* to repay in a flashloan-and-swap-based liquidation is:

$$rd^* = \min\left(cf \cdot d, \frac{\sqrt{p^{\text{liq}} \cdot x_c \cdot x_d} - x_c}{p^{\text{liq}}}\right) \tag{16}$$

The optimal strategy presented in Eq. (15) can improve profits substantially when compared to a naïve liquidation strategy which liquidates up to the close factor, even when liquidating relatively small loans. To illustrate, in Fig. 6 we plot the ratio between the profit obtained by the naïve strategy and our optimal one. In addition, we plot the corresponding absolute and relative amounts which should be liquidated in Fig. 5. Both figures assume the debt to be liquidated is in Aave's USDC liquidity pool, and the DEX used for the swap is Uniswap v2, with all relevant parameters set to their real-world values as of block 15731128.

Case Study 7 (Automatic Tools For Flash-Liquidation)**.** *Popular tools which automatically look for insolvent debt positions and liquidate them, while using flashwaps (or flashloans and swaps) to do so, do not follow the optimal strategy we present in Eq.* (16)*, instead always performing liquidations up to the close factor [1, 25, 69].*

6.4 Discussion

Aave advertises their liquidation mechanism as beneficial for the health of the platform [3], as liquidations ensure that unhealthy positions are adequately collateralized. When users liquidate bad debt, they essentially perform a service for the platform, driven by the possibility of obtaining a debtor's collateral at a discount.

For large unhealthy positions, the funds required for liquidation can be substantial. Users who do not have such funds are required to rely on actions such as flashloans and swaps. As we have shown, in such cases utility-maximizing users might prefer to repay only a small amount of debt, which might be short of the extent that is required to pull the position's health factor back to a reasonable level (according to the mechanism's risk parameters). Thereby, the health of the platform is potentially harmed.

7 RELATED WORK

As far as we are aware, no previous work has examined user suboptimality in the context of the DeFi primitives we covered.

Empirical Studies. Gudgeon *et al.* [46] formalized popular DeFi interest-rate mechanisms and used historical data to compare platforms that use them. A systematization of common liquidation mechanisms was given by Qin *et al.* [64], which also empirically analyzed liquidations performed on large Ethereum lending platforms. The historical performance of Uniswap v2 users performing cyclic arbitrage swaps was explored by Wang *et al.* [81], where the profits from each swap are used in their entirety to pay for the next swap in the cycle, a strategy which is far from optimal. A larger spectrum of actions which users can use to obtain profits from DEXs is formalized and examined by Zhou *et al.* [92], for example users can both back-run and front-run other transactions (place their own transactions before or after other transactions, respectively) and even combine the two into a "sandwich" attack; they estimate the profits which can be obtained in that manner in thousands of dollars per day. Qin *et al.*[65] quantified the historical profits made from liquidation, arbitrage and sandwich attacks, and show that large profits risk the security of the underlying blockchain as they incentivize miners to misbehave. Piet *et al.* [62] and Weintraub *et al.* [82] analyze the profits produced by private transactions in Ethereum, but ignored collateralized lending.

Automatic Tools. Zhou et *al.* [91] introduce two suboptimal tools for creating profitable DeFi transactions, one which can detect a profitable swap cycle, and another that also detects non-cyclic ones. Qin *et al.* [65] present a tool which monitors unconfirmed transactions and attempts to front-run profitable ones by replicating their logic in an application-agnostic manner. Angeris *et al.* [8] use convex optimization to identify arbitrage opportunities in a network of constant function market makers (CFMMs) DEXs and approximate the optimal solution.

Optimal Uniswap V3 Liquidity Provisioning. A strand of works [40, 58] study strategic Uniswap V3 liquidity providers who have some predefined set of beliefs regarding the price evolution of assets, and formalize their optimal behavior in various settings. In contrast, our work considers liquidity provisioning for interest-bearing liquidity pools such as Aave and Compound, and limit our scope of user interactions with Uniswap-esque DEXs to flashswaps.

