Suboptimality in DeFi

Aviv Yaish[1], Maya Dotan[1], Kaihua Qin[2,4], Aviv Zohar[1], and Arthur Gervais[3,4]

[1] The Hebrew University
[2] Imperial College London
[3] University College London
[4] Berkeley RDI

Abstract The decentralized finance (DeFi) ecosystem has proven to be popular in facilitating financial operations, such as token exchange and lending. The public availability of DeFi platforms' code, together with real-time data on all user interactions with them, has given rise to complex tools that find and seize profit opportunities on behalf of users. In this work, we show that both users and the aforementioned tools sometimes act suboptimally. In specific instances which we examine, their profits can be increased by more than 100%, with the highest amount of missed revenue by a suboptimal action reaching 428.14ETH ($517K).
To reach these findings, we examine core DeFi primitives which are responsible for a daily volume of over 100 million USD in Ethereum alone: (1) lending and borrowing funds, (2) using *flash*swaps to close arbitrage opportunities between decentralized exchanges (DEXs), (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 show that missed opportunities to make a profit are noticed by others, and are sometimes followed by back-running transactions which extract profits using similar actions. By analyzing these events, we find that some transactions are circumstantially tied to specific miners, and hypothesize they use their knowledge of private orderflow for a profit. Essentially, this is an instance of miner-extractable value (MEV) "in action".

Keywords: Cryptocurrency, Blockchain, DeFi, Miner Extractable Value.

1 Introduction

DeFi is an umbrella term for financial platforms that run as smart contracts on cryptocurrencies, covering use-cases such as exchanging tokens, to lending and borrowing [51]. While on-chain DeFi services appear similar to their off-chain equivalents, the optimal usage of each differs due to design considerations which are required for the secure and efficient implementation of financial services in the decentralized and pseudonymous setting which DeFi inhabits.

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

(1) 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 where the user was to leave its funds unchanged.

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 their profits can be increased by up to 700%. Furthermore, we go over empirical evidence which suggests that currently, market forces are slow to react to shocks, such as when a large amount of deposits is withdrawn in a single transaction. Thus, even myopically optimizing actions can produce long-lasting profits.

(2) Flashswap-based arbitrage. If an asset has different prices in two or more markets, actors can take advantage of the price difference to make a profit by simultaneously buying and selling the asset in these markets, a practice 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 initial funds, using the profits of the arbitrage instead.

The search space for arbitrage opportunities in Ethereum is large, owing to the number of DEXs currently active on it, and the 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 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 $\$517K$.

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 regard to their sources of revenue. Surprisingly, we find 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.

(3) Liquidation of insolvent loans using flashswaps. This primitive combines the previous two. Lending platforms allow users to repay under collateralized debt positions and receive the collateral at a discount. The amount of debt which can be repaid is usually limited by some *close factor*, i.e., a close factor of 50% permits repaying up to half of the debt position. Tools that automatically act upon potentially profitable liquidation opportunities commonly repay the maximal amount of debt possible, and use flashswaps to do so.

We show that this liquidation strategy is suboptimal, and provide a better one. Our strategy 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-rates between debt and collateral assets.

Our Contributions. Our contributions can be summarized like so:

- **Optimal Execution**. We examine three core DeFi primitives: collateralized lending, flashswaps and flashswap-based liquidations. The execution of each is cast as an optimization problem. To the best of our knowledge, we are the first to optimize these primitives, and examine flashswaps.
- **Evaluation of Suboptimal Cases**. Our optimal execution strategies are used to find and evaluate multiple case studies showing the suboptimal behavior of significant DeFi platform actors. We show that in some cases, their revenue can 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. A summary of our findings is given in Table 1.
- **Longitudinal Study of Suboptimal Arbitrage Flashswaps**. We devise a heuristic to detect suboptimal arbitrage transactions which rely on flashswaps and find over 10K such instances, which combined have missed revenue in excess of more than 4 Million USD. Furthermore, we find circumstantial evidence tying the Ethermine mining pool to an address which capitalized on suboptimal transactions, suggesting that miners might be using or sharing "inside information" for benefit, 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 used to manage over 400 Million USD, such as Yearn Finance [18], are suboptimal. We find that, surprisingly, such suboptimality can benefit certain users. For example, borrowers benefit from the suboptimality of liquidity providers (LPs).

2 Background

Blockchains. Cryptocurrencies like Ethereum [9, 66] facilitate financial *transactions* between users. Transactions are collected in a ledger comprised of *blocks*,

Table 1: A summary of our case studies.

Case	"Lost" Profits	Action Type
Case Study 4.2: $0x5e1...6c5$	More than 94%	Flashswap
Case Study 4.3: $0x0f4...74b$	More than 7000%	Flashswap
Case Study 3.2: Justin Sun	More than 8.7%	Lending
Case Study 3.3: $0x7a1...428$	More than 700%	Lending

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. As blocks are size-limited, users compete for block-space and can offer *fees* to prioritize their transactions [33]. Most cryptocurrencies operate in a decentralized manner and rely on pseudonymous users called *miners* to create blocks, a process also called *mining*. In proof-of-stake (PoS) blockchains such as Ethereum, these entities are also known as *proposers*; for brevity and generality, we use the term miners. Miners, who are supposed to maintain the system's integrity, can increase their revenue by deviating from the mining protocol, or by manipulating transaction order [15]. The value earned from these exploits is termed *MEV*.

Lending Platforms. Certain DeFi platforms, like Aave and Compound, let users take and give loans [72]. Commonly, platforms rely on user-provided liquidity for their operation, with funds collected in *liquidity pools*, and with users who supply funds called *LPs*. 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 upfront 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 [47].

Decentralized Exchanges. DEXs are DeFi platforms that enable the exchange, or *swap*, of tokens [69]. Platforms like Uniswap [60] allow swaps between a pair of tokens to be performed using user-provided liquidity which is collected in liquidity pools, 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 [57, 59].

ERC20 Tokens. Ethereum specifies the ERC20 standard, that lets contracts create and interact with fungible tokens known as ERC20 tokens [23, 61]. DeFi platforms may furthermore create ERC20 tokens for protocol-specific versions of other ERC20 tokens they support. For example, Compound created *cUSDT*, a platform-integrated version of the USDT token [12]. By interacting with cUSDT's

contract, users can deposit and withdraw USDT to and from Compound. In our work, we examine case studies involving such tokens, specifically the USDT and USDC tokens (which strive to maintain a one-to-one peg with the USD [41]), Wrapped Ethereum (WETH) (a "wrapped" version of the Ethereum token designed to enable easier interoperability between contracts and the token [10]), Wrapped Bitcoin (WBTC), and CRV [14].

