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]Yale University
[3]University College London
[4]UC 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: 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 used by over 850 platforms that are responsible for a daily volume of more than 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, which combines the previous two. The profit that 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 that extract profits using similar actions. By analyzing these events, we find that some transactions are circumstantially tied to specific miners and hypothesize that they use their knowledge of private orderflow for a profit. Essentially, this is an instance of miner-extractable value (MEV) "in action".
Keywords: Decentralized Finance, Miner Extractable Value, Lending, Exchanges.

1 Introduction

Decentralized finance (DeFi) is an umbrella term for financial platforms that run as smart contracts on cryptocurrencies, covering use cases such as token exchange to lending [54]. Although on-chain DeFi services appear similar to their off-chain equivalents, they may differ in subtle but important ways, leading to suboptimal usage of them. This is due to design considerations that are required for the secure and efficient implementation of financial services in the decentralized and pseudonymous setting in which DeFi platforms operate.

1.1 Our Work

We take the first step towards characterizing theoretically optimal usage of DeFi primitives that underlie common use-cases, and the suboptimal usage of them in practice:

1. Collateralized lending in utilization-based platforms

Table 1: A summary of the case studies we analyze in this work.

Case	"Lost" Profits	Action Type
Case Study 4.7: Justin Sun	More than 8.7%	Lending
Case Study 4.8: $0x7a1...428$	More than 700%	Lending
Case Study 5.7: $0x5e1...6c5$	More than 94%	Flashswap
Case Study 5.8: $0x0f4...74b$	More than 7000%	Flashswap
Case Study 5.9: Inverse Finance Attack	More than 39%	Flashswap
Case Study 6.8: liquidation tools	More than 53%	Liquidation

2. Using flashswaps to profit from arbitrage opportunities in decentralized exchanges (DEXs)

3. Liquidation of insolvent collateralized loans using flashswaps

In particular, while the first two are conceptually independent of each other, they are frequently combined in practice, giving rise to the last primitive. We now give an overview of our theoretical and empirical results for each one.

Collateralized Lending

In Section 4 we show that 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 (as well as Compound's 127 forks [22]) set their interest rate 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.

We formalize such mechanisms in Section 4.1, and make the novel observation that they sometime allow users who withdraw a partial amount of deposited funds to *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 (Optimization Problem 1). We then use it to analyze the performance of several large investors and show that their profits can be increased by up to 700% (Case Studies 4.6 to 4.8). Furthermore, we go over empirical evidence which suggests that currently, market forces are slow to react to shocks (Remark 4.5), such as when a large amount of deposits is withdrawn in a single transaction. Thus, even myopically optimizing actions can produce long-lasting profits.

Flashswap-based Arbitrage

In Section 5 we turn our focus to DEXs. If an asset has different prices in two or more DEXs, actors can take advantage of the discrepancy and make a profit by simultaneously buying and selling the asset in these markets, a practice known as *arbitrage*.

Some DEXs such as Uniswap (Uniswap v2 and v3, as well as the 715 platforms that are forks of either [21]) allow users to perform *flashswaps*. We formalize this primitive in Section 5.1; Intuitively, it allows users to 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 they make to fund their operations instead.

We define an optimization problem for the task of finding arbitrage opportunities using flashswaps (Appendix A), and show that blockchain users have performed suboptimally in the past via several case studies (Case Studies 5.3 and 5.7 to 5.9) and a longitudinal study (Section 5.2.1). In particular, we find instances where missed profits reach more than $\$517K$. This can be explained by the huge search space for arbitrage opportunities in Ethereum, stemming from 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.

In Section 5.2.2 we show that by closely inspecting the data we analyze in our longitudinal study, a surprising finding arises that could explain the business model of so-called *private transaction relays*. 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, and can be used, for example, by would-be arbitrageurs and others to prevent profitable transactions from being front-run. Popular relay services offer their services free of charge, piquing the community's interest with regard to their sources of revenue. In particular, we find circumstantial evidence suggesting that the relay service operated by the Ethermine mining pool has been using "inside information" obtained from private transactions to trim the search space to find arbitrage opportunities, thus gaining an advantage over others.

Flashswap-based Liquidation

In Section 6, we analyze a primitive which combines the previous two. We formalize in Section 6.1 the so-called *liquidation* mechanisms used by lending platforms to allow users to liquidate (i.e., repay) under-collateralized debt positions and receive the collateral at a discount. The amount of debt that can be repaid is usually limited by some *close factor*, that is, 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 (Case Study 6.8).

We show that this liquidation strategy is suboptimal, and provide a better one. In particular, we obtain a closed-form solution for the optimal execution of flashswap-based liquidations, and prove our result's optimality in Theorem 6.7. Our analysis 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 in constant product automated market makers (CPAMMs).

1.2 Our Contributions

Our contributions can be summarized like so:

- **Optimal Execution**. We examine three core DeFi primitives which are used by over 850 platforms: 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.

- **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 517K USD at the time. Our findings are summarized 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).

1.3 Organization

We review related work in Section 2, and go over the required background on cryptocurrencies in Section 3. We present our analysis of collateralized lending in Section 4, and of the use of flashswaps for arbitrage in Section 5. In Section 6, we combine the previous two primitives and analyze the optimal use of flashswaps for the liquidation of insolvent loans. We conclude with a discussion on the implications of our results in Section 7.

2 Related Work

As far as we know, no previous work has examined user suboptimality in the context of the DeFi primitives we cover.

Empirical Studies. Gudgeon et al. [37] formalize popular DeFi interest-rate mechanisms and use historical data to compare them. A systematization of common liquidation mechanisms is given by Qin et al. [50], who also empirically analyze liquidations performed on large Ethereum lending platforms. The historical performance of Uniswap v2 users performing cyclic arbitrage swaps is explored by Wang et al. [66], where the profits from each swap are used in their entirety to pay for the next swap in the cycle, a strategy which is not necessarily optimal. A larger spectrum of actions that users can use to obtain profits from DEXs is formalized and examined by Zhou et al. [78]. For example, the authors show that users can both back-run and front-run other transactions (place their own transactions before or after other transactions, respectively), possibly even combining the two into a "sandwich" attack. The authors estimate the profits which can be obtained in that manner in thousands of dollars per day. Qin, Zhou, and Gervais [51] quantify 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, Fairoze, and Weaver [48] and Weintraub et al. [67] analyze the profits produced by private transactions, but did not consider collateralized lending.

Automatic Tools. Zhou et al. [77] introduce two tools for creating profitable DeFi transactions, one which can detect profitable swap cycles, and another that also detects non-cyclic ones. Qin, Zhou, and Gervais [51] 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. [7] use convex optimization to identify arbitrage opportunities in a network of constant function market makers (CFMMs) DEXs and approximate the optimal solution. In our work, we show that such tools (and their open-source equivalents) may perform suboptimally.

Optimal Uniswap V3 Liquidity Provisioning. Fan et al. [34, 33] study strategic Uniswap V3 LPs who have some set of beliefs regarding the price evolution of assets, and formalize their optimal behavior in various settings. In contrast, our work differs by considering additional forms of DeFi platforms other than DEXs (e.g., interest-bearing liquidity pools such as Aave and Compound, for which we consider both liquidity provisioning and liquidations), and with respect to Uniswap-esque DEXs, we examine flashswaps, which were not considered by these works.

Optimizing Flashloan Attacks. Previous work examined specific instances of flashloan attacks and has shown that their profits can be improved. For example, Qin et al. [52] did so in two specific cases, the famous bZx "Pump & Arbitrage" and "Oracle Manipulation" attacks. Cao, Zou, and Cheng [11] present a tool that analyzes flashloan attacks, and use it to perform a similar analysis on the bZx "Pump & Arbitrage" attack. We extend these cases to a more general formalization of suboptimality. In particular, we also examine "honest" users, and additional types of primitives, such as flashswaps (which generalize flashloans), lending, and liquidations.

Miner Extractable Value. Angeris, Evans, and Chitra [6] 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. [45] present a formal study of the MEV that can be obtained by miners that operate on multiple blockchains. The game-theoretic aspects of MEV are analyzed by Kulkarni, Diamandis, and Chitra [38], specifically in the context of CFMM DEXs. Yaish, Tochner, and Zohar [74] 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 lending platforms. The authors show that the attack can be prevented by limiting system parameters, e.g., the interest offered by DeFi platforms.

Preventing MEV and Miner Malfeasance. Orda and Rottenstreich [46] and Xavier Ferreira and Parkes [70] attempt to prevent profitable transaction reordering, with the latter focusing on specifically on such manipulations in the context of DEXs. Yaish, Stern, and Zohar [73] 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" [51, 49] profitable transactions.

3 Preliminaries

We now go over the preliminaries required for our work. Notations are introduced as necessary, and all are summarized in Appendix B.

3.1 Background

Blockchains. Cryptocurrencies like Ethereum [68, 9] process financial *transactions* between users by collecting them in a ledger comprised of *blocks*, with each block specifying an order on transactions contained within it. Each block points to a preceding block, implying that the resulting *blockchain* maintains order between transactions. As blocks are size-limited, users compete for block-space and can offer *fees* to prioritize their transactions [36]. 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 [16]. The value earned from these exploits is termed *MEV*.

Lending Platforms. Certain DeFi platforms, like Aave and Compound, let users take and give loans [74]. 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 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 [50].

Decentralized Exchanges. DEXs allow the exchange, or *swapping*, of tokens [71]. Platforms like Uniswap [63] 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). Platforms commonly collect liquidity for each token pair in a different pool. If users wish to swap between pairs that do not have a designated pool, they can do so by chaining together multiple swaps in a single *multi-hop* swaps, with certain platforms offering tools to facilitate such actions [60, 62].

ERC20 Tokens. Ethereum specifies the ERC20 standard, that lets contracts create and interact with fungible tokens known as ERC20 tokens [26, 64]. DeFi platforms may also create ERC20 tokens for protocol-specific versions of other ERC20 tokens that they support. For example, Compound created *cUSDT*, a platform-integrated version of the USDT token [13]. 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 [44]), 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 [15].

3.2 Model

System Model. Our system consists of a blockchain that supports smart contracts. On top of this blockchain, there are at least two types of DeFi platforms implemented as smart contracts:

1. Collateralized lending platforms where users can lend or borrow funds
2. DEXs which let users trade one currency for another

User Model. Blockchain users can perform the following actions, which we define as needed:

1. Taking and repaying loans
2. Exchanging one currency for another
3. Liquidating insolvent loans

All actions can be executed using single transactions (e.g., if necessary, can be implemented in a smart contract that executes multiple actions in sequence).

4 Lending Suboptimality

Aave [69] and Compound [40] are collateralized lending platforms. At the time of writing, both hold a combined amount of 12.96 billion USD [23], making them the most popular lending platforms on Ethereum. Moreover, Compound's mechanism is used by 129 other lending protocols, contributing an additional 2.9 billion USD in stored funds [22]. To incentivize liquidity provisioning by users, platforms give interest on funds stored in their pools [37]. Although interest is usually associated with lending, any platform that rewards liquidity provisioning in a compounding manner is essentially paying interest.

4.1 Utilization-based Kinked Interest Rate Schemes

We now present a general model for the lending mechanisms that we consider. At their core is the *utilization* metric, which we denote by u and formalize in Definition 4.1.

Definition 4.1 (Utilization). *Given a liquidity pool where the total amount of deposited liquidity equals d and $b \leq d$ is borrowed, then the pool's* utilization *is:* $u \overset{def}{=} \frac{b}{d}$.

When the utilization is close to 1, a lending pool can be vulnerable in the event of economic shock or borrower insolvency. To protect against such events, lending pools commonly reserve some of the interest paid by borrowers, as formalized by Definition 4.2.

Definition 4.2 (Reserve Factor). *The* reserve factor *of a pool $r \in [0, 1)$ is the fraction of the interest paid by borrowers that is retained by the pool, and not paid to liquidity suppliers.*

For each token, the interest rates for deposits and borrows are determined according to the utilization of the token's liquidity pool. For the mechanisms we consider, the interest-rate curve for borrowers is "kinked" (see Definition 4.3): it has some "baseline" rate I_0, which increases at some rate I_1 before a predefined target utilization value u_{opt}, and at a (commonly higher) rate of I_2 afterward. The resulting function looks like it has a "kink" at that point, as can be seen in Fig. 1a, hence the name.

Definition 4.3 (Borrow Rate). *Let the liquidity pool's baseline interest rate be I_0 and the slopes before and after the kink u_{opt} be I_1 and I_2, respectively. Given a utilization of u, the* per-block interest for borrowers *is:*

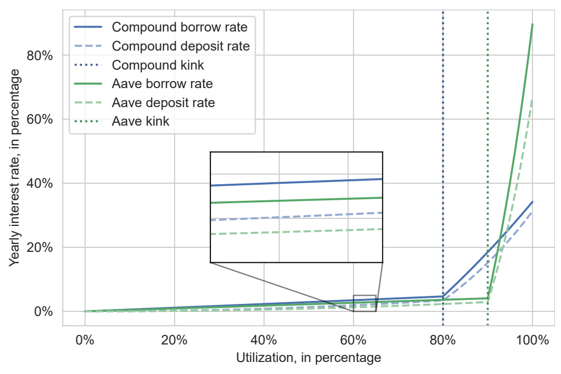
$$I_b(u) \overset{def}{=} I_0 + \min(u, u_{opt}) \cdot I_1 + (\max(u, u_{opt}) - u_{opt}) \cdot I_2.$$

For suppliers, the curve is a function of the borrow rate and the reserve factor (see Definition 4.4). By definition, it is quadratic in the utilization of the pool and, moreover, lower than the borrow rate, thereby preventing trivial arbitrage opportunities where actors can profit by borrowing funds and depositing them in the same pool.

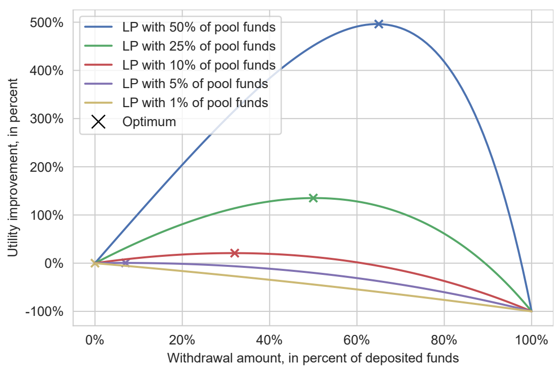
Definition 4.4 (Supply Rate). *Let the liquidity pool's reserve factor be r, and its interest on borrows be I_b. Given a utilization of u, the* per-block interest for liquidity suppliers *is:*

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

Real-world examples of interest curves are shown in Fig. 1a. Note that Definitions 4.3 and 4.4 are per-block, while the depicted examples extrapolate the per-block interest into the annualized interest, also known as the annual percentage yield (APY).



(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.

Figure 1: Optimal liquidity provisioning strategies may not always prescribe depositing all funds available to a user, as this can sometime *lower* profits.

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

Optimization Problem 1: Our optimization problem for optimal liquidity provisioning to utilization-based lending pools, see Section 4.1 for additional details.

4.2 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 rate 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 Optimization Problem 1.

Remark 4.5 (Long-term Considerations)**.** *Optimization Problem 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 [16], or by sending the transaction as part of a bundle [6].*

In case the actor's liquidity is large compared to others', Fig. 1b shows that the profits from even a single block can be substantial. As previous research shows, the majority of funds in most pools belong to a few independent actors [37], therefore, such considerations are relevant.

Furthermore, empirical data suggest 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 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 [41, 72].

4.3 Suboptimal Lending

Fig. 1b shows that the optimal action given by Optimization Problem 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 4.6 that Yearn, the largest investment platform on the Ethereum blockchain, is suboptimal.

Case Study 4.6 (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 [14, 74]. 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 [31, 76]. As we have shown, this "all-in" approach is, in fact, suboptimal.

Case Study 4.7 covers a specific instance of a user performing suboptimally.

Case Study 4.7 (Justin Sun)**.** *Compound's largest LP for the cUSDT token at block* 13632753 *was address 0x3dd...296, associated with Justin Sun [30], providing* 2.8 *billion cUSDT. At the time, the supply annual percentage yield (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 supply APY to* 5.15%*. Therefore, his per-block profit would grow by* 8.7% *to* 44.1 *cUSDT. This increase will remain until other users interact with the liquidity pool to the degree that utilization is changed, thus this profit can be assured for the next block if the transaction that performs the withdrawal is placed as the last transaction of the block.*

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 4.8, also depicted in Fig. 2, we cover a more dramatic case.

Case Study 4.8 (Account 0*x*7*a*1...428)**.** *Address 0x7a1...428 had a deposit of* 132.9 *million CRV in Aave at block* 15529008*, amounting to* 85% *of the pool's liquidity. The pool's utilization at the time was equal to* 45.66%*, and, moreover, it had more than* 100 *million tokens in its reserves. 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*

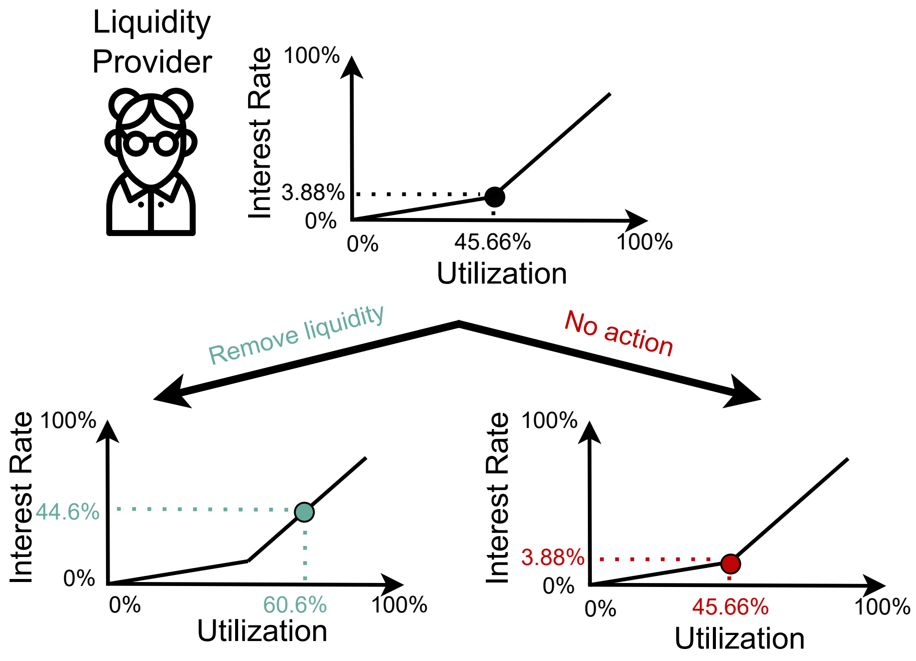


Figure 2: Counter-intuitively, the LP presented in Case Study 4.8 could increase its per-block profit by 700% by *withdrawing* 38 million CRV from the pool.

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%*.*

4.4 Discussion

Although LPs miss revenue by acting suboptimally, any optimal action they may take serves 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 profit off the suboptimality of the former.

5 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 [57] and Balancer [43], which together comprise roughly 75% of the daily volume of all token exchanges in Ethereum [20]. In particular, 627 additional platforms rely on the same mechanism employed by Uniswap v2, and similarly, 88 platforms are based on the one used by Uniswap v3 [21]. We note that Uniswap v1 is considered as a proof-of-concept by its creators [59], and thus was not included in our work. Surprisingly, even sophisticated users who are savvy enough to find extremely short-lived opportunities to make a profit by swapping assets across different DEXs, do not always do so optimally.

5.1 Using Flashswaps to Perform Arbitrage

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 [56] where users can generate a net profit by trading across platforms with different exchange-rates [77, 66], as formalized in Definition 5.1.

Definition 5.1 (Arbitrage)**.** *Profiting from the buying and selling of an asset across multiple markets which list different prices for it.*

The act of profiting from arbitrage is called *closing* the arbitrage, and users who engage in this practice are often called *arbitrageurs.* Arbitrage is prevalent in the current ecosystem, with arbitrageurs making an average monthly profit of more than 12 million USD [66].

Flashswaps. Flashswaps [61] 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, 32, 59], while Balancer's "batch swaps" are also implemented as flashswaps [8].

Flashswaps let users exchange one currency for another without having the initial liquidity requirements in advance, and work similarly to flashloans [16]. Generally, given that a user wants to obtain a c amount of token $\mathcal{C}$ as part of a flashswap, 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 [61]. 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 [72].

Using Flashswaps for Arbitrage. To illustrate how flashswaps can be used, in Example 5.2 (and the corresponding Fig. 3) we go over a real-world transaction that relies on this primitive to close an arbitrage opportunity.

Example 5.2. *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 repaid 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 at the time.*

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

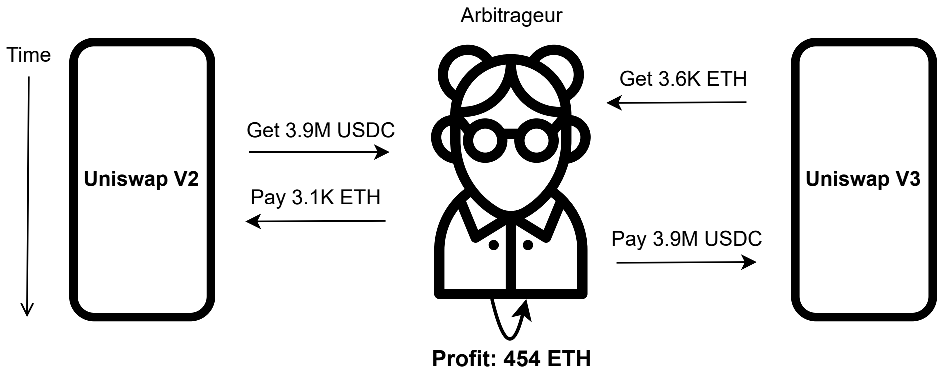


Figure 3: A depiction of Example 5.2, showing how flashswaps can be used to close arbitrage opportunities without having the required funds up-front. Instead, one can use part of the profits made to pay for the necessary operations.

5.2 Flashswap Arbitrage Suboptimality

We now turn to quantifying the suboptimality of arbitrageurs. Intuitively, even without going into the details of how to optimally devise flashswap arbitrage transactions (which we cover in Appendix A), there are many tokens and DEXs that 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 with every new block. In particular, Ethereum's block rate is currently set to 12 seconds [25, 75].

The time it takes 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 [58, 24]. 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 5.3 (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. [77] formulate and implement an algorithm that detects arbitrage opportunities in a variety of DEXs and tokens. The authors note that their tool obtains results in an average of 5.39 *seconds, although it does not perform an exhaustive search and thus is suboptimal. The tool's run-time was achieved by (1) applying heuristics to trim the search space, and (2) running on powerful hardware (AMD Ryzen Threadripper* 3990*X processor with* 64 *cores operating 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 [47, 2], which can under-perform relative to multi-hop ones.

Remark 5.4. *It is reasonable to assume that, unlike Case Study 5.3, arbitrageurs investing resources in developing automated arbitrage tools would avoid sharing their tools, thus preserving the potential advantages they have. This advantage is diminished when others have access to private transactions, as we discuss later in Section 5.2.2.*

5.2.1 Longitudinal Study

To further make the case that even sophisticated actors fall short of optimality, we devise a simple yet effective heuristic in Definition 5.5 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 5.5 (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 5.5, 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 5.6.

Definition 5.6 (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 discover over 9875 suboptimal transactions which fit Definition 5.5, with a total lost profit of more than 4.38 million USD. During the same period, an average of 2.91 ETH per day was lost 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 losses in profit over time are shown in Fig. 4.

Using our heuristic, we discover interesting cases that are worth additional attention. For example, Case Study 5.7 shows that even arbitrageurs who are proficient enough to make large profits are not always optimal.

Case Study 5.7 (Account $0x9ae...bc4$)**.** *Recall the transaction covered in Example 5.2. Although it produced a net profit of 454 ETH (equal at the time to 600K USD), it is suboptimal. 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 5.8.

Case Study 5.8 (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 $0xc07...a2b$, which involved the same order of operations and tokens, but on different DEXs; Uniswap v2 was used to swap ETH for USDT and USDT for

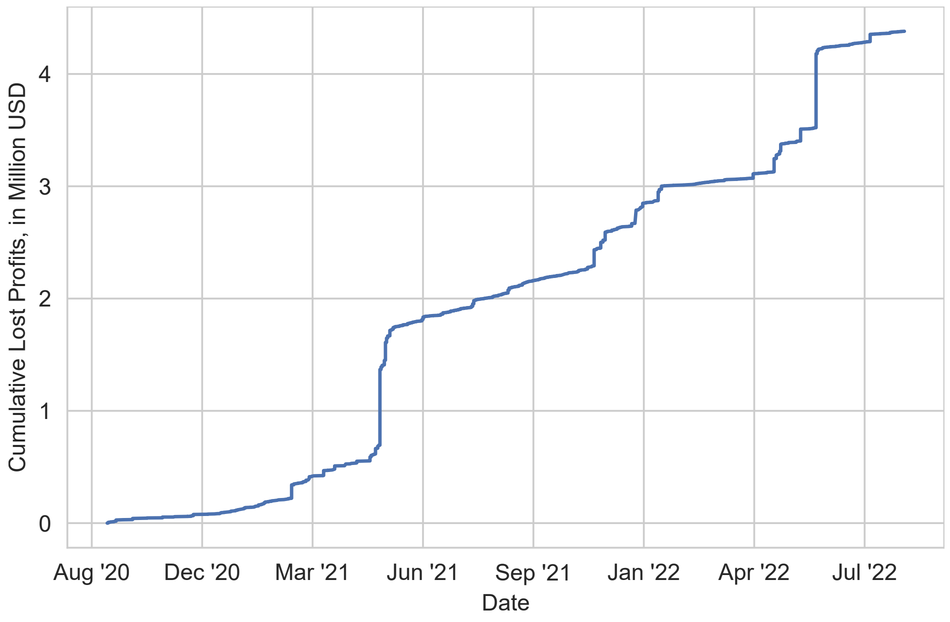


Figure 4: In our longitudinal study (see Section 5.2), we find that the cumulative lost profits by suboptimal flashswaps exceed 4 million USD over the examined time period.

Table 2: The top 5 arbitrageurs found in our longitudinal study.

Arbitrageur Address	**Number of Backrun Transactions**
$0x000...f56$	13K
$0x860...f66$	6K
$0xe33...c85$	6K
$0x911...116$	3.6K
$0x584...dba$	3.6K

USDC, and Sushiswap was used to swap USD for ETH. This latter transaction was even more successful, extracting an additional 170 *ETH in revenue (worth* 694*K USD at the time), making the initial transaction suboptimal by a factor of* ≈ 770.

5.2.2 Trimming the Search Space With Inside Information

A careful inspection of the results we gather reveals that some arbitrageurs are more actively back-running suboptimal transactions, with the top 5 arbitrageurs listed in Table 2.

Some miners allow users to relay transactions directly to them via so-called *private relays* instead of using the public peer to peer (p2p) network, while guaranteeing the privacy of transactions sent in that manner [35, 27]. This may be done to prevent adversarial actors from taking advantage of transactions, e.g., by front-running them. Such channels are widely adopted by arbitrageurs and miners alike [51].

In this regard, arbitrageur $0x584...dba$ should be particularly noted: circumstantial evidence suggests it is possibly related to the Ethermine mining pool [53], which was the largest before Ethereum's transition to PoS [73]. 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 [29, 28].

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. This information can be used, for example, to trim the large search space by heuristically copying the actions employed by private transactions. Indeed, in the cases found by our heuristics, neighboring transactions use similar operations, usually only differing in the DEXs used (e.g., Case Studies 5.7 and 5.8).

5.3 Discussion

Suboptimal arbitrage transactions, by their nature, do not always close all arbitrage opportunities and may even create new ones. As arbitrageurs are actively seeking out such opportunities [66, 51], this means that unexploited arbitrage leaves the door open to potentially other transactions which will attempt to capitalize on it. Thus, this suboptimality can result in increased congestion [42], leading to higher transaction fees for all users.

A different aspect which is interesting to consider is that, in certain cases, the suboptimality of a transaction may incentivize its prompt inclusion in a block, if, for example, it creates arbitrage opportunities. This can be considered as the DeFi equivalent of the so-called "undercutting" attack by Carlsten et al. [12]. We discuss an example of such an occurrence in Case Study 5.9.

Case Study 5.9 (DeFi Attack Suboptimality: Inverse Finance)**.** *A transaction with hash* $0x958...13c$*, sent by* $0x7b7...ec6$*, 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 flashloan 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* 83.95 *ETH to the winning miner, Ethermine, making a profit of* 168.88 *ETH in the process. At the time, the net profit and the fee paid were worth a total of* $471K$ *USD, meaning the attacker could have increased its profit by more than* 39%*. Thus, even savvy DeFi attackers do not always execute optimal attacks.*

Knowingly or not, the arbitrageur as well as the FaaS assisted the attacker, while also rewarding the miner with a reward exceeding the average per-block rewards than can be earned 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 [78]. Note that at the time, over* 50% *of Ethereum mining power was held by* 4 *mining pools, each holding more than* 8% *of all mining power [73].*

6 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*.

6.1 Fixed Spread Liquidation

Currently, fixed spread liquidation (FSL) is the prevalent method for securing LP capital [50]. 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.

Liquidation Mechanism. 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.

To measure the "health" of a debt position, that is, how close it is to insolvency, platforms use a metric called the *health factor*, which we formalize in Definition 6.1.

Definition 6.1 (Health Factor)**.** *Consider a debt position of a δ amount of $\mathcal{D}$ tokens, collateralized by a c amount of $\mathcal{C}$ tokens, where the price of 1 $\mathcal{D}$ token is $p_{\mathcal{D}}$ USD, and of 1 $\mathcal{C}$ token is $p_{\mathcal{C}}$ USD. Given a liquidation threshold $\tau \in (0,1)$, the* health factor *of the debt position: $\eta \overset{def}{=} \frac{p_{\mathcal{C}} \cdot c}{p_{\mathcal{D}} \cdot \delta} \cdot \tau$.*

When a debt position's collateral value decreases because of, for example, price decline (i.e., $\frac{p_{\mathcal{C}}}{p_{\mathcal{D}}} \downarrow$), then its health factor decreases, indicating that the borrower is more likely to default.

Once a borrowing position's health factor falls below one, it becomes available for liquidation. Liquidators can repay the debt of unhealthy positions and acquire a corresponding part of their collateral at a discount called the *liquidation spread*, which we denote by a ratio σ. We define the obtained discounted collateral in Definition 6.2.

Definition 6.2 (Acquired Collateral)**.** *Given a liquidation spread of σ and a debt position of δ tokens with a health factor $\eta < 1$, a liquidator that repays ϱ tokens acquires the following part of the collateral: $\alpha \overset{def}{=} \frac{p_{\mathcal{D}} \cdot (1+\sigma)}{p_{\mathcal{C}}} \cdot \varrho$.*

Remark 6.3. *For brevity, we define the following: $p_{liq} \overset{def}{=} \frac{p_{\mathcal{D}} \cdot (1+\sigma)}{p_{\mathcal{C}}}$, thereby we can simplify the acquired collateral to $\alpha = p_{liq} \cdot \varrho$.*

Pools can limit the amount of debt that can be repaid in a single FSL, per Definition 6.4.

Definition 6.4 (Close Factor)**.** *The* close factor *κ constrains the maximal amount of debt that can be repaid in a single FSL. Given a debt position of δ tokens and a close factor κ, one may only repay up to ϱ tokens, where: $\kappa \geq \frac{\varrho}{\delta}$.*

Using the aforementioned definitions, we can reason about liquidator profits.

Definition 6.5 (Liquidation Profit)**.** *Given a liquidation spread of σ, a close factor of κ, and a debt position of δ tokens with a health factor $\eta < 1$, a liquidator that repays ϱ tokens (where the amount does not exceed the close factor, i.e., $\varrho \leq \kappa \cdot \delta$) obtains $\alpha = \frac{p_{\mathcal{D}} \cdot (1+\sigma)}{p_{\mathcal{C}}} \cdot \varrho$ tokens and receives a profit equal to $\text{profit}^t(\varrho) \overset{def}{=} p_{\mathcal{C}} \cdot \alpha - p_{\mathcal{D}} \cdot \varrho = \sigma \cdot p_{\mathcal{D}} \cdot \varrho$.*

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

Optimization Problem 2: Our program for optimal liquidations, see Section 6.2 for details.

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 [52] have become a popular option for liquidators [39, 65]. 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 DEXs that use the constant product rule, a prevalent DEX pricing formula [78]. Such DEXs are referred to as 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 per the CPAMM's exchange rate, as defined in Definition 6.6. As the name suggests, the constant product rule requires that trades preserve the product of the amounts of each asset held.

Definition 6.6 (Constant Product Market Maker Swap)**.** *Given a CPAMM with x_c of $\mathcal{C}$ and x_d of $\mathcal{D}$ reserved, the amount of $\mathcal{D}$ tokens received for providing a Δ amount of $\mathcal{C}$ is:*

$$\text{Swap}^{\mathcal{C}\to\mathcal{D}}(\Delta) \overset{def}{=} x_d - \frac{x_c \cdot x_d}{x_c + \Delta}.$$

6.2 Optimal Flashswap-based Liquidation

A program for optimal-profit FSLs is formalized in Optimization Problem 2. Presumably, to optimize liquidation profits, one 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 6.7.

Theorem 6.7 (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. We assume that the liquidator wishes to perform the liquidation using a flashswap over a CPAMM DEX $\mathbb{D}$, and thus will obtain token $\mathcal{D}$, use it to perform the liquidation, then receive token $\mathcal{C}$ 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 swap 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

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

Optimization Problem 3: Our program for optimal flashswap-based liquidations, see Section 6.2 for details.

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*(Δ).

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

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

$$\text{SlippageRate}(\Delta) \overset{\text{def}}{=} \frac{p_{\mathcal{C}\to\mathcal{D}} \cdot \Delta - \text{Swap}^{\mathcal{C}\to\mathcal{D}}(\Delta)}{p_{\mathcal{C}\to\mathcal{D}} \cdot \Delta} \tag{2}$$

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. (3).

$$\text{Swap}^{\mathcal{C}\to\mathcal{D}}(\Delta) = (1 - \text{SlippageRate}(\Delta)) \cdot p_{\mathcal{C}\to\mathcal{D}} \cdot \Delta \tag{3}$$

The slippage rate is defined generally. However, it can be written more concretely when considering DEXs that use the constant product rule, and have x_c and x_d reserved.

$$\text{SlippageRate}(\Delta) = 1 - \frac{x_c}{\Delta} + \frac{\frac{x_c}{\Delta}}{1 + \frac{\Delta}{x_c}} \tag{4}$$

Observe that the slippage rate is monotonically increasing 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. (5).

$$\begin{aligned} \text{profit}^f(\varrho) &= p_{\mathcal{D}} \cdot \text{Swap}^{\mathcal{C}\to\mathcal{D}}(\alpha) - p_{\mathcal{D}} \cdot \varrho \\ &= p_{\mathcal{D}} \cdot (1 - \text{SlippageRate}(\alpha)) \cdot p_{\mathcal{C}\to\mathcal{D}} \cdot \alpha - p_{\mathcal{D}} \cdot \varrho \\ &= p_{\mathcal{D}} \cdot (p_{\mathcal{C}\to\mathcal{D}} \cdot p_{\text{liq}} \cdot (1 - \text{SlippageRate}(p_{\text{liq}} \cdot \varrho)) - 1) \cdot \varrho \end{aligned} \tag{5}$$

Note that $\text{SlippageRate}(p_{\text{liq}} \cdot \varrho)$ increases monotonically 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 this insight, we write a program for devising optimal flashswap-based liquidations in Optimization Problem 3. By differentiating Optimization Problem 3 with respect to ϱ, we get:

$$\frac{p_{\text{liq}} \cdot x_c \cdot x_d}{(\varrho \cdot p_{\text{liq}} + x_c)^2} - 1 \tag{6}$$

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

$$\frac{\sqrt{p_{\text{liq}} \cdot x_c \cdot x_d} - x_c}{p_{\text{liq}}} \tag{7}$$

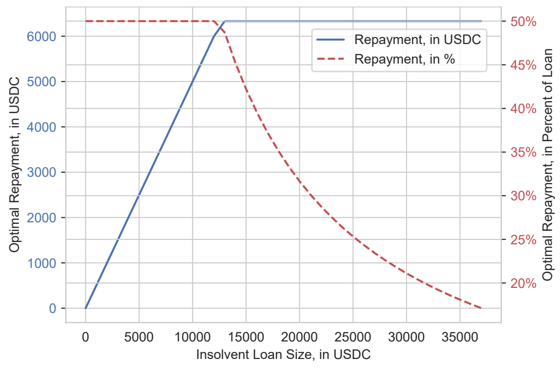


Figure 5: Optimal repayment in a swap-based liquidation for Aave's USDC pool, that has a close factor of 50%. Due to exchange-rate slippage, it is suboptimal to repay large illiquid positions up to the close factor.

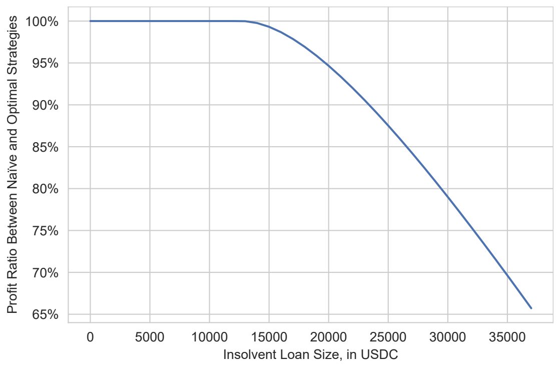


Figure 6: The ratio between profits made by naïve and optimal flashswap based liquidations for Aave's USDC pool. The ratio drops below 1 for large loans, indicating that the naïve strategy is suboptimal.

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

$$\varrho^* = \min\left(\kappa \cdot \delta, \frac{\sqrt{p_{\text{liq}} \cdot x_c \cdot x_d} - x_c}{p_{\text{liq}}}\right) \tag{8}$$

□

As we show in Case Study 6.8, the optimal strategy we present in Theorem 6.7 can improve profits compared to a naïve strategy that liquidates up to the close factor, even when considering relatively small debt positions.

Case Study 6.8 (Flashswap 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 6.7, instead naïvely performing liquidations up to the close factor [55, 1, 17].*

To illustrate the gap between the two strategies mentioned above, in Fig. 6 we plot the ratio between the profit obtained by the naïve one and our optimal one. Moreover, we plot the corresponding absolute and relative amounts that should be liquidated in Fig. 5. Both figures assume that the debt to be liquidated is in Aave's USDC liquidity pool, and the DEX used is Uniswap v2, with all parameters set to their real-world values as of block 15731128*.*

As Fig. 6 shows, naïve liquidations may produce 65% *less profit than optimal ones in certain cases, or, conversely, the latter can outperform the former by more than* 53%*.*

6.3 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.

7 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.

Acknowledgements

This research was supported by the Ministry of Science & Technology, Israel.

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

[7] 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. Boulder, CO, USA: Association for Computing Machinery, 2022, pp. 115–128. ISBN: 9781450391504. DOI: 10.1145/3490486.3538336. URL: https://doi.org/10.1145/3490486.3538336.

[8] Balancer. *Flash Swaps*. 2023. URL: https://github.com/balancer/docs/blob/309ee3/docs/reference/swaps/flash-swaps.md.

[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". In: *Information* 13.1 (2021), p. 6. DOI: 10.3390/info13010006.

[11] Yixin Cao, Chuanwei Zou, and Xianfeng Cheng. *Flashot: A Snapshot of Flash Loan Attack on DeFi Ecosystem*. 2021. DOI: 10.48550/ARXIV.2102.00626. URL: https://arxiv.org/abs/2102.00626.

[12] Miles Carlsten, Harry Kalodner, S. Matthew Weinberg, and Arvind Narayanan. "On the Instability of Bitcoin Without the Block Reward". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 154–167. ISBN: 9781450341394. DOI: 10.1145/2976749.2978408. URL: https://doi.org/10.1145/2976749.2978408.

[13] Compound. *cTokens*. 2022. URL: https://compound.finance/docs/ctokens.

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

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

[16] 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*. San Francisco, CA, USA: IEEE, 2020, pp. 910–927. DOI: 10.1109/SP40000.2020.00040.

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

[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. URL: https://defillama.com.

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

[21] DefiLlama. *Forks*. May 2024. URL: https://defillama.com/forks.

[22] DefiLlama. *Forks - Compound V2*. May 2024. URL: https://defillama.com/forks/Compound%20V2.

[23] DefiLlama. *Lending TVL Rankings*. 2024. URL: https://web.archive.org/web/20240517081433/https://defillama.com/protocols/Lending.

[24] Ben Edgington. *Upgrading Ethereum | One Page Annotated Spec*. Dec. 2022. URL: https://web.archive.org/web/20221218133524/https://eth2book.info/bellatrix/annotated-spec/%5C#seconds_per_slot.

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

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

[27] 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.

[28] 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.

[29] 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.

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

[31] Etherscan. *Contract 0x83f798e925BcD4017Eb265844FDDAbb448f1707D*. 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. 2022. URL: https://etherscan.io/address/0x83f798e925bcd4017eb265844fddabb448f1707d%5C#code%5C#L682.

[32] Etherscan. *Contract 0xCBCdF9626bC03E24f779434178A73a0B4bad62eD*. 2022. URL: https://etherscan.io/address/0xcbcdf9626bc03e24f779434178a73a0b4bad62ed%5C#code%5C#F1%5C#L777.

[33] Zhou Fan, Francisco Marmolejo-Cossio, Daniel Moroz, Michael Neuder, Rithvik Rao, and David C. Parkes. "Strategic Liquidity Provision in Uniswap V3". In: *5th Conference on Advances in Financial Technologies (AFT 2023)*. Ed. by Joseph Bonneau and S. Matthew Weinberg. Vol. 282. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 25:1–25:22. ISBN: 978-3-95977-303-4. DOI: 10.4230/LIPIcs.AFT.2023.25. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.AFT.2023.25.

[34] 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. DOI: 10.48550/ARXIV.2204.00464. URL: https://arxiv.org/abs/2204.00464.

[35] Flashbots. *Flashbots*. 2022. URL: https://github.com/flashbots/pm.

[36] Yotam Gafni and Aviv Yaish. *Greedy Transaction Fee Mechanisms for (Non-)myopic Miners*. 2022. DOI: 10.48550/arXiv.2210.07793. URL: https://arxiv.org/abs/2210.07793.

[37] 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*. ACM, 2020, pp. 92–112. DOI: 10.1145/3419614.3423254.

[38] Kshitij Kulkarni, Theo Diamandis, and Tarun Chitra. *Towards a Theory of Maximal Extractable Value I: Constant Function Market Makers*. 2022. DOI: 10.48550/ARXIV.2207.11835. URL: https://arxiv.org/abs/2207.11835.

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

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

[41] 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. Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, pp. 1645–1667. ISBN: 9781450384544. DOI: 10.1145/3460120.3485369. URL: https://doi.org/10.1145/3460120.3485369.

[42] 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. Los Angeles, CA, USA: Association for Computing Machinery, 2022, pp. 2099–2113. ISBN: 9781450394505. DOI: 10.1145/3548606.3559341. URL: https://doi.org/10.1145/3548606.3559341.

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

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

[45] 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. DOI: 10.48550/ARXIV.2112.01472. URL: https://arxiv.org/abs/2112.01472.

[46] Ariel Orda and Ori Rottenstreich. “Enforcing Fairness in Blockchain Transaction Ordering”. In: *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. San Francisco, CA, USA: IEEE, May 2019, pp. 368–375. DOI: 10.1109/BLOC.2019.8751349.

[47] paco0x. *AMM Arbitrageur*. 2021. URL: https://github.com/paco0x/amm-arbitrageur.

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

[49] Kaihua Qin, Stefanos Chaliasos, Liyi Zhou, Benjamin Livshits, Dawn Song, and Arthur Gervais. “The blockchain imitation game”. In: *Proceedings of the 32nd USENIX Conference on Security Symposium*. SEC ’23. Anaheim, CA, USA: USENIX Association, 2023. ISBN: 978-1-939133-37-3.

[50] 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. Virtual Event: Association for Computing Machinery, 2021, pp. 336–350. ISBN: 9781450391290. DOI: 10.1145/3487552.3487811. URL: https://doi.org/10.1145/3487552.3487811.

[51] 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*. San Francisco, CA, USA: IEEE, 2022, pp. 198–214. DOI: 10.1109/SP46214.2022.9833734. URL: https://doi.org/10.1109/SP46214.2022.9833734.

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

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

[54] Fabian Schär. “Decentralized Finance: On Blockchain-and Smart Contract-based Financial Markets”. In: *Available at SSRN 3571335* 103.2 (Apr. 2021), pp. 153–174. DOI: 10.20955/r.103.153-74. URL: https://ideas.repec.org/a/fip/fedlrv/91428.html.

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

[56] Andrei Shleifer and Robert W. Vishny. “The Limits of Arbitrage”. In: *The Journal of Finance* 52.1 (1997), pp. 35–55. DOI: https://doi.org/10.1111/j.1540-6261.1997.tb03807.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1997.tb03807.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1997.tb03807.x.

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

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

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

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

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

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

[63] Uniswap. *Uniswap v2 SDK*. 2022. URL: https://github.com/Uniswap/v2-sdk.

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

[65] 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. Virtual Event, Hong Kong: Association for Computing Machinery, 2021, pp. 23–28. ISBN: 9781450384056. DOI: 10.1145/3457977.3460301. URL: https://doi.org/10.1145/3457977.3460301.

[66] Ye Wang, Yan Chen, Haotian Wu, Liyi Zhou, Shuiguang Deng, and Roger Wattenhofer. *Cyclic Arbitrage in Decentralized Exchanges*. 2021. DOI: 10.48550/ARXIV.2105.02784. URL: https://arxiv.org/abs/2105.02784.

[67] 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. Nice, France: Association for Computing Machinery, 2022, pp. 458–471. ISBN: 9781450392594. DOI: 10.1145/3517745.3561448. URL: https://doi.org/10.1145/3517745.3561448.

[68] Gavin Wood et al. "Ethereum: A secure decentralised generalised transaction ledger". In: *Ethereum project yellow paper* 151.2014 (2014), pp. 1–32.

[69] wow. *AAVE Protocol*. 2020. eprint: https://git.io/JLQVx. URL: https://git.io/JLQVx.

[70] 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. Orlando, FL, USA: Association for Computing Machinery, 2023, pp. 723–736. ISBN: 9781450399135. DOI: 10.1145/3564246.3585233. URL: https://doi.org/10.1145/3564246.3585233.

[71] Jiahua Xu, Krzysztof Paruch, Simon Cousaert, and Yebo Feng. "SoK: Decentralized Exchanges (DEX) with Automated Market Maker (AMM) Protocols". In: *ACM Comput. Surv.* 55.11 (Feb. 2023). ISSN: 0360-0300. DOI: 10.1145/3570639. URL: https://doi.org/10.1145/3570639.

[72] Aviv Yaish, Kaihua Qin, Liyi Zhou, Aviv Zohar, and Arthur Gervais. "Speculative Denial-of-Service Attacks in Ethereum". In: *33rd USENIX Security Symposium (USENIX Security 24)*. USENIXSEC '24. Philadelphia, PA: USENIX Association, Aug. 2024. eprint: https://ia.cr/2023/956. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/yaish.

[73] 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 Computer and Communications Security (CCS '23)*. CCS '23. Copenhagen, Denmark: Association for Computing Machinery, 2023. ISBN: 9798400700507. DOI: 10.1145/3576915.3616674.

[74] 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. Boulder, CO, USA: Association for Computing Machinery, 2022, pp. 65–88. ISBN: 9781450391504. DOI: 10.1145/3490486.3538250. URL: https://doi.org/10.1145/3490486.3538250.

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

[76] yearn. *yearn-vaults/contracts/BaseStrategy.sol*. 2022. URL: https://github.com/yearn/yearn-vaults/blob/efb47d8a/contracts/BaseStrategy.sol%5C#L539.

[77] 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*. San Francisco, CA, USA: IEEE, 2021, pp. 919–936. DOI: 10.1109/SP40001.2021.00113. URL: https://doi.org/10.1109/SP40001.2021.00113.

[78] 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*. San Francisco, CA, USA: IEEE, May 2021, pp. 428–445. DOI: 10.1109/SP40001.2021.00027.

A Optimal Flashswap Arbitrage

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

Tokens and DEXs. Denote all tokens that 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 $\mathcal{C}_i$. 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:

$$\text{TokenPairs}(\mathbb{D}) \overset{\text{def}}{=} \{(i, j) \mid i, j \text{ are valid token pair in } \mathbb{D}\}.$$

Swaps. 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 $\phi_i = \text{Debt}\left(\text{Swap}_{\mathbb{D}}^{i \to j}(\Delta)\right)$.

Remark A.1. *Note that a flashswap which is to be repaid with the received token can be denoted as a swap between token pair (i, i).*

Arbitrageur. 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 (when $k_{end} > 1$, this is also known as a multi-hop swap; see Remark A.2 for additional details). Note that decreasing k_{end} allows reducing the time required for finding a solution, thus allowing the consideration of time-constrained arbitrageurs. In case $k_{end} > 1$, we denote by $c_{k,i}$ the amount of token i that is held or owed by the user at the end of the k-th step of the transaction.

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

Optimization Problem 4: A program for creating optimal flashswap arbitrage transactions, see Appendix A for details.

Remark A.2 (Multi-hop Swaps)**.** *"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 Ethereum's specification, a single transaction can only call one function. This function can, in turn, call multiple other ones. To create a single transaction that performs multiple function calls, one must either use an existing smart contract which supports such logic, or create a new one [62] which incorporates all logic within a single function.*

Optimal Execution. Finally, the arbitrageur should solve the problem in Optimization Problem 4.

B Glossary

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

B.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.

B.2 Acronyms

AMM	automated market maker
APY	annual percentage yield
CFMM	constant function market maker
CPAMM	constant product automated market maker
CRV	Curve token
cUSDT	Compound's wrapped version of USDT
DeFi	decentralized finance
DEX	decentralized exchange
ERC	Ethereum request for comments
ETH	Ethereum
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
USDC	USD Coin, a USD stablecoin
USDT	Tether's USD stablecoin
w.r.t.	with regards to
WBTC	Wrapped Bitcoin
WETH	Wrapped Ethereum