Optimizing Specific Instances of Flash Loan Attacks. Previous work has examined specific instances of flash loan attacks, and have shown that the profits made by them can be improved. For example, Qin *et al.* [66] did so in two specific cases, the famous bZx "Pump & Arbitrage" and "Oracle Manipulation" attacks. Cao *et al.* [14] have produced a tool that analyzes flash loan attacks, and used it to conduct a similar analysis on the bZx "Pump & Arbitrage" attack. We extend these cases into a general formalization of suboptimality of honest, non-malicious, user behaviour.

Miner Extractable Value. Angeris *et al.* [7] cast the action-space which miners have at their disposal when constructing a block as an optimization problem (e.g., which transactions to include and in what order), while taking into account the profit that can be made from transaction reordering and sandwich attacks. Obadia *et al.* [59] present a formal study of the MEV that can be obtained by miners which operate in multiple blockchains. The game-theoretic aspects of MEV are analyzed by Kulkarni *et al.* [49], specifically in the context of CFMM DEXs. Yaish *et al.* [87] show that miners can manipulate the rate at which blocks are mined in popular proof-of-work (PoW) mechanisms, and that although such manipulations can reduce the profits from mining incentives (such as block rewards), they can be used to create profitable interest-rate gaps between on-chain lending platforms. The authors show that the attack can be prevented by limiting system parameters, e.g., the interest-rate offered by DeFi platforms.

Preventing Miner Malfeasance. Both Orda *et al.* [60] and Ferreira *et al.* [41] attempt to prevent miners from manipulating transaction order to their benefit, with the latter focusing on order manipulations specifically in the context of decentralized exchanges. Yaish *et al.* [86] show that in PoW-based Ethereum-like blockchains, miners can risklessly retroactively replace blocks, and indeed have done so in Ethereum before it transitioned to a different consensus mechanism. The authors suggest various mitigation techniques for the attack, and note such manipulations let miners to replay [65] profitable transactions.

8 CONCLUSION

In this work we study three core DeFi primitives: lending, liquidations which rely on swaps, and flashswaps. To the best of our knowledge, we are the first to show that even large platforms and big market actors commonly use them suboptimally.

We perform a longitudinal study on one of them, and show that the losses due to suboptimality exceeded 4 Million USD over the examined period. Importantly, our study is the first to uncover a large miner using inside information to its benefit.

Through our work, we hope to draw attention to certain under-explored aspects of DeFi which are important for the integrity and efficient operation of platforms.

REFERENCES

[1] 0xnivek. 2021. Joe Liquidator. https://web.archive.org/web/20221012180624/https://github.com/0xnivek/joe-liquidator

[2] 6eer. 2021. uniswap-sushiswap-arbitrage-bot. https://github.com/6eer/uniswap-sushiswap-arbitrage-bot

[3] Aave. 2022. Liquidations. https://web.archive.org/web/20221013175355/https://docs.aave.com/developers/guides/liquidations

[4] Hayden Adams, Noah Zinsmeister, and Dan Robinson. 2020. Uniswap v2 core. https://web.archive.org/web/20220126073458/https://uniswap.org/whitepaper.pdf

[5] Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. 2021. Uniswap v3 core. https://web.archive.org/web/20221011013240/https://uniswap.org/whitepaper-v3.pdf

[6] Ether Alpha. 2022. Client Diversity. https://web.archive.org/web/20221207190225/https://clientdiversity.org/#distribution

[7] Guillermo Angeris, Alex Evans, and Tarun Chitra. 2021. A Note on Bundle Profit Maximization.

[8] Guillermo Angeris, Alex Evans, Tarun Chitra, and Stephen Boyd. 2022. Optimal Routing for Constant Function Market Makers. In *Proceedings of the 23rd ACM Conference on Economics and Computation* (Boulder, CO, USA) *(EC '22)*. Association for Computing Machinery, New York, NY, USA, 115–128. https://doi.org/10.1145/3490486.3538336

[9] Adam Bavosa. 2020. Supplying Assets to the Compound Protocol. https://medium.com/compound-finance/supplying-assets-to-the-compound-protocol-ec2cf5df5aa

[10] Alex Beregszaszi and Nikolai Mushegian. 2017. EIP-140: REVERT instruction. https://eips.ethereum.org/EIPS/eip-140

[11] blck. 2020. Upgrade cUSDT Interest Rate Model. https://compound.finance/governance/proposals/20

[12] Vitalik Buterin. 2022. Ethereum Whitepaper. https://web.archive.org/web/20220728020709/https://ethereum.org/en/whitepaper/

[13] Giulio Caldarelli. 2021. Wrapping Trust for Interoperability: A Preliminary Study of Wrapped Tokens. *Information* 13, 1 (2021), 6.

[14] Yixin Cao, Chuanwei Zou, and Xianfeng Cheng. 2021. Flashot: A Snapshot of Flash Loan Attack on DeFi Ecosystem. https://doi.org/10.48550/ARXIV.2102.00626

[15] Weili Chen, Tuo Zhang, Zhiguang Chen, Zibin Zheng, and Yutong Lu. 2020. *Traveling the Token World: A Graph Analysis of Ethereum ERC20 Token Ecosystem.* Association for Computing Machinery, New York, NY, USA, 1411–1421. https://doi.org/10.1145/3366423.3380215

[16] Compound. 2021. Interest accrual function. https://git.io/JP5Ao

[17] Compound. 2022. cTokens. https://compound.finance/docs/ctokens

[18] Simon Cousaert, Jiahua Xu, and Toshiko Matsui. 2021. SoK: Yield Aggregators in DeFi. arXiv:2105.13891 [q-fin.PM]

[19] Curve. 2021. Curve DAO. https://web.archive.org/web/20210811065239/https://curve.fi/files/CurveDAO.pdf

[20] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. 2020. Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*. IEEE, San Francisco, CA, USA, 910–927. https://doi.org/10.1109/SP40000.2020.00040

[21] DefiLlama. 2022. DEXs Ethereum Volumes. https://web.archive.org/web/20221011072608/https://defillama.com/dexs/ethereum

[22] DefiLlama. 2022. Lending TVL Rankings. https://web.archive.org/web/20221011073058/https://defillama.com/protocols/lending/Ethereum

[23] DeFiLlama. 2022. Yearn Finance: TVL and Stats. https://web.archive.org/web/20221013234041/https://defillama.com/protocol/yearn-finance

[24] DefiLlama. 2022 (accessed October 9, 2022). DeFi Dashboard. https://defillama.com

[25] Mike De'Shazer. 2020. FlashLoanLiquidation. https://web.archive.org/web/20221012180627/https://github.com/mikedeshazer/FlashLoanLiquidation

[26] Dharma. 2020. Update DAI Interest Rate Model to Better Accommodate Zero Stability Fee. https://compound.finance/governance/proposals/2

[27] Dharma. 2020. Upgrade cUSDC Interest Rate Model. https://compound.finance/governance/proposals/23

[28] Ben Edgington. 2022. Upgrading Ethereum | One Page Annotated Spec. https://web.archive.org/web/20221218133524/https://eth2book.info/bellatrix/annotated-spec/#seconds_per_slot

[29] Ethereum. 2022. Blocks. https://web.archive.org/web/20220922171539/https://ethereum.org/en/developers/docs/blocks/#block-time

[30] Ethereum.org. 2021. ERC-20 Token Standard. https://ethereum.org/en/developers/docs/standards/tokens/erc-20/

[31] ethereum.org. 2022. Ethereum Virtual Machine (EVM). https://ethereum.org/en/developers/docs/evm/

[32] ethermine.eth. 2021. Want to keep your dex trades away from the public mempool? We are proud to announce the Ethermine Private RPC endpoint. https://web.archive.org/web/20221013133550/https://nitter.it/ethermine_org/status/1443502516604477445

[33] ethermine.eth. 2022. Ethermine is pleased to say that it has secured the #Ethereum network for the past 7 years and mined 3,271,518 Blocks and a total of 9,836,656 Ether. https://web.archive.org/web/20221013230053/https://nitter.it/ethermine_org/status/1570302744992583681

[34] ethermine.org. 2021. We are proud to announce the next step of #Ethermine MEV Beta Ethermine MEV Relay! https://web.archive.org/web/20221019190843/https://nitter.it/ethermine_org/status/1404464604663713798?lang=en

[35] Etherscan. 2021. Compound's deployed CToken contract for the USDT token. https://etherscan.io/address/0xa035b9e130f2b1aedc733eefb1c67ba4c503491f#code#F3#L384

[36] Etherscan. 2021. Compound's deployed JumpRateModelV2 for the USDT token. https://etherscan.io/address/0xfb564da37b41b2f6b6edcc3e56fbf523bd9f2012#code

[37] Etherscan. 2022. Compound USDT (cUSDT) Token Tracker. https://etherscan.io/token/0xf650c3d88d12db855b8bf7d11be6c55a4e07dcc9#balances

[38] Etherscan. 2022. Contract 0x83f798e925BcD4017Eb265844FDDAbb448f1707D. https://etherscan.io/address/0x83f798e925bcd4017eb265844fddabb448f1707d#code#L682 Yearn's yUSDT token rebalances itself in line 682 if a certain platform offers a better interest rate, by withdrawing all funds from the current platform and depositing in the new one.

[39] Etherscan. 2022. Contract 0xCBCdF9626bC03E24f779434178A73a0B4bad62eD. https://etherscan.io/address/0xcbcdf9626bc03e24f779434178a73a0b4bad62ed#code#F1#L777

[40] Zhou Fan, Francisco Marmolejo-Cossío, Ben Altschuler, He Sun, Xintong Wang, and David C. Parkes. 2022. Differential Liquidity Provision in Uniswap v3 and Implications for Contract Design. https://doi.org/10.48550/ARXIV.2204.00464

[41] Matheus V. X. Ferreira and David C. Parkes. 2022. Credible Decentralized Exchange Design via Verifiable Sequencing Rules. https://arxiv.org/abs/2209.15569

[42] Compound Finance. 2022. USDT Market. https://compound.finance/markets/USDT

[43] Flashbots. 2022. Flashbots. https://github.com/flashbots/pm

[44] Yotam Gafni and Aviv Yaish. 2022. Greedy Transaction Fee Mechanisms for (Non-)myopic Miners. https://doi.org/10.48550/ARXIV.2210.07793

[45] Robert P Goldberg. 1974. Survey of virtual machine research. *Computer* 7, 6 (1974), 34–45.

[46] Lewis Gudgeon, Sam Werner, Daniel Perez, and William J. Knottenbelt. 2020. DeFi Protocols for Loanable Funds: Interest Rates, Liquidity and Market Efficiency. In *AFT '20: 2nd ACM Conference on Advances in Financial Technologies, New York, NY, USA, October 21-23, 2020.* ACM, 92–112. https://doi.org/10.1145/3419614.3423254

[47] Everett Hildenbrandt, Manasvi Saxena, Nishant Rodrigues, Xiaoran Zhu, Philip Daian, Dwight Guth, Brandon Moore, Daejun Park, Yi Zhang, Andrei Stefanescu, et al. 2018. Kevm: A complete formal semantics of the ethereum virtual machine. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF).* IEEE, 204–217.

[48] Kamil Jezek. 2021. Ethereum Data Structures. https://doi.org/10.48550/ARXIV.2108.05513

[49] Kshitij Kulkarni, Theo Diamandis, and Tarun Chitra. 2022. Towards a Theory of Maximal Extractable Value I: Constant Function Market Makers. https://doi.org/10.48550/ARXIV.2207.11835

[50] Duc V. Le and Arthur Gervais. 2021. *AMR: Autonomous Coin Mixer with Privacy Preserving Reward Distribution.* Association for Computing Machinery, New York, NY, USA, 142–155. https://doi.org/10.1145/3479722.3480800

[51] Alfred Lehar and Christine A Parlour. 2022. Systemic Fragility in Decentralized Markets.

[52] Robert Leshner and Geoffrey Hayes. 2019. Compound: The money market protocol.

[53] Calvin Liu. 2018. How to Earn Interest and Borrow Ethereum Assets. https://medium.com/compound-finance/the-compound-guide-to-supplying-borrowing-crypto-assets-94821f2950a0

[54] Yulin Liu, Yuxuan Lu, Kartik Nayak, Fan Zhang, Luyao Zhang, and Yinhong Zhao. 2022. Empirical Analysis of EIP-1559: Transaction Fees, Waiting Times, and Consensus Security. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22).* Association for Computing Machinery, New York, NY, USA, 2099–2113. https://doi.org/10.1145/3548606.3559341

[55] Fernando Martinelli and Nikolai Mushegian. 2019. Balancer Whitepaper. https://web.archive.org/web/20220623220539/https://balancer.fi/whitepaper.pdf

[56] Amani Moin, Kevin Sekniqi, and Emin Gun Sirer. 2020. SoK: A classification framework for stablecoin designs. In *International Conference on Financial Cryptography and Data Security.* Springer, Springer International Publishing, 174–197. https://doi.org/10.1007/978-3-030-51280-4_11

[57] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. https://web.archive.org/web/20100704213649/https://bitcoin.org/bitcoin.pdf

[58] Michael Neuder, Rithvik Rao, Daniel J. Moroz, and David C. Parkes. 2021. Strategic Liquidity Provision in Uniswap v3. *CoRR* abs/2106.12033 (2021). arXiv:2106.12033 https://arxiv.org/abs/2106.12033

[59] Alexandre Obadia, Alejo Salles, Lakshman Sankar, Tarun Chitra, Vaibhav Chellani, and Philip Daian. 2021. Unity is Strength: A Formalization of Cross-Domain Maximal Extractable Value. https://doi.org/10.48550/ARXIV.2112.01472

[60] Ariel Orda and Ori Rottenstreich. 2019. Enforcing Fairness in Blockchain Transaction Ordering. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, San Francisco, CA, USA, 368–375. https://doi.org/10.1109/BLOC.2019.8751349

[61] paco0x. 2021. AMM Arbitrageur. https://github.com/paco0x/amm-arbitrageur

[62] Julien Piet, Jaiden Fairoze, and Nicholas Weaver. 2022. Extracting Godl [sic] from the Salt Mines: Ethereum Miners Extracting Value.

[63] DeFi Pulse. 2021. Historical Data. https://defipulse.com.

[64] Kaihua Qin, Liyi Zhou, Pablo Gamito, Philipp Jovanovic, and Arthur Gervais. 2021. An Empirical Study of DeFi Liquidations: Incentives, Risks, and Instabilities. In *Proceedings of the 21st ACM Internet Measurement Conference* (Virtual Event) *(IMC '21)*. Association for Computing Machinery, New York, NY, USA, 336–350. https://doi.org/10.1145/3487552.3487811

[65] Kaihua Qin, Liyi Zhou, and Arthur Gervais. 2022. Quantifying Blockchain Extractable Value: How dark is the forest?. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*. IEEE, San Francisco, CA, USA, 198–214. https://doi.org/10.1109/SP46214.2022.9833734

[66] Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais. 2021. Attacking the DeFi Ecosystem with Flash Loans for Fun and Profit. https://doi.org/10.1007/978-3-662-64322-8_1 arXiv:2003.03810 [cs.CR]

[67] M. Rosenfeld. 2011. Analysis of Bitcoin Pooled Mining Reward Systems. arXiv:1112.4980 [cs.DC]

[68] Fabian Schär. 2021. Decentralized Finance: On Blockchain-and Smart Contract-based Financial Markets. *Available at SSRN 3571335* 103, 2 (April 2021), 153–174. https://doi.org/10.20955/r.103.153-74

[69] Hayden Shively and Adam Egyed. 2021. Nantucket. https://web.archive.org/web/20221012180650/https://github.com/haydenshively/Nantucket

[70] Andrei Shleifer and Robert W. Vishny. 1997. The Limits of Arbitrage. *The Journal of Finance* 52, 1 (1997), 35–55. https://doi.org/10.1111/j.1540-6261.1997.tb03807.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1997.tb03807.x

[71] SushiSwap. 2022. SushiSwap Docs. https://web.archive.org/web/20220901181030/https://docs.sushi.com/

[72] Adrian Sutton. 2022. Understanding Attestation Misses. https://web.archive.org/web/20220925110330/https://symphonious.net/2022/09/25/understanding-attestation-misses/

[73] Uniswap. 2020. Uniswap v2 Overview. https://web.archive.org/web/20220917011113/https://uniswap.org/blog/uniswap-v2

[74] Uniswap. 2021. Auto Router V2. https://web.archive.org/web/20211216202805/https://uniswap.org/blog/auto-router-v2

[75] Uniswap. 2022. Flash Swaps. https://web.archive.org/web/20220905142459/https://docs.uniswap.org/protocol/V2/guides/smart-contract-integration/using-flash-swaps

[76] Uniswap. 2022. Multihop Swaps. https://docs.uniswap.org/protocol/guides/swaps/multihop-swaps

[77] Uniswap. 2022. Uniswap v2 SDK. https://github.com/Uniswap/v2-sdk

[78] Friedhelm Victor and Bianca Katharina Lüders. 2022. Measuring Ethereum-Based ERC20 Token Networks. In *Financial Cryptography and Data Security* (Frigate Bay St. Kitts and Nevis). Springer-Verlag, Berlin, Heidelberg, 113–129. https://doi.org/10.1007/978-3-030-32101-7_8

[79] Fabian Vogelsteller and Vitalik Buterin. 2015. EIP-20: Token Standard. https://eips.ethereum.org/EIPS/eip-20

[80] Dabao Wang, Siwei Wu, Ziling Lin, Lei Wu, Xingliang Yuan, Yajin Zhou, Haoyu Wang, and Kui Ren. 2021. 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* (Virtual Event, Hong Kong) *(SBC '21)*. Association for Computing Machinery, New York, NY, USA, 23–28. https://doi.org/10.1145/3457977.3460301

[81] Ye Wang, Yan Chen, Haotian Wu, Liyi Zhou, Shuiguang Deng, and Roger Wattenhofer. 2021. Cyclic Arbitrage in Decentralized Exchanges. https://doi.org/10.48550/ARXIV.2105.02784

[82] Ben Weintraub, Christof Ferreira Torres, Cristina Nita-Rotaru, and Radu State. 2022. A Flash(Bot) in the Pan: Measuring Maximal Extractable Value in Private Pools. In *Proceedings of the 22nd ACM Internet Measurement Conference* (Nice, France) *(IMC '22)*. Association for Computing Machinery, New York, NY, USA, 458–471. https://doi.org/10.1145/3517745.3561448

[83] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.

[84] wow. 2020. AAVE Protocol. arXiv:https://git.io/JLQVx https://git.io/JLQVx

[85] Jiahua Xu, Nazariy Vavryk, Krzysztof Paruch, and Simon Cousaert. 2021. SoK: Decentralized Exchanges (DEX) with Automated Market Maker (AMM) protocols. arXiv:2103.12732 https://arxiv.org/abs/2103.12732

[86] Aviv Yaish, Gilad Stern, and Aviv Zohar. 2022. Uncle Maker: (Time)Stamping Out The Competition in Ethereum. https://ia.cr/2022/1020

[87] Aviv Yaish, Saar Tochner, and Aviv Zohar. 2022. Blockchain Stretching & Squeezing: Manipulating Time for Your Best Interest. In *Proceedings of the 23rd ACM Conference on Economics and Computation* (Boulder, CO, USA) *(EC '22)*. Association for Computing Machinery, New York, NY, USA, 65–88. https://doi.org/10.1145/3490486.3538250

[88] Aviv Yaish and Aviv Zohar. 2020. Correct Cryptocurrency ASIC Pricing - Are Miners Overpaying? arXiv:2002.11064

[89] YCHARTS. 2022. Ethereum Average Block Time. https://web.archive.org/web/20221009031833/https://ycharts.com/indicators/ethereum_average_block_time

[90] yearn. 2022. yearn-vaults/contracts/BaseStrategy.sol. https://github.com/yearn/yearn-vaults/blob/efb47d8a/contracts/BaseStrategy.sol#L539

[91] Liyi Zhou, Kaihua Qin, Antoine Cully, Benjamin Livshits, and Arthur Gervais. 2021. On the Just-In-Time Discovery of Profit-Generating Transactions in DeFi Protocols. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, San Francisco, CA, USA, 919–936. https://doi.org/10.1109/SP40001.2021.00113

[92] Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc Viet Le, and Arthur Gervais. 2021. High-Frequency Trading on Decentralized On-Chain Exchanges. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, San Francisco, CA, USA, 428–445. https://doi.org/10.1109/SP40001.2021.00027

[93] Liyi Zhou, Xihan Xiong, Jens Ernstberger, Stefanos Chaliasos, Zhipeng Wang, Ye Wang, Kaihua Qin, Roger Wattenhofer, Dawn Song, and Arthur Gervais. 2022. SoK: Decentralized Finance (DeFi) Attacks. https://doi.org/10.48550/ARXIV.2208.13035

A ATTACK SUBOPTIMALITY

In this section we show that even savvy DeFi attackers do not always execute optimal attacks.

Case Study 8 (DeFi Attack Suboptimality: Inverse Finance). *A transaction with hash* $0x958\ldots13c$ *began propagating on Ethereum's p2p network at Thursday,* 16^{th} *of June 2022, about 9am Central European Time. The transaction contained code for a DeFi attack, which entailed requesting a* 0.5 *Billion USD flash-loan [66] that was intended to be used by an attacker to extract a profit of* 1.2 *Million USD from its victim.*

An automated arbitrage "bot" which was listening to Ethereum's p2p network noticed this transaction, which was not yet included in a block. This bot realizes that the transaction's significant financial volume opens up an arbitrage opportunity that was overlooked by the attacker and which can be used by the bot to make a profit. Thus, the arbitrageur bundles the attack together with an arbitrage transaction with hash $0xfa1\ldots6bd$*, and submits the bundle to a front-running-as-a-service (FaaS) provider, while paying a bribe of* $90k$ *USD to the winning miner.*

Knowingly or not, the arbitrageur as well as the FaaS helped execute the attack, while also rewarding the miner with a reward exceeding the average Ethereum mining rewards from block rewards and transaction tips [54] by more than 4000%.

Related work has shown how such a MEV opportunity would incentivize even a 2% *hash rate miner to fork the chain and destabilize the consensus mechanism [92]. Note that over* 50% *of Ethereum mining power is held by* 4 *mining pools, each holding more than* 8% *of all mining power [86].*

B GLOSSARY

A summary of all symbols and acronyms used in the paper.

SYMBOLS

ac	The collateral acquired by a liquidator.
b	Total amount of borrowed funds, denoted in tokens.
cf	The debt that can be repaid within a single FSL, also known as the close factor.
d	Total amount of deposited funds, denoted in tokens.
$\mathbb{D}$	A DEX.
hf	The health of a debt position, or how close it is to insolvency.
I	The yearly interest-rate.
I_b	The yearly interest-rate for taking liquidity.
I_d	The yearly interest-rate for supplied liquidity.
ls	The liquidation spread.
lt	Liquidation threshold, defined to be $\in (0,1)$.
p	The United States Dollar (USD) price of a token.
rd	The amount of debt that a liquidator is repaying.
r	The reserve factor.
u	The utilization of the liquidity pool.

ACRONYMS

AMM	automated market maker
APY	annual percentage yield
CFMM	constant function market maker
CPAMM	constant product automated market maker

DeFi	decentralized finance
DEX	decentralized exchange
ERC	Ethereum request for comments
EVM	Ethereum virtual machine
FaaS	front-running-as-a-service
FSL	fixed spread liquidation
LP	liquidity provider
MEV	miner-extractable value
p2p	peer to peer
PoS	proof-of-stake
PoW	proof-of-work
TVL	total value locked
USD	United States Dollar
w.r.t.	with regards to
WBTC	Wrapped Bitcoin
WETH	Wrapped Ethereum