Stochastic fluctuations of sandwich attack in decentralized exchanges

Huisu Jang[1*], Bumho Son[2], and Yunyoung Lee[3]

[1]School of Finance, Soongsil University, Sangdo-ro 369, Seoul, 06978, Korea
[2]Department of Business Administration, Chung-Ang University, Heukseok-ro 84, Seoul, 06974, Korea
[3]Department of Data Science, Sejong University, Neungdong-ro 209, Seoul, 05006, Korea

January 25, 2024

Abstract

This study aims to investigate the complexities of Maximal Extractable Value (MEV) in Decentralized Finance (DeFi), with a focus on sandwich attacks on decentralized exchanges (DEXs). We present a novel model that revises traditional assumptions about attacker behavior, highlighting a strategic balance between transaction fees and the unpredictability of profits, by introducing the concept of 'external slippage'. Our approach contrasts with previous beliefs that attackers always maximize slippage against victims. Our empirical analysis demonstrate that the theoretical insight is aligned with the actual market behavior based on sandwich attack data with Uniswap liquidity pool swap transactions. This research not only deepens the understanding of sandwich attack but also paves the way for future explorations into the market dynamics of MEV. Additionally, our results underscore the necessity of refining assumptions regarding attack volumes for better alignment between theoretical models and empirical data.

1 Introduction

MEV (Maximal Extractable Value) is a feature that occurs in the DeFi (decentralized finance) industry and refers to the additional benefit that a validator or other participant can extract during the process of creating a transaction in a block. MEV encompasses various forms, including profit generation via general arbitrage, liquidation, and sandwich attacks. While not all instances of MEV negatively impact the market — with rapid arbitrage and liquidation often enhancing DeFi market efficiency — sandwich attacks represent a more problematic aspect. In such attacks, an entity manipulates market prices to yield extra profits, adversely affecting the victim through this manipulation. Consequently, sandwich attacks are categorized as a form of MEV with potentially harmful implications for the DeFi market.

In this research, we introduce a novel model that determines the optimal attack volume for attackers in sandwich attacks on decentralized exchanges (DEXs). Our model takes into account the variability in the attacker's expected profits, offering a realistic reflection of DeFi market conditions. Contrary to [HW22]'s assertion that attackers exploit the maximum slippage available to victims, empirical data suggests a more nuanced approach where attackers do not fully utilize the victim's maximum slippage.

A critical aspect of sandwich attacks is the uncertainty faced by both attackers and victims regarding the transaction order within a block. To influence this order, participants pay a priority fee in addition to the base gas fee, increasing the likelihood of their transaction being executed earlier. The preliminary analysis reveals a direct correlation between increased priority fees and earlier transaction placement. This finding underscores the trade-off between the cost of achieving earlier order of transaction in a block and the consequent reduction in the unpredictability of sandwich attack profit.

We leverage this empirical data to introduce the concept of 'external slippage,' which plays a pivotal role in shaping attackers' strategic decisions. We presents a novel model that guides attackers

in optimizing their attack strategies within these market dynamics. External slippage represents a balancing act for attackers, weighing the costs against the variability in expected returns. A larger priority fee increases the likelihood of a transaction being executed earlier in a block, consequently encountering fewer price shifts due to subsequent transactions. This reduction in price fluctuation leads to a more predictable range of potential profits. However, this advantage comes at the expense of higher fees. Conversely, a smaller priority fee follows the opposite principle, with reduced costs but increased variance in returns. Our model quantifies the probability of a successful sandwich attack and suggests that, contrary to previous studies, attackers may opt for smaller-scale attacks, a hypothesis we substantiate through empirical evidence.

Our approach is not solely theoretical; we validate our model against actual market data. Through an analysis of swap transactions in Uniswap versions 2 (V2) and 3 (V3) liquidity pools in November 2022, our study validates the main idea of our model. It demonstrates that sandwich attackers execute transactions at an optimal volume, aiming for partial rather than full slippage against victims, by balancing the trade-off between incurred costs and the minimization of profit volatility.

The structure of the paper is as follows: Section 2 reviews related research. Section 3 details our proposed theoretical model and the decision-making process for attackers. Section 4 presents empirical evidence supporting our model. Finally, Section 5 summarizes our contributions and outlines directions for future research.