3 Lending Suboptimality

Aave [67] and Compound [37] are collateralized lending Ethereum. At the time of writing, both hold a combined amount of 6.2 Billion USD [17], making them the most popular lending platforms on Ethereum. To incentivize liquidity provisioning by users, lending platforms give interest on funds stored in their pools [34]. Although interest-accretion is usually associated with lending, any platform that rewards liquidity provisioning in a proportional manner is essentially paying interest. A notable example is Tornado Cash and other mixers who offer rewards long-term liquidity provisioning [62].

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

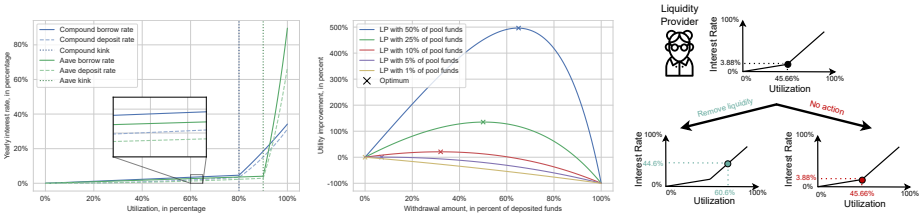
3.1 Utilization-based Interest Rate Schemes

Utilization. The lending mechanism used by Compound and Aave relies on the *utilization* metric, which we denote by u. 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 defined as: $u \stackrel{\text{def}}{=} \frac{b}{d}$.

Reserve Factor. When the utilization is close to 1, a lending pool may be vulnerable in case of an economic shock or borrower insolvency. To safeguard against such events, lending pools commonly reserve a fraction r of the interest paid by borrowers; this fraction is termed the *reserve factor*.

Kinked Utilization-based Interest-rate. 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 [41], uses a *kinked*-curve interest-rate scheme [31, 34]. In this type of scheme, the rates for both deposits and borrows are increasing in the utilization of the 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. 1a, hence the name.

Interest-rate Curves. We now characterize the interest-rate schemes commonly used by collateralized lending platforms, like Compound. Let the baseline interest-rate be I_0 and the slopes before and after u_{opt} be I_1 and I_2 respectively. The per-block *interest on borrows* is: $I_b(u) \stackrel{\text{def}}{=} I_0 + \min(u, u_{opt}) \cdot I_1 +$



(a) Compound's and Aave's APY on USDT 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.

(c) Counter intuitively, the LP presented in Case Study 3.3 could increase its per-block profit by 700% by *withdrawing* 38 Million CRV from the pool.

Figure 1: Optimal liquidity provisioning strategies may not rely on depositing all funds, as this can *lower* profits.

$(\max(u, u_{opt}) - u_{opt}) \cdot I_2$. The per-block *interest for supplying liquidity* is defined as: $I_d(u) \stackrel{\text{def}}{=} u \cdot (1-r) \cdot I_b(u)$. Real-world examples of such curves are depicted in Fig. 1a. Note that our characterization is for the per-block interest, while the depicted examples extrapolate the per-block interest into the annualized interest, also known as the annual percentage yield (APY).

3.2 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. Crucially, the latter amount can be manipulated by LPs.

We now generalize this observation to any liquidity pool relying on utilization-based mechanisms. By inputting the pool's parameters, the optimal amount to deposit is found by solving the optimization problem written in Eq. (1).

$$\begin{aligned} \underset{d^*}{\text{maximize}} \quad & d^* \cdot I_d(u) \\ \text{subject to} \quad & d^* \geq 0 \\ & d^* \leq d_{max} \\ & u = \frac{b}{d + d^*} \end{aligned} \tag{1}$$

Remark 3.1 *Eq. (1) considers a one-shot setting. 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 the last in the block [15], or by sending the transaction as part of a bundle [7].*

In case that the actor's liquidity is large compared to others', Fig. 1b 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 few independent actors [34], meaning that such considerations are relevant.

Furthermore, empirical data suggests that the market is slow to react to liquidity shocks. Notably, Aave's CRV 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 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-one 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 restrictions were lifted, the actor continued performing actions of a similar magnitude.*

We further note that the impact of such manipulations can be extended to multiple blocks by relying on other techniques which are outside the scope of this work, such as mempool Denial-of-Service attacks which allow an attacker to discard transactions from miner's queue of pending transactions, thereby possibly postponing their acceptance to a block [38, 70].

3.3 Suboptimal Lending

Fig. 1b shows that the optimal action given by Eq. (1) for a LP 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%.

We now share notable examples of suboptimal capital allocation to interest-bearing pools by large LPs. We begin by showing in Case Study 3.1 that Yearn, the largest investment platform on the Ethereum blockchain, is suboptimal.

Case Study 3.1 (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 [13, 72]. Yearn is currently the largest on-chain yield aggregator on the Ethereum blockchain [19], with a total value locked (TVL) of* $400 *Million at the time of writing (and reaching over* $6.5 *Billion at its peak) [18].*

Yearn follows the "all-in" approach by comparing interest rates across platforms, and depositing all funds in the platform offering the best rates [28, 74]. As we have shown, this "all-in" approach is, in fact, suboptimal.

Case Study 3.2 covers a specific instance of a user performing suboptimally.

Case Study 3.2 (Justin Sun) *Compound's largest LP for the cUSDT token at block* 13632753 *was address* 0*x*3*dd*...296*, associated with Justin Sun [27], 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.

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

In Case Study 3.3, also depicted in Fig. 1c, we cover a more dramatic case.

Case Study 3.3 (Account 0*x*7*a*1...428**)** *Address* 0*x*7*a*1...428 *had a deposit of* 132.9 *Million CRV in Aave's liquidity pool at block* 15529008, *amounting to almost* 85% *of the pool's liquidity at the time. 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%.

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

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 [54] and Balancer [40], which together comprise roughly 75% of the daily volume of all token exchanges in Ethereum [16]. We note that Uniswap v1 is considered as a proof-of-concept by its creators [56], 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 Decentralized Exchanges

Arbitrage. 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 [53], where users can generate a net profit by trading across platforms with different exchange-rates [64, 75], these users are oftentimes referred to as *arbitrageurs*. The act of profiting from arbitrage is called *closing* the arbitrage. Arbitrage opportunities are prevalent in today's DeFi ecosystem [15], with arbitrageurs making an average monthly profit of more than 12 Million USD [64].

Flashswaps. Flashswaps [58] 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, 29, 56].

Flashswaps let users swap between two currencies without having the initial liquidity requirements upfront, and work similarly to flashloans [15]. Generally, given that a user wants to obtain a c amount of token $\mathcal{C}$ to perform an action (e.g., perform arbitrage), it has to:

1. Write and deploy a smart contract which has a *callback* function that receives this amount and executes the wanted action.
2. Call a DEX's *flashswap* function, passing c and the callback as arguments.

Then, given that the DEX has more than a c amount of token $\mathcal{C}$, 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 $\mathcal{C} \leftrightarrow \mathcal{D}$ 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 [58]. If the user's callback function did not satisfy these conditions, the transaction is reverted. To prevent spamming by users, reverted transactions still have to pay transaction fees [70].

To further illustrate how flashswaps work, in Example 4.1 and Fig. 2 we provide a real-world example and a corresponding depiction.

Example 4.1 *In transaction 0x9ae...bc4, the address 0x5e1...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 3.6*K 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,* 3.1*K 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* 600*K USD, according to the rate at the time.*

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.

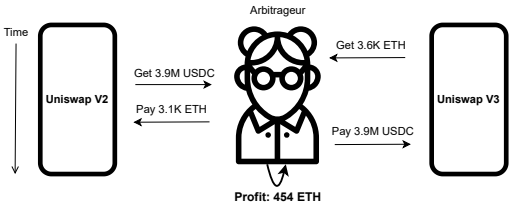


Figure 2: A depiction of Example 4.1.

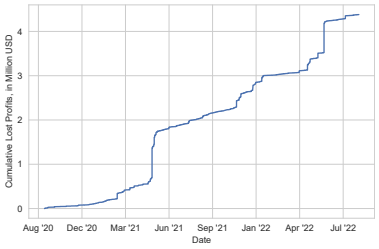


Figure 3: Cumulative lost profits by suboptimal flashswaps.

4.2 Flashswap Arbitrage Suboptimality

We now turn to quantify the suboptimality of arbitrageurs. Although outside the scope of the section, in Appendix A.1 we provide an optimization problem for the task of devising optimal flashswap arbitrage transactions. Intuitively, even without viewing the optimization problem, there are many tokens and DEXs which can be used to capitalize on arbitrage opportunities, thus the search space for the problem is huge. Furthermore, the optimal solution also depends on the current state of the blockchain, which changes every 12 seconds [22, 73].

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 [21, 55], 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. As such, it is not surprising that existing tools and works do not attempt to perform an exhaustive search.

Case Study 4.1 (Arbitrage Tools) *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 to obtain profits.*

Zhou et al. *[75] formulated and implemented an algorithm that detects profitable arbitrage opportunities across a variety of DEXs and tokens. The authors noted that their tool obtains results within an average of* 5.39 *seconds, even though it does not perform an exhaustive search and thus is not optimal. The tool's runtime was achieved by (1) applying heuristics to trim the search-space (thereby potentially losing in profits), and (2) running on considerably powerful hardware.*

Popular open-source tools further trim the search space by limiting themselves to single-hop swaps [2, 45], which can under-perform relative to multi-hop ones.

Longitudinal Study. We devise a simple yet effective heuristic in Definition 4.1 to uncover suboptimal flashswaps from blockchain data and use this heuristic to perform a longitudinal study of arbitrage suboptimality. At each block, our

heuristic searches for sequences of consecutive flashswap-based arbitrage transactions, where the profit made by each transaction could have been made by the one preceding it, thereby meaning that preceding transactions are suboptimal.

Definition 4.1 (Suboptimal arbitrage heuristic). *Let τ_1, τ_2 be two consecutive transactions contained within the same block, sorted by their order. We call τ_1 a* suboptimal arbitrage transaction *if τ_1 relies on swaps to profit off of arbitrage opportunities, and transaction τ_2 solely relies on* flash*swaps to do so.*

In Definition 4.1, as τ_2 only utilized flashswaps to perform the arbitrage, then transaction τ_1 could have imitated the actions taken by τ_2, thereby making the same profit. Thus, τ_1 was suboptimal. We generalize this notion in Definition 4.2.

Definition 4.2 (Suboptimal sequence heuristic). *Let $\tau_1, \ldots, \tau_n$ be a sequence of consecutive transactions contained within the same block. We call this a* suboptimal arbitrage sequence *if $\forall i < n$, transaction τ_i is suboptimal.*

Results. We now provide evidence that suboptimal behavior is prevalent. By applying our heuristics to the time period starting at block 10749295 and ending at 15450669, we discovered over 9875 suboptimal transactions which fit Definition 4.1, with a total lost profit of over 4.38 Million USD. Over the same period, an average of 2.91 ETH were lost per day due to suboptimality. 0.2% of blocks contain at least one suboptimal sequence, where the longest is 7 transactions long, all contained in block 15243809, starting at transaction 209. The cumulative lost profits over time are depicted in Fig. 3.

Using our heuristic, we discovered interesting case studies. For example, Case Study 4.2 shows that even arbitrageurs who are proficient enough to make large profits are not always optimal.

Case Study 4.2 (Account $0x9ae...bc4$) *Recall the transaction covered in Example 4.1. Although it produced a net profit of 454 ETH (equal at the time to 600K USD), it is, in fact, not optimal. Afterward, came a second transaction, with hash $0x580...eff$. 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. This transaction was able to extract an extra 428 ETH in revenue, equal to \$517K, and 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 4.3.

Case Study 4.3 (Account $0x0f4...74b$) *In transaction $0xb67...4a6$, the user $0x0f4...74b$ 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 when considering*

the exchange-rate at the time (note we refer to USD, not to the USDC token). Although not to be sneezed at, this is suboptimal.

This was followed by transaction $0xc07...a2b$*. This transaction followed in the footsteps of the previous one, and involves 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*!*

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: $0x000...f56$ with 13K such transactions, while both $0x860...f66$ and $0xe33...c85$ are tied with 6K transactions each. Similarly, $0x911...116$ and $0x584...dba$ 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 [24, 32], with the intention of preventing transaction front-running or other potential attacks. Such channels, also known as *private relays*, are widely adopted by arbitrageurs and miners alike [48].

In this aspect, arbitrageur $0x584...dba$ should be particularly noted: circumstantial evidence suggests it is possibly tied to the *Ethermine* mining pool [50], which was the largest before Ethereum's transition to PoS [71]. Specifically:

1. All the arbitrageur's transactions appeared solely in Ethermine's blocks.
2. The arbitrageur's first and last transactions correspond to the times at which Ethermine launched and shut-down its private channel [25, 26].

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

We raise the possibility that the success of $0x584...dba$ in identifying unexploited arbitrage is due to "inside information" it has from ties to a private relay, in the form of access to private transactions. This information can be used, for example, to trim the large arbitrage search-space. Indeed, in cases found by our heuristics, neighboring transactions use almost identical operations, usually only differing in the DEXs used (e.g., Case Study 4.2 and Case Study 4.3).

Discussion. Suboptimal arbitrage transactions might leave arbitrage opportunities open. As arbitrageurs are actively seeking out such opportunities [48, 64], 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 [39], leading to higher transaction fees for all users.

5 Liquidation Suboptimality

In this section, we analyze the suboptimality of common liquidation strategies, primarily of those who rely on flashswaps (or equivalently, on flashloans followed

by ordinary swaps), and thus this section combines the previous two. In lending platforms, LPs bear the risk that borrowers may default. A default can happen if the collateral asset cannot cover the loan, for example due to a drop in the asset's exchange-rate. Platforms commonly define a *liquidation threshold*, below which other users can buy, or *liquidate*, a debt position, while receiving the collateral at a discount. This discount is called the liquidation *spread*.

5.1 Fixed Spread Liquidation

Currently, fixed spread liquidation (FSL) is the prevalent method for securing LP capital [47]. On a high level, FSL incentivizes so-called liquidators to repay the outstanding debt of a borrower. In return, the liquidator is allowed to acquire pro rata collateral from the borrower. By design, the 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.

Notations. We assume a debt position collateralized by cryptocurrency $\mathcal{C}$ with an outstanding debt in cryptocurrency $\mathcal{D}$, and denote the amount of collateral and debt by c and δ respectively. In practice, a debt position can have multi-cryptocurrency collateral (debt). For simplicity, we assume that the collateral (debt) is in a single cryptocurrency. We apply the USD price of $\mathcal{C}$ and $\mathcal{D}$ to unify financial value calculation, denoted by $p_{\mathcal{C}}$ and $p_{\mathcal{D}}$ respectively.

Health Factor. To measure the "health" of a debt position, meaning how close it is to insolvency, platforms use a metric called the *health factor*. Given a liquidation threshold $0 < \tau < 1$, the health factor is defined as: $\eta \stackrel{\text{def}}{=} \frac{p_{\mathcal{C}} \cdot c}{p_{\mathcal{D}} \cdot \delta} \cdot \tau$. When the collateral value decreases because of, for example, price decline (i.e., $\frac{p_{\mathcal{C}}}{p_{\mathcal{D}}} \downarrow$), η decreases, indicating that the borrower is more likely to default.

Liquidation Spread. Once η falls below one, the borrowing position is available for liquidations. Any liquidator is then allowed to repay ϱ 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 σ.

Liquidation Profit. The collateral acquired by the liquidator within this liquidation is denoted by α and equals $\alpha \stackrel{\text{def}}{=} \frac{p_{\mathcal{D}} \cdot (1+\sigma)}{p_{\mathcal{C}}} \cdot \varrho$. For brevity, we define the following: $p_{\text{liq}} \stackrel{\text{def}}{=} \frac{p_{\mathcal{D}} \cdot (1+\sigma)}{p_{\mathcal{C}}}$, thereby we can simplify the acquired collateral to $\alpha = p_{\text{liq}} \cdot \varrho$. Without considering fees, the liquidator's profit is formulated as: $\text{profit}^t(\varrho) = p_{\mathcal{C}} \cdot \alpha - p_{\mathcal{D}} \cdot \varrho = \sigma \cdot p_{\mathcal{D}} \cdot \varrho$.

Close Factor. Platforms often constrain the fraction of debt that a liquidator is allowed to repay within a single FSL by a *close factor* κ, defined as: $\kappa \geq \frac{\varrho}{d}$.

Flashloans and Flashwaps. 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, flashloans [49] have become a popular option for liquidators [36, 63]. Exactly in the same manner, these risks can be avoided by using flashswaps. With a flashswap, a liquidator can borrow assets in $\mathcal{D}$ to repay liquidated debt and acquire collateral in $\mathcal{C}$, with the latter user to repay the flashswap, all within the span of a single transaction. We emphasize that due to the equivalence between flashwaps and combining a flashloan and a swap, the results contained within this section are applicable to both liquidation schemes.

Constant Product Rule. We focus on flashswaps obtained from DEXs that use the constant product rule, a prevalent DEX pricing formula [76]. Such DEXs are referred to as constant product automated market makers (CPAMMs). A CPAMM contract reserves a pair of tokens, allowing any trader to trade against it by sending one cryptocurrency to the contract and, in exchange, taking the other cryptocurrency from the contract. The constant product rule, as the name suggests, requires that such trades preserve the product of the amounts of each asset held before the trade. More precisely, assume a CPAMM with x_c of $\mathcal{C}$ and x_d of $\mathcal{D}$ reserved, thus by the definition of the constant product rule their product $x_c \cdot x_d$ should remain constant after exchanges. Thus, the amount of $\mathcal{D}$ tokens received in a swap for providing a Δ amount of $\mathcal{C}$ is: $\text{Swap}^{\mathcal{C}\to\mathcal{D}}(\Delta) \stackrel{\text{def}}{=} x_d - \frac{x_c \cdot x_d}{x_c + \Delta}$.

5.2 Optimal Flashswap-based Liquidation

Presumably, to optimize liquidation profits, a liquidator should liquidate up to the close factor constraint, because $\text{profit}^t(\varrho)$ increases monotonically with regards to (w.r.t.) ϱ. However, this intuition does not hold when a CPAMM is used to perform an asset exchange from $\mathcal{C}$ to $\mathcal{D}$ in the liquidation process, due to the effect large trades have on CPAMM exchange-rates. This insight is crystallized in Theorem 5.1, with the proof given in Appendix A.

Theorem 5.1 (Optimal liquidation). *Consider a debt position that is available for liquidation, where the debt is in cryptocurrency $\mathcal{D}$, and is collateralized by funds in cryptocurrency $\mathcal{C}$. If a user wishes to perform the liquidation using a swap obtained from a CPAMM with x_c of $\mathcal{C}$ and x_d of $\mathcal{D}$ reserved, then the optimal amount to repay is:* $\varrho^* = \min\left(\kappa \cdot \delta, \frac{\sqrt{p_{liq} \cdot x_c \cdot x_d} - x_c}{p_{liq}}\right)$.

The optimal strategy presented in Theorem 5.1 can improve profits compared to a naïvely liquidating up to the close factor, even when liquidating relatively small loans. To illustrate, in Fig. 5 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. 4. 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.

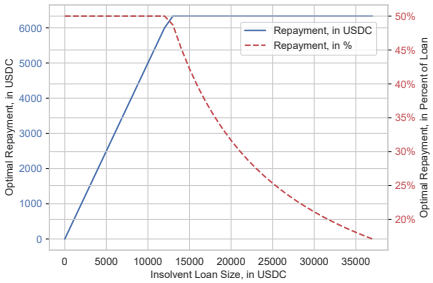


Figure 4: Optimal repayment in a swap-based liquidation for Aave's USDC pool, that has a close factor of 50%. Due to exchange-rate slippage, it is suboptimal to repay large illiquid positions up to the close factor.

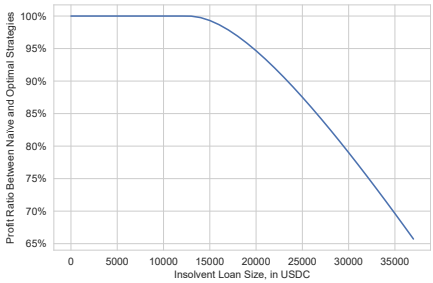


Figure 5: The ratio between profits made by naïve and optimal flashswap based liquidations for Aave's USDC pool. The ratio drops below 1 for large loans, indicating that the naïve strategy is suboptimal.

Case Study 5.1 (Flash Liquidation Tools) *Popular tools which automatically look for insolvent debt positions and liquidate them using flashwaps (or flashloans and swaps), do not follow the optimal strategy we present in Theorem 5.1, instead performing liquidations up to the close factor [1, 20, 52].*

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 perform a service for the platform, driven by the possibility of obtaining a debtor's collateral at a discount. For large positions, the funds required for liquidation can be substantial. Users who lack such funds are required to rely on actions such as flashloans and swaps. As we have shown, in such cases utility-maximizing users may prefer to repay only a small amount of debt, possibly not enough to pull the position's health factor back to a reasonable level according to the mechanism's risk parameters. Thus, the health of the platform is potentially harmed.

6 Conclusion

In this work, we study three core DeFi primitives: lending, flashswaps, and flashswap-based liquidations. We derive their optimal usage, and show that even large platforms and market actors behave 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, through our longitudinal study, we uncover the first instance of a miner using inside information to its benefit. We hope our work draws attention to under-explored aspects of DeFi which are important for the integrity and efficient operation of platforms.

References

1. 0xnivek. Joe liquidator, 2021. URL: https://web.archive.org/web/20221012180624/https://github.com/0xnivek/joe-liquidator.
2. 6eer. uniswap-sushiswap-arbitrage-bot, 2021. URL: https://github.com/6eer/uniswap-sushiswap-arbitrage-bot.
3. Aave. Liquidations, 2022. URL: https://web.archive.org/web/20221013175355/https://docs.aave.com/developers/guides/liquidations.
4. Hayden Adams, Noah Zinsmeister, and Dan Robinson. Uniswap v2 core, 2020. URL: https://web.archive.org/web/20220126073458/https://uniswap.org/whitepaper.pdf.
5. Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. Uniswap v3 core, 2021. URL: https://web.archive.org/web/20221011013240/https://uniswap.org/whitepaper-v3.pdf.
6. Ether Alpha. Client diversity, December 2022. URL: https://web.archive.org/web/20221207190225/https://clientdiversity.org/#distribution.
7. Guillermo Angeris, Alex Evans, and Tarun Chitra. A note on bundle profit maximization, 2021.
8. Guillermo Angeris, Alex Evans, Tarun Chitra, and Stephen Boyd. Optimal routing for constant function market makers. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, EC '22, page 115–128, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3490486.3538336.
9. Vitalik Buterin. Ethereum whitepaper, July 2022. URL: https://web.archive.org/web/20220728020709/https://ethereum.org/en/whitepaper/.
10. Giulio Caldarelli. Wrapping trust for interoperability: A preliminary study of wrapped tokens. *Information*, 13(1):6, 2021.
11. Yixin Cao, Chuanwei Zou, and Xianfeng Cheng. Flashot: A snapshot of flash loan attack on defi ecosystem, 2021. URL: https://arxiv.org/abs/2102.00626, doi:10.48550/ARXIV.2102.00626.
12. Compound. ctokens, 2022. URL: https://compound.finance/docs/ctokens.
13. Simon Cousaert, Jiahua Xu, and Toshiko Matsui. Sok: Yield aggregators in defi, May 2021. arXiv:2105.13891.
14. Curve. Curve dao, 2021. URL: https://web.archive.org/web/20210811065239/https://curve.fi/files/CurveDAO.pdf.
15. Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. 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*, pages 910–927, San Francisco, CA, USA, 2020. IEEE. doi:10.1109/SP40000.2020.00040.
16. DefiLlama. Dexs ethereum volumes, 2022. URL: https://web.archive.org/web/20221011072608/https://defillama.com/dexs/ethereum.
17. DefiLlama. Lending tvl rankings, 2022. URL: https://web.archive.org/web/20221011073058/https://defillama.com/protocols/lending/Ethereum.
18. DeFiLlama. Yearn finance: Tvl and stats, 2022. URL: https://web.archive.org/web/20221013234041/https://defillama.com/protocol/yearn-finance.
19. DefiLlama. Defi dashboard, 2022 (accessed October 9, 2022). URL: https://defillama.com.
20. Mike De'Shazer. Flashloanliquidation, 2020. URL: https://web.archive.org/web/20221012180627/https://github.com/mikedeshazer/FlashLoanLiquidation.

21. Ben Edgington. Upgrading ethereum | one page annotated spec, December 2022. URL: https://web.archive.org/web/20221218133524/https://eth2book.info/bellatrix/annotated-spec/#seconds_per_slot.
22. Ethereum. Blocks, 2022. URL: https://web.archive.org/web/20220922171539/https://ethereum.org/en/developers/docs/blocks/#block-time.
23. Ethereum.org. Erc-20 token standard, 2021. URL: https://ethereum.org/en/developers/docs/standards/tokens/erc-20/.
24. ethermine.eth. Want to keep your dex trades away from the public mempool? we are proud to announce the ethermine private rpc endpoint., 2021. URL: https://web.archive.org/web/20221013133550/https://nitter.it/ethermine_org/status/1443502516604477445.
25. ethermine.eth. 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., 2022. URL: https://web.archive.org/web/20221013230053/https://nitter.it/ethermine_org/status/1570302744992583681.
26. ethermine.org. We are proud to announce the next step of #ethermine mev beta ethermine mev relay!, 2021. URL: https://web.archive.org/web/20221019190843/https://nitter.it/ethermine_org/status/1404464604663713798?lang=en.
27. Etherscan. Compound usdt (cusdt) token tracker, 2022. URL: https://etherscan.io/token/0xf650c3d88d12db855b8bf7d11be6c55a4e07dcc9#balances.
28. Etherscan. Contract 0x83f798e925bcd4017eb265844fddabb448f1707d, 2022. 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. URL: https://etherscan.io/address/0x83f798e925bcd4017eb265844fddabb448f1707d#code#L682.
29. Etherscan. Contract 0xcbcdf9626bc03e24f779434178a73a0b4bad62ed, 2022. URL: https://etherscan.io/address/0xcbcdf9626bc03e24f779434178a73a0b4bad62ed#code#F1#L777.
30. Zhou Fan, Francisco Marmolejo-Cossío, Ben Altschuler, He Sun, Xintong Wang, and David C. Parkes. Differential liquidity provision in uniswap v3 and implications for contract design, 2022. URL: https://arxiv.org/abs/2204.00464, doi:10.48550/ARXIV.2204.00464.
31. Compound Finance. Usdt market, 2022. URL: https://compound.finance/markets/USDT.
32. Flashbots. Flashbots, 2022. URL: https://github.com/flashbots/pm.
33. Yotam Gafni and Aviv Yaish. Greedy transaction fee mechanisms for (non-)myopic miners, 2022. URL: https://arxiv.org/abs/2210.07793, doi:10.48550/ARXIV.2210.07793.
34. Lewis Gudgeon, Sam Werner, Daniel Perez, and William J. Knottenbelt. 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*, pages 92–112. ACM, 2020. doi:10.1145/3419614.3423254.
35. Kshitij Kulkarni, Theo Diamandis, and Tarun Chitra. Towards a theory of maximal extractable value i: Constant function market makers, 2022. URL: https://arxiv.org/abs/2207.11835, doi:10.48550/ARXIV.2207.11835.
36. Alfred Lehar and Christine A Parlour. Systemic fragility in decentralized markets, 2022.

37. Robert Leshner and Geoffrey Hayes. Compound: The money market protocol, 2019.
38. Kai Li, Yibo Wang, and Yuzhe Tang. Deter: Denial of ethereum txpool services. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 1645–1667, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3460120.3485369.
39. Yulin Liu, Yuxuan Lu, Kartik Nayak, Fan Zhang, Luyao Zhang, and Yinhong Zhao. 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*, CCS '22, page 2099–2113, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3548606.3559341.
40. Fernando Martinelli and Nikolai Mushegian. Balancer whitepaper, 2019. URL: https://web.archive.org/web/20220623220539/https://balancer.fi/whitepaper.pdf.
41. Amani Moin, Kevin Sekniqi, and Emin Gun Sirer. Sok: A classification framework for stablecoin designs. In *International Conference on Financial Cryptography and Data Security*, pages 174–197. Springer, Springer International Publishing, 2020. doi:10.1007/978-3-030-51280-4_11.
42. Michael Neuder, Rithvik Rao, Daniel J. Moroz, and David C. Parkes. Strategic liquidity provision in uniswap v3. *CoRR*, abs/2106.12033, 2021. URL: https://arxiv.org/abs/2106.12033, arXiv:2106.12033.
43. Alexandre Obadia, Alejo Salles, Lakshman Sankar, Tarun Chitra, Vaibhav Chellani, and Philip Daian. Unity is strength: A formalization of cross-domain maximal extractable value, 2021. URL: https://arxiv.org/abs/2112.01472, doi:10.48550/ARXIV.2112.01472.
44. Ariel Orda and Ori Rottenstreich. Enforcing fairness in blockchain transaction ordering. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 368–375, San Francisco, CA, USA, May 2019. IEEE. doi:10.1109/BLOC.2019.8751349.
45. paco0x. Amm arbitrageur, 2021. URL: https://github.com/paco0x/amm-arbitrageur.
46. Julien Piet, Jaiden Fairoze, and Nicholas Weaver. Extracting godl [sic] from the salt mines: Ethereum miners extracting value, 2022.
47. Kaihua Qin, Liyi Zhou, Pablo Gamito, Philipp Jovanovic, and Arthur Gervais. An empirical study of defi liquidations: Incentives, risks, and instabilities. In *Proceedings of the 21st ACM Internet Measurement Conference*, IMC '21, page 336–350, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3487552.3487811.
48. Kaihua Qin, Liyi Zhou, and Arthur Gervais. 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*, pages 198–214, San Francisco, CA, USA, 2022. IEEE. doi:10.1109/SP46214.2022.9833734.
49. Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais. Attacking the defi ecosystem with flash loans for fun and profit, March 2021. arXiv:2003.03810, doi:10.1007/978-3-662-64322-8_1.
50. M. Rosenfeld. Analysis of Bitcoin Pooled Mining Reward Systems, December 2011. arXiv:1112.4980.
51. Fabian Schär. Decentralized finance: On blockchain-and smart contract-based financial markets. *Available at SSRN 3571335*, 103(2):153–174, April 2021. URL: https://ideas.repec.org/a/fip/fedlrv/91428.html, doi:10.20955/r.103.153-74.

52. Hayden Shively and Adam Egyed. Nantucket, 2021. URL: https://web.archive.org/web/20221012180650/https://github.com/haydenshively/Nantucket.
53. Andrei Shleifer and Robert W. Vishny. The limits of arbitrage. *The Journal of Finance*, 52(1):35–55, 1997. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1997.tb03807.x, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1997.tb03807.x, doi:10.1111/j.1540-6261.1997.tb03807.x.
54. SushiSwap. Sushiswap docs, 2022. URL: https://web.archive.org/web/20220901181030/https://docs.sushi.com/.
55. Adrian Sutton. Understanding attestation misses, September 2022. URL: https://web.archive.org/web/20220925110330/https://symphonious.net/2022/09/25/understanding-attestation-misses/.
56. Uniswap. Uniswap v2 overview, 2020. URL: https://web.archive.org/web/20220917011113/https://uniswap.org/blog/uniswap-v2.
57. Uniswap. Auto router v2, 2021. URL: https://web.archive.org/web/20211216202805/https://uniswap.org/blog/auto-router-v2.
58. Uniswap. Flash swaps, 2022. URL: https://web.archive.org/web/20220905142459/https://docs.uniswap.org/protocol/V2/guides/smart-contract-integration/using-flash-swaps.
59. Uniswap. Multihop swaps, 2022. URL: https://docs.uniswap.org/protocol/guides/swaps/multihop-swaps.
60. Uniswap. Uniswap v2 sdk, 2022. URL: https://github.com/Uniswap/v2-sdk.
61. Fabian Vogelsteller and Vitalik Buterin. Eip-20: Token standard, 2015. URL: https://eips.ethereum.org/EIPS/eip-20.
62. Anton Wahrstätter, Jens Ernstberger, Aviv Yaish, Liyi Zhou, Kaihua Qin, Taro Tsuchiya, Sebastian Steinhorst, Davor Svetinovic, Nicolas Christin, Mikolaj Barczentewicz, and Arthur Gervais. Blockchain censorship, 2023.
63. Dabao Wang, Siwei Wu, Ziling Lin, Lei Wu, Xingliang Yuan, Yajin Zhou, Haoyu Wang, and Kui Ren. 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*, SBC '21, page 23–28, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3457977.3460301.
64. Ye Wang, Yan Chen, Haotian Wu, Liyi Zhou, Shuiguang Deng, and Roger Wattenhofer. Cyclic arbitrage in decentralized exchanges, 2021. URL: https://arxiv.org/abs/2105.02784, doi:10.48550/ARXIV.2105.02784.
65. Ben Weintraub, Christof Ferreira Torres, Cristina Nita-Rotaru, and Radu State. A flash(bot) in the pan: Measuring maximal extractable value in private pools. In *Proceedings of the 22nd ACM Internet Measurement Conference*, IMC '22, page 458–471, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3517745.3561448.
66. Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.
67. wow. Aave protocol, 2020. URL: https://git.io/JLQVx, arXiv:https://git.io/JLQVx.
68. Matheus Venturyne Xavier Ferreira and David C. Parkes. Credible decentralized exchange design via verifiable sequencing rules. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, page 723–736, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3564246.3585233.

69. Jiahua Xu, Nazariy Vavryk, Krzysztof Paruch, and Simon Cousaert. Sok: Decentralized exchanges (DEX) with automated market maker (AMM) protocols, 2021. URL: https://arxiv.org/abs/2103.12732, arXiv:2103.12732.
70. Aviv Yaish, Kaihua Qin, Liyi Zhou, Aviv Zohar, and Arthur Gervais. Speculative denial-of-service attacks in ethereum. Cryptology ePrint Archive, Paper 2023/956, 2023. https://eprint.iacr.org/2023/956. URL: https://eprint.iacr.org/2023/956.
71. Aviv Yaish, Gilad Stern, and Aviv Zohar. Uncle maker: (time)stamping out the competition in ethereum. In *Proceedings of the 2023 ACM SIGSAC Conference on Computerand Communications Security (CCS '23)*, CCS '23, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3576915.3616674.
72. Aviv Yaish, Saar Tochner, and Aviv Zohar. Blockchain stretching & squeezing: Manipulating time for your best interest. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, EC '22, page 65–88, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3490486.3538250.
73. YCHARTS. Ethereum average block time, 2022. URL: https://web.archive.org/web/20221009031833/https://ycharts.com/indicators/ethereum_average_block_time.
74. yearn. yearn-vaults/contracts/basestrategy.sol, 2022. URL: https://github.com/yearn/yearn-vaults/blob/efb47d8a/contracts/BaseStrategy.sol#L539.
75. Liyi Zhou, Kaihua Qin, Antoine Cully, Benjamin Livshits, and Arthur Gervais. 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*, pages 919–936, San Francisco, CA, USA, 2021. IEEE. doi:10.1109/SP40001.2021.00113.
76. Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc Viet Le, and Arthur Gervais. 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*, pages 428–445, San Francisco, CA, USA, may 2021. IEEE. doi:10.1109/SP40001.2021.00027.

A Optimization Problems

A.1 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 flashswaps.

Denote all tokens which exist in our economy by $\mathcal{C}_1, \ldots, \mathcal{C}_n$. For brevity, we may refer to tokens by their index, e.g., use i instead of $\mathcal{C}_i$. 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. Let the set of all DEXs be S, and for each DEX $\mathbb{D} \in S$, denote the set of token-pairs which it allows to swap as $\text{TokenPairs}(\mathbb{D}) \stackrel{\text{def}}{=} \{(i,j) \mid i,j \text{ are valid token pair in } \mathbb{D}\}$. 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. (2).

$$\begin{aligned}
\text{maximize} \quad & c_{k_{end},i^*} \\
\text{subject to} \quad & \forall k \in [k_{end}] : \mathbb{D}_k \in S \\
& \forall k \in [k_{end}] : (i_k, j_k) \in \text{TokenPair}\left(\mathbb{D}_k\right) \\
& \forall k \in [k_{end}] : \Delta_k > 0 \\
& \forall k \in [k_{end}] : \delta_k = \text{Swap}_{\mathbb{D}_k}^{i_k \to j_k}(\Delta_k) \\
& \forall k \in [k_{end}] : c_{k,j_k} = c_{k-1,j_k} + \delta_k \\
& \forall k \in [k_{end}] : \phi_{k,i} = \text{Debt}(\text{Swap}_{\mathbb{D}_k}^{i_k \to j_k}(\Delta_k)) \\
& \forall i \in [n] : c_{k_{end},i} - \sum_{k=0}^{k_{end}} \phi_{k,i} \geq 0
\end{aligned} \tag{2}$$

Remark A.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 Ethereum 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 [59] which incorporates all logic within a single function.*

A.2 Optimal Flashswap-based Liquidation

Theorem 5.1 (Optimal liquidation). *Consider a debt position that is available for liquidation, where the debt is in cryptocurrency $\mathcal{D}$, and is collateralized by funds in cryptocurrency $\mathcal{C}$. If a user wishes to perform the liquidation using a swap obtained from a CPAMM with x_c of $\mathcal{C}$ and x_d of $\mathcal{D}$ reserved, then the optimal amount to repay is: $\varrho^* = \min\left(\kappa \cdot \delta, \frac{\sqrt{p_{liq} \cdot x_c \cdot x_d} - x_c}{p_{liq}}\right)$.*

Proof. Given the previously defined liquidation profit and close factor, the profitability of FSL is formalized in Eq. (3).

$$\begin{aligned}
\underset{\varrho}{\text{maximize}} \quad & \sigma \cdot p_{\mathcal{D}} \cdot \varrho \\
\text{subject to} \quad & \varrho \leq \kappa \cdot d
\end{aligned} \tag{3}$$

We assume that the liquidator wishes to perform the liquidation using a flashswap over a CPAMM DEX $\mathbb{D}$, and thus will obtain cryptocurrency $\mathcal{D}$ via

flashswap, use it to perform the liquidation, then receive cryptocurrency $\mathcal{C}$ in return, which is finally used to cover the flashswap. Technically, a liquidator can exchange an arbitrary amount of $\mathcal{C}$ to $\mathcal{D}$ as long as the flashswap can be covered.

At the time of liquidation, the spot price from $\mathcal{C}$ to $\mathcal{D}$ on $\mathbb{D}$ is $p_{\mathcal{C}\to\mathcal{D}} = \frac{x_d}{x_c}$, by the definition of the constant product rule. The exchange-rate from $\mathcal{C}$ to $\mathcal{D}$ on $\mathbb{D}$ may, however, diverge from the spot price due to the so-called *slippage*, which is defined as the difference between the current price of a trade, and the actual price at which it was carried out. Given a trade of size Δ, we denote the slippage as $Slippage(\Delta)$.

$$\text{Slippage}(\Delta) \stackrel{\text{def}}{=} p_{\mathcal{C}\to\mathcal{D}} \cdot \Delta - \text{Swap}^{\mathcal{C}\to\mathcal{D}}(\Delta) \tag{4}$$

The *slippage rate* is a multiplicative version, defined in the following manner.

$$\text{SlippageRate}(\Delta) \stackrel{\text{def}}{=} \frac{p_{\mathcal{C}\to\mathcal{D}} \cdot \Delta - \text{Swap}^{\mathcal{C}\to\mathcal{D}}(\Delta)}{p_{\mathcal{C}\to\mathcal{D}} \cdot \Delta} \tag{5}$$

Therefore, $\text{Swap}^{\mathcal{C}\to\mathcal{D}}(\Delta)$, representing the amount of $\mathcal{D}$ that is received when exchanging Δ units of $\mathcal{C}$ over $\mathbb{D}$, is formulated in Eq. (6).

$$\text{Swap}^{\mathcal{C}\to\mathcal{D}}(\Delta) = (1 - \text{SlippageRate}(\Delta)) \cdot p_{\mathcal{C}\to\mathcal{D}} \cdot \Delta \tag{6}$$

The slippage rate is defined generally. However, it can be written in a more concrete manner, under our assumptions that the DEX follows the constant product rule, and has x_c and x_d reserved.

$$\text{SlippageRate}(\Delta) = 1 - \frac{x_c}{\Delta} + \frac{\frac{x_c}{\Delta}}{1 + \frac{\Delta}{x_c}} \tag{7}$$

Observe that the slippage rate is a monotonically increasing function w.r.t. to the trade, implying that the exchange rate becomes less favorable for a trader when the trading volume increases.

Using the above results and previously made notations and definitions, the liquidation profit with a flashswap is hence outlined in Eq. (8).

$$\begin{aligned} \text{profit}^f(\varrho) &= p_{\mathcal{D}} \cdot \text{Swap}^{\mathcal{C}\to\mathcal{D}}(\alpha) - p_{\mathcal{D}} \cdot \varrho \\ &= p_{\mathcal{D}} \cdot (1 - \text{SlippageRate}(\alpha)) \cdot p_{\mathcal{C}\to\mathcal{D}} \cdot \alpha - p_{\mathcal{D}} \cdot \varrho \\ &= p_{\mathcal{D}} \cdot (p_{\mathcal{C}\to\mathcal{D}} \cdot p_{\text{liq}} \cdot (1 - \text{SlippageRate}\,(p_{\text{liq}} \cdot \varrho)) - 1) \cdot \varrho \end{aligned} \tag{8}$$

Note that $\text{SlippageRate}\,(p_{\text{liq}} \cdot \varrho)$ is monotonically increasing w.r.t. ϱ. So, $\text{profit}^f(\varrho)$ is concave w.r.t. ϱ, indicating that liquidating up to the close factor might not be the optimal strategy for FSL.

Given the above, we write the liquidation profit with flashswaps in Eq. (9).

$$\begin{aligned} \underset{\varrho}{\text{maximize}} \quad & x_d - \frac{x_c \cdot x_d}{x_c + p_{\text{liq}} \cdot \varrho} - \varrho \\ \text{subject to} \quad & \varrho \leq \kappa \cdot \delta \end{aligned} \tag{9}$$

By differentiating Eq. (9) with respect to ϱ, we get:

$$\frac{p_{\text{liq}} \cdot x_c \cdot x_d}{(\varrho \cdot p_{\text{liq}} + x_c)^2} - 1 \tag{10}$$

By solving this equation, we find that the optimal amount to repay is upper bounded by:

$$\frac{\sqrt{p_{\text{liq}} \cdot x_c \cdot x_d} - x_c}{p_{\text{liq}}} \tag{11}$$

Due to the close factor, this means that the optimal amount ϱ^* to repay is:

$$\varrho^* = \min\left(\kappa \cdot \delta, \frac{\sqrt{p_{\text{liq}} \cdot x_c \cdot x_d} - x_c}{p_{\text{liq}}}\right) \tag{12}$$

□

B Attack Suboptimality

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

Case Study B.1 (DeFi Attack Suboptimality: Inverse Finance) *A transaction with hash 0x958...13c began propagating on Ethereum's p2p network at Thursday, 16th 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 [49] 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 listening to Ethereum's p2p network overheard this transaction, which was not yet included in a block, and realized that the transaction's significant financial volume opens up an arbitrage opportunity that was overlooked by the attacker and which can be used to make a profit. Thus, the arbitrageur bundles the attack together with an arbitrage transaction with hash 0xfa1...6bd, 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 assisted the attacker, while also rewarding the miner with a reward exceeding the average Ethereum mining rewards from block rewards and transaction tips 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 [76]. Note that over* 50% *of Ethereum mining power is held by* 4 *mining pools, each holding more than* 8% *of all mining power [71].*

C 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.* [34] 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.* [47], 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.* [64], 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.* [76], 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.* [48] 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.* [46] and Weintraub *et al.* [65] analyze the profits produced by private transactions, but ignored collateralized lending.

Automatic Tools. Zhou *et al.* [75] 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.* [48] 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 [30, 42] study strategic Uniswap V3 LPs 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 Flashloan Attacks. Previous work has examined specific instances of flashloan attacks, and have shown that the profits made by them can be improved. For example, Qin *et al.* [49] did so in two specific cases, the famous bZx "Pump & Arbitrage" and "Oracle Manipulation" attacks. Cao *et al.* [11] have produced a tool that analyzes flashloan 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 accounting for the profit that can be made from transaction reordering and sandwich attacks.

Obadia *et al.* [43] 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.* [35], specifically in the context of CFMM DEXs. Yaish *et al.* [72] 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 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.* [44] and Ferreira *et al.* [68] 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.* [71] 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 replay [48] profitable transactions.

D Glossary

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

D.1 Symbols

α	The collateral acquired by a liquidator.
b	Total amount of borrowed funds, denoted in tokens.
κ	The debt that can be repaid within a single FSL, also known as the close factor.
c	The amount of collateral securing a position.
δ	The amount of debt for a position.
d	Total amount of deposited funds, denoted in tokens.
$\mathbb{D}$	A DEX.
η	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.
σ	The liquidation spread.
τ	Liquidation threshold, defined to be $\in (0, 1)$.
p	The United States Dollar (USD) price of a token.
ϱ	The amount of debt that a liquidator is repaying.
r	The reserve factor.
u	The utilization of the liquidity pool.

D.2 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
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