2 Background

2.1 Automated Market Maker (AMM)

Many DEXs implement automated market makers (AMMs) as their core mechanism, which are algorithms facilitating instant trades without the need for traditional order books. Most AMM-based DEXs use Constant Function Market Makers (CFMMs), with a CFMM employing a fixed pricing mechanisms, whereby the asset's price is determined solely by the ratio of the two assets in the liquidity pool. This design choice eliminates the need for complex algorithms ore external price oracles, streamlining the trading process and enhancing the transparency of price discovery. The constant function nature of CFMMs, often associated with the well-known automated market maker Uniswap, has garnered attention for its simplicity and resistance to certain types of market manipulation. However, challenges such as impermanent loss and the potential for suboptimal pricing under extreme market conditions necessitate further research and optimization in order to fully realize the benefits and address potential drawbacks associated with CFMMs in decentralized financial ecosystems.

Uniswap V2 In Uniswap V2 [AZR20], we denote the amount of tokens X and Y reserved in the liquidity pool as x and y respectively, and the overall liquidity of the pool as L. Assuming that there is no additional liquidity provision, the amount of tokens in the pool should always follow the CFMM formula as follows:

$$x \cdot y = L^2 \tag{1}$$

The marginal price of the asset Y with respect to X at time t can be computed as :

$$p_t = -\frac{\partial x_t}{\partial y_t} = \frac{L_t^2}{y_t} = \frac{x_t}{y_t} \tag{2}$$

A token swap within the liquidity pool triggers a state change in the token reserves. However, the pool must consistently adhere to the invariant formula expressed in Equation 1. Consequently, when there is an alteration in quantities denoted by Δx and $-\Delta y$, the resulting amounts $(x + \Delta x)$ and $(y - \Delta y)$ must still satisfy:

$$(x + \Delta x) \cdot (y - \Delta y) = L^2 \tag{3}$$

Uniswap V3 & Concentrated Liquidity Uniswap V3 [AZS+21], represented a significant advancement beyond Uniswap V2 by introducing the concept of concentrated liquidity, deviating from the equal distribution of liquidity across the entire price spectrum seen in Uniswap V2. Formally,

liquidity providers in Uniswap V3 possess the capability to concentrate their liquidity within specific price ranges, referred to as "ticks," rather thanuniformly providing liquidity across the entire price curve. This design allows liquidity providers to strategically target specific price ranges where they anticipate more favorable trading opportunities or reduced impermanent loss. The introduction of concentrated liquidity in Uniswap V3 aims to enhance capital efficiency, providing liquidity providers with more nuanced control over their assets within the trading range. This innovation reflects a sophisticated approach to liquidity provision, catering to a broader spectrum of user preferences and risk profiles within the dynamic landscape of DeFi.

As each Uniswap V3 liquidity provider has unique liquidity positions characterized by distinct tick ranges, consider a liquidity position with liquidity L, the lower price boundary p_l, and the upper price boundary p_u. In this context, the following equation should hold:

$$(x + \frac{L}{\sqrt{p_u}})(y + L\sqrt{p_l}) = L^2 \tag{4}$$

The description of Uniswap V3 provided here is inherently localized, focusing on trade dynamics within specific price intervals. Integrating these local dynamics across all price points results in the formation of an aggregate reserve curve, governing trades across the entire spectrum of possible prices.

The swap mechanism employed in Uniswap V3 adheres CPMM model, which is in line with the approach employed in Uniswap V2. Assuming that the current price is P_c and a trader endeavors to input Δy of token Y and receive Δx of token X in return. We know the fact that when swapping within a price range, only P_c changes and the liquidity L remains unchanged. Then, we can find the post-swap price by using:

$$\Delta\sqrt{P} = \frac{\Delta y}{L} \tag{5}$$

As we know the input amount Δy, the post-swap price P_a is:

$$\sqrt{P_a} = \sqrt{P_c} + \frac{\Delta y}{L} \tag{6}$$

After calculating the post-swap price, we can calculate the token amounts by using the amount calculations functions:

$$\begin{aligned} x &= \frac{L(\sqrt{P_c} - \sqrt{P_a})}{\sqrt{P_c}\sqrt{P_a}} \\ y &= L(\sqrt{P_c} - \sqrt{P_a}) \end{aligned} \tag{7}$$

2.2 Sandwich Attack

The sandwich attack is typically done on the DEX liquidity pool by actual attackers or the predatory trading bots. It aims to attack the traders who want to swap two tokens in the liquidity pool by adding the front- and back- run transactions to the trader's transaction. The basic principles of success of the sandwich attack is to temporarily manipulate the swap price in the liquidity pool. The front-run transaction increase the swap price of the token that the trader want to receive, and the back-run transaction realize the profit of attackers.

For the rest of our model analysis, we will consider the sandwich attack on single liquidity pool consisted of tokens X and Y with swap transaction fee f. The initial amounts of tokens in the liquidity pool before the transactions are sent (i.e., the state of the liquidity pool in the most recently created block) are x_0 and y_0, respectively. The trader sends the transaction T_v, which exchanges δ_{v_x} amount of tokens X to δ_{v_y} amount of tokens Y. T_v also contains information about the maximum slippage tolerance s, which determines how much loss the trader is willing to accept and proceed with the swap. Since δ_{v_y} is derived from the most recently observed state of the liquidity pool, the actual state of the pool when the T_v is executed might be different, which leads to the difference of the amount of tokens Y that the trader will receive. We will denote the actual amount of tokens Y that the trader will receive as $\tilde{\delta}_{v_y}$. The transaction will be only executed when the following condition is satisfied:

$$\tilde{\delta}_{v_y} \geq (1 - s)\delta_{v_y} \tag{8}$$

When the trader, the victim of the sandwich attack, submits transaction T_v to the mempool, the attacker observes it and creates front-run transaction T_a^{front} and back-run transaction T_a^{back}. T_a^{front} swaps $\delta_{a_x}^{in}$ tokens X for δ_{a_y} tokens Y and T_a^{back} swaps δ_{a_y} tokens Y for $\delta_{a_x}^{out}$ tokens X. The attacker binds three transactions $(T_a^{front}, T_v, T_a^{back})$ and submits it to the mempool with base fee b and priority fee r.

3 Model

3.1 The relationship between transaction fee and transaction order

To support our theoretical model that the amount of fee that attackers are willing to pay to the block proposers (or validators) can significantly influence the position of their transactions within a block, we empirically investigate the relationship between transaction costs and transaction indices. Accordingly, we compute the correlation between transaction index and the cost-related variables(gas price, transaction fee, and the overall cost=transaction fee + coinbase transfer). Our attack model is based on the assumption that block proposers determine the transaction order within the block based on the potential revenue they can receive from each transaction. As a result, in addition to the absolute value of each cost-related variables, we also compute the relative rankings of transaction index, gas price and overall cost within the block in descending order. For instance, if a transaction has a lower gas price ranking, it indicates a higher gas price compared to other transactions. Using these data, we begin by calculating the Pearson's correlation matrix among transaction ranking, gas price, overall cost, cost ranking, and gas price ranking. As shown in Table 1, the correlation matrix reveals a high correlation between the transaction order and the relative ranking of gas price (0.7912). This suggests that block proposers accord higher priority to gas price compared to other variables when determining the sequence of transactions within the block. This behavior of block proposers seems quite reasonable, given that the actual transaction fee cannot be precisely predicted in advance, as block proposers face challenges in estimating the exact amount of gas used for each transaction. Instead, they utilize the relative gas price ranking of the transactions in the mempool as a criterion for establishing the order of transactions. Additionally, Figure 1 plots the gas price ranking and transaction order ranking of 5,000 transactions, visually confirming the high correlation between them. It clearly demonstrates that transactions with a higher gas price compared to others are positioned earlier in the block. We also provide the Spearman and Kendall's correlation between transaction order and the cost-related variables in Table 2. The results closely align with those in Table 1.

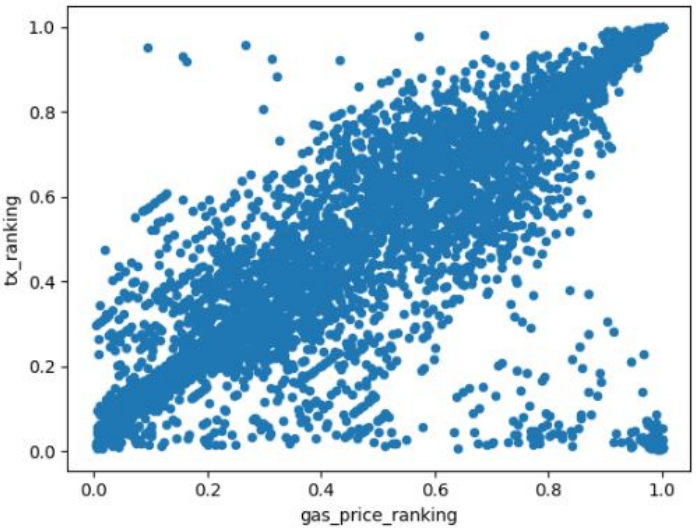

Figure 1: The transaction order and the gas price ranking of 5,000 transactions.

Table 1: Pearson's Correlation Matrix between transaction order and cost-related variables

	tx ranking	**gas price**	**gas price ranking**	**overall cost**	**cost ranking**
tx ranking	1.0	-0.1160	**0.7912**	-0.0147	0.1603
gas price	-0.1160	1.0	-0.1280	-0.1262	-0.0803
gas price ranking	**0.7912**	-0.1280	1.0	0.0064	0.0576
overall cost	-0.0147	0.1262	0.0064	1.0	-0.0533
cost ranking	0.1603	-0.0803	0.0576	-0.0533	1.0

Table 2: Spearman and Kendall's Correlation Matrix between transaction order and cost-related variables

Panel A: Sperman Correlation					
	tx ranking	**gas price**	**gas price ranking**	**overall cost**	**cost ranking**
tx ranking	1.0	-0.4351	**0.6842**	-0.1358	0.1464
Panel B: Kendall Correlation					
	tx ranking	**gas price**	**gas price ranking**	**overall cost**	**cost ranking**
tx ranking	1.0	-0.3978	**0.6814**	-0.09678	0.1187

3.2 External Slippage

At the point that trader submits a swap transaction, T_v, to the liquidity pool, he expects to receive δ_{v_y} amounts of token Y as follows:

$$\delta_{v_y} = \frac{y_0(1-f)\delta_{v_x}}{x_0 + (1-f)\delta_{v_x}} \tag{9}$$

However, the amounts of tokens in the liquidity pool will change for every swap transactions. Therefore, the actual states of the liquidity pool when each transaction is executed will be different. We will define the states of the liquidity pool just before when transactions T_v^{front}, T_v, and T_V^{back} are executed as (x'_0, y'_0), (x'_1, y'_1), and (x'_2, y'_2), respectively.

It is important to point out that (x'_0, y'_0) will be different from (x_0, y_0), which is the crucial difference between our work and [HW22]. It occurs because of the *external slippage*. We will define *internal slippage* and *external slippage*, which when added together will equal to the total slippage that the transaction of trader will actually face. The sandwich attack of attackers itself causes the *internal slippage* to the transaction of trader who would be a victim of sandwich attack. On the other hand, *external slippage* means the change in the state of liquidity pool that is not expected by the attackers. Even though the attacker who is willing to do the sandwich attack tries to put its transactions at the front of the block, we cannot affirm that it will locate as the first transaction of the newly created block. Therefore, there exists the possibility of other transactions participating in the liquidity pool may precede the T_v^{front}. These other transactions affects the state of the liquidity pool and it becomes the *external slippage*. As a result, the sum of the internal and external slippage should be small enough to satisfy 9, which leads to the execution of T_v.

During the overall procedures of sandwich attack, the external slippage will be only occurred just before the T_a^{front} is executed because the transactions $(T_a^{front}, T_v.T_a^{back})$ will be bundled together and no other transactions will be able to get in between them. We will assume that the external slippage will change (x_0, y_0) to (x'_0, y'_0), where $x'_0 \sim N(x_0, \frac{\sigma^2}{r})$ and $x_0 y_0 = x'_0 y'_0$. σ^2 denotes the magnitude of external slippage. This assumption is plausible in two senses. First, a change in the state of the liquidity pool can cause the amount of X to increase or decrease for same chance. x'_0 following the normal distribution well captures the bi-directional change of the pool state. Second, as the attacker pays more priority fee r, it becomes more likely to place its transactions at the front part of the block. It reduces the likelihood of sandwich attack transactions being preceded by other transactions participating in the same liquidity pool, which leads to the decrease of the magnitude of external slippage. Even though the normal distribution assumption has the risk that the number of tokens can have negative value, it has significantly small probability because $\frac{\sigma^2}{r}$ will be much smaller than x_0 in most cases.

3.3 Procedures of Sandwich Attack

Given the state of the liquidity pool (x'_0, y'_0) just before the sandwich attack transactions are executed, we can derive the quantities of the number of tokens that each transactions will receive. When T_a^{front} is executed, it will receive δ_{a_y} tokens Y as follows:

$$\delta_{a_y} = \frac{y'_0(1-f)\delta_{a_x}^{in}}{x'_0 + (1-f)\delta_{a_x}^{in}} \tag{10}$$

Therefore, (x'_1, y'_1) becomes $x'_1 = x'_0 + \delta_{a_x}^{in}$ and $y'_1 = \frac{x'_0 y'_0}{x'_0 + (1-f)\delta_{a_x}^{in}}$. Consequently, when T_v is executed, it will receive $\tilde{\delta}_{v_y}$ tokens Y as follows:

$$\tilde{\delta}_{v_y} = \frac{\frac{x'_0 y'_0}{x'_0 + (1-f)\delta_{a_x}^{in}}(1-f)\delta_{v_x}}{x'_0 + \delta_{a_x}^{in} + (1-f)\delta_{v_x}} \tag{11}$$

(x'_2, y'_2) becomes $x'_2 = x'_1 + \delta_{v_x}$ and $y'_2 = \frac{x'_1 y'_1}{x'_1 + (1-f)\delta_{v_x}}$. Finally, when T_a^{back} is executed, it will receive $\delta_{a_x}^{out}$ tokens Y as follows:

$$\delta_{a_x}^{out} = \frac{x'_2(1-f)\delta_{a_y}}{y'_2 + (1-f)\delta_{a_y}} \tag{12}$$

When the every procedures are over, the profit of adversary sandwich attackers becomes as follows:

$$P_a = \delta_{a_x}^{out} - \delta_{a_x}^{in} - 2b - 2r \tag{13}$$

However, T_v will not be executed if the total slippage is larger than s. Therefore, we can write the profit more precisely as follows:

$$P_a = \begin{cases} \delta_{a_x}^{out} - \delta_{a_x}^{in} - 2b - 2r, & \text{if } \tilde{\delta}_{v_y} \geq (1-s)\delta_{v_y} \\ -2b - 2r, & \text{otherwise} \end{cases} \tag{14}$$

Since $\delta_{a_x}^{out}$ is the random variable affected by x'_0, we will now consider the expected profit $\mathbb{E}[P_a | \delta_{a_x^{in}}, r]$.

The adversaries aim to maximize its expected profit by controlling $\delta_{a_x}^{in}$ and r. These two parameters that adversaries can control affects the expected profit in the terms of attack success probability and cost. As they increase $\delta_{a_x}^{in}$, $\mathbb{E}[\delta_{a_x}^{out}]$ will increase since the gave more input to the liquidity pool. On the other hand, the victim will face more internal slippage and it will make harder to meet the slippage tolerance condition of T_v. Another parameter r directly affects the expected profit since itself is a cost, and also affects the attack success probability by changing the magnitude of external slippage.

Considering the effects of $\delta_{a_x}^{in}$ and r, we can derive the attack success probability $f(\delta_{a_x}^{in}, r)$ (i.e. probability of slippage tolerance condition is met) as follows:

$$f(\delta_{a_x}^{in}, r) = \mathbb{P}\left(\frac{\frac{x'_0 y'_0}{x'_0 + (1-f)\delta_{a_x}^{in}}(1-f)\delta_{v_x}}{x'_0 + \delta_{a_x}^{in} + (1-f)\delta_{v_x}} \geq (1-s)\frac{y_0(1-f)\delta_{v_x}}{x_0 + (1-f)\delta_{v_x}} \right) \tag{15}$$

We show that $f(\delta_{a_x}^{in}, r)$ has a closed form solution in Lemma 1.

Lemma 1. *If a quadratic equation* $h(x) = -(1-s)x^2 - (1-s)(2\delta_{a_x}^{in} + (1-f)\delta_{v_x})x + x_0^2 + (1-f)\delta_{v_x}x_0 - \delta_{a_x}^{in}(1-s)(\delta_{a_x}^{in} + (1-f)\delta_{v_x}) = 0$ *has two real roots* $h_1 < h_2$*, then*

$f(\delta_{a_x}^{in}, r) = \Phi(\frac{h_2 - x_0}{\sigma^2/r}) - \Phi(\frac{h_1 - x_0}{\sigma^2/r})$*, where* $\Phi(\cdot)$ *is the cumulative distribution function of standard normal distribution.*

Proof.

$$\begin{aligned} f(\delta_{a_x}^{in}, r) &= \mathbb{P}(h(x_0) \geq 0) \\ &= \mathbb{P}(-(1-s)(x'_0 - h_1)(x'_0 - h_2) \geq 0) \\ &= \mathbb{P}(h_1 \leq x'_0 \leq h'_2) \ (\because 0 \leq s \leq 1) \\ &= \Phi\left(\frac{h_2 - x_0}{\sigma^2/r}\right) - \Phi\left(\frac{h_1 - x_0}{\sigma^2/r}\right) \end{aligned} \tag{16}$$

□

Using Lemma 1, we can derive the expected profit of the adversary attacker as follows:

$$\mathbb{E}[P_a] = \left(\Phi\left(\frac{h_2 - x_0}{\sigma^2/r}\right) - \Phi\left(\frac{h_1 - x_0}{\sigma^2/r}\right)\right)(\delta_{a_x}^{out} - \delta_{a_x}^{in}) - 2b - 2r \tag{17}$$

In the perspective of attackers, they will act to maximize its expected profit. Specifically, they will solve the following two variables optimization problem.

$$\begin{aligned} \max_{\delta_{a_x}^{in}, r} \quad & \mathbb{E}[P_a] \\ \text{s.t.} \quad & \delta_{a_x}^{in}, r \geq 0 \end{aligned} \tag{18}$$

We can compare the optimal solution $\delta_{a_x}^s$ derived from [HW22] with the optimal solution $(\delta_{a_x}^*, r^*)$ of Equation 18. $\delta_{a_x}^*$ will be smaller than $\delta_{a_x}^s$ because $\delta_{a_x}^s$ is based on the belief that optimal adversaries will attack as much property that just satisfies the slippage tolerance. Since we have pointed out the existence of external slippage, adversary should maintain enough safety margin or high level of transaction fee to guarantee high possibility of the attack success.

4 Empirical study

4.1 Data Collection

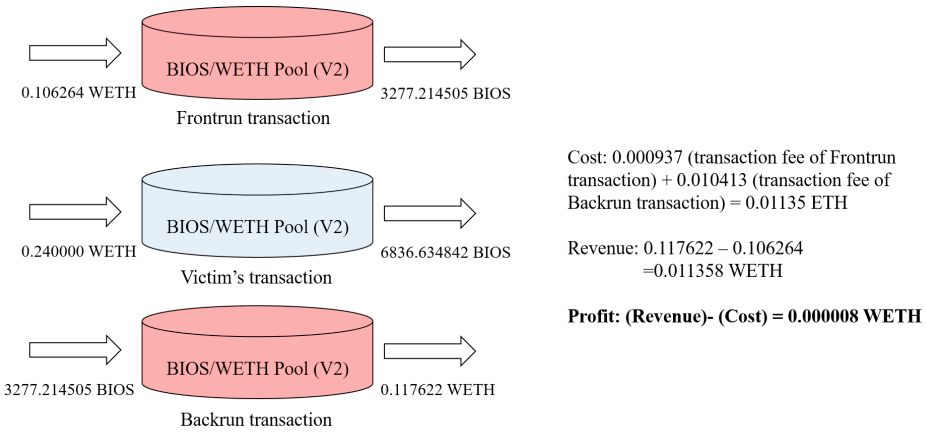

Figure 2: Example of a sandwich attack detected at the Uniswap V2 BIOS/WETH Pool in block 16009112.

We analyze the sandwich attacks detected on Uniswap V2 and V3 to comprehend the practical nature of how these attacks are executed. Initially, we identify sandwich attacks carried out between block 16,000,000 (Nov 18, 2022) and block 16,010,000 (Nov 20, 2022) employing the detection algorithm proposed in [PJL+24]. To streamline our analysis, we exclusively focus on attack cases where WETH (Wrapped Ether) is among the underlying assets in the pool. Furthermore, we filter the attacks to include only cases where the attacker's profit is denominated in WETH, and the sum of the other assets amounts to zero. As a result, our empirical analysis comprises of 1463 sandwich attacks, consisting of 4810 swap transactions. To acquire comprehensive details for each transaction within our dataset, we leverage the Erigon node of the Ethereum blockchain. Our access to the Erigon archive node facilitated the retrieval of substantial information pertaining to the transactions. This encompassed critical data such as the transaction fee (computed as the product of gas price and used gas), coinbase transfer (direct transfer of Ethereum to the proposer of a block), transaction index, and the quantity of underlying assets exchanged within Uniswap pools.

Moreover, our dataset includes pools from both Uniswap V2 and V3. Consequently, we compute the marginal price for each pool differently, depending on the respective version of the pool. For Uniswap V2 pools, we denote the amount of each token deposited at time t as x_t, y_t for two tokens X and Y, and L_t denoting the liquidity of the pool. With the invariant formula $x_t y_t = L_t^2$, the marginal

Table 3: Summary of the sandwich attack dataset

Panel A: Uniswap V2 & V3 Comparison		
	Uniswap V2	**Uniswap V3**
# of frontrun transactions	1263	423
# of victim transactions	1089	349
# of backrun transactions	1263	423
Panel B: Descriptive Statistics (in WETH)		
	Cost	**Revenue**	**Profit**
# of observations	1463	1463	1463
Mean	0.026640	0.028505	0.001865
Median	0.008073	0.008609	0.000101
Standard Deviation	0.085542	0.089568	0.009376
Min	0.001611	0.001612	-0.000626
Max	1.403588	1.464834	0.184897

price of token Y respect to token X can be calculated as $p_t = -\frac{\partial x_t}{\partial y_t} = \frac{L_t^2}{y_t} = \frac{x_t}{y_t}$. To apply this formula, we collect the token reserves' amount for Uniswap V2 pools at each block. However, for Uniswap V3 pools, the marginal price cannot be calculated simply as the ratio of token reserves between X and Y since the provided liquidity varies based on the given price interval. Therefore, we additionally obtain SqrtPriceX96 at time t, where SqrtPriceX96 $= 2^{96} \times \sqrt{p_t}$. Thus, the marginal price can be computed as $p_t = (\text{SqrtPriceX96})^2/2^{192}$ for Uniswap V3 pools. Table 3 represents the descriptive statistics of sandwich attack dataset used for our empirical analysis.

To enhance readers' understanding of the sandwich attack, we provide an illustration of an actual sandwich attack detected at block 16009112 in Figure 2. The attack was executed at BIOS/WETH pool of Uniswap V2, where the attacker initially inflated the price of BIOS by buying 3277.214505 BIOS with 0.106264 WETH. Subsequently, the victim had to purchase 6836.634842 BIOS at an elevated price, further driving up the price OF BIOS in the pool. Finally, the attacker sold the 3277.214505 BIOS obtained in the frontrun transaction at a higher price, resulting in an arbitrage revenue of 0.001358 WETH. After deducting the transaction fees of 0.00135 ETH for both the frontrun and backrun transactions, the total profit from the attack amounted to 0.000008 WETH.

4.2 Simulation results

We have done numerical analysis of our proposed model framework. For the simulation, we set parameters $s = 0.001, x_0 = 1000, y_0 = 1000, f = 0.003, \delta_{v_x} = 100, \sigma = 0.01, b = 0$. The Figure 3 shows the surface of expected profit by varying $\delta_{a_x}^{in}$ and r. The red dot is the optimal $(\delta_{a_x}^*, r^*) = (0.405, 0.005)$, while the dashed red line is the line with $\delta_{a_x}^{in} = 0.405$. In this parameter setting $\delta_{a_x}^s$ is 0.524. We can check that there exists optimal solution in the point that smaller than when we do not account for the external slippage.

However, $\delta_{a_x}^*$ is far smaller than the attack amount from the real world transaction data. It is the major limitation of our model structure. We assumed that x_0' will follow normal distribution, which has the advantage of fully reflecting changes in both directions. However, the problem with this assumption is that it inherently assumes a low probability of attack success. For example, applying the optimal attack volume derived from [HW22] to the current model results in only a 50% chance of attack success.

4.3 Empirical Analysis of Sandwich Attacks

To investigate the practical evidence of our sandwich attack model proposed in Section 3, we assess whether the attackers attempt to maximize their profit from the attack by selecting the highest possible input amount ($\delta_{a_x}^{in}$), thereby reaching the maximum slippage tolerance of the victim's transaction. However, in order to correctly compare the slippage tolerance chosen by the victim with the actual slippage observed through the transaction, we require the historical memory pool data of the Ethereum blockchain, as the slippage tolerance itself is not recorded in the main blockchain. Nevertheless, due to constraints in accessing historical memory pool data, which are not stored on the blockchain, our

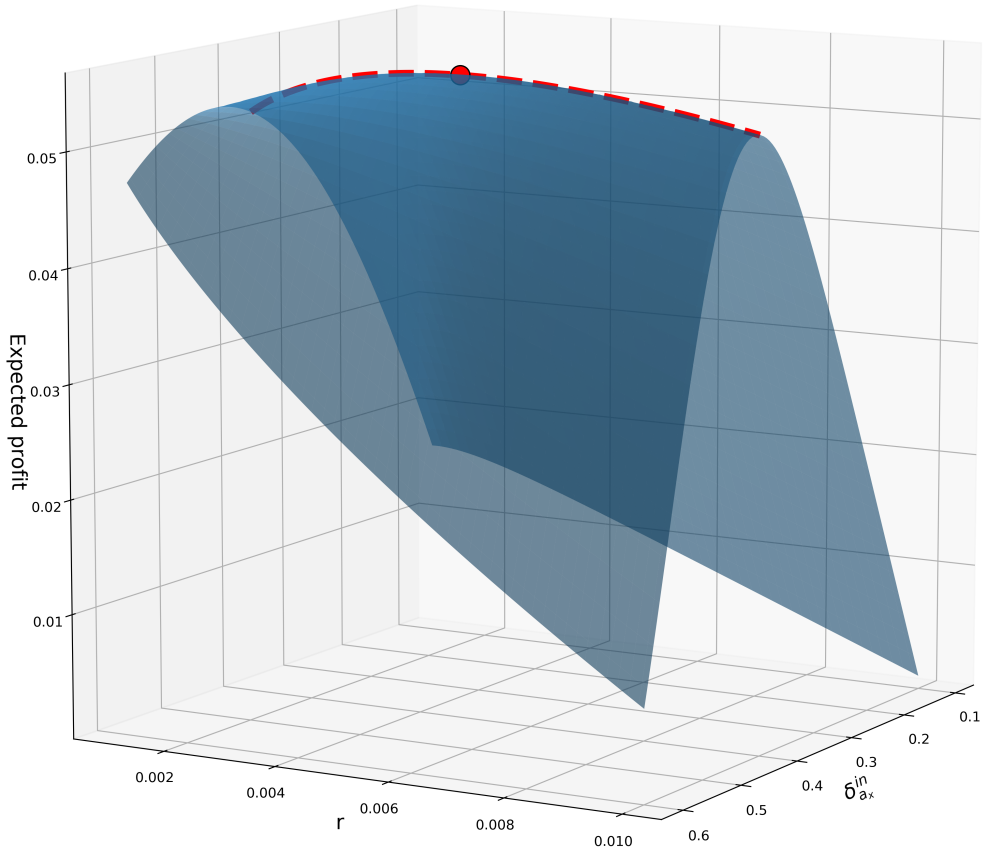

Figure 3: Simulation result of the optimal solution of Equation 18

analysis uses Uniswap's default maximum slippage thresholds of 0.5% and 5.5% as proxies for these values, rather than the actual historical pool data. In specific instances, users may opt for the highest permissible slippage, with Uniswap's maximum slippage threshold set at 20%. Consequently, in our empirical analysis, we estimated slippage by examining the transaction volume between the victim and the attacker, employing the most relevant slippage benchmarks of 0.5%, 5.5%, and 20%.

We introduced a variable termed "ratio", representing the proportion of the actual attack input volume ($\delta^{in}_{a_x}(actual)$) executed by the sandwich attacker relative to the maximum feasible attack input volume ($\delta^{in}_{a_x}(feasible)$), constrained by the victim's maximum slippage allowance. Figure 4 presents a box plot for the 'ratio' variable in our sandwich attack dataset, clearly indicating that the ratio value for 75% of the total data is below 1. This suggests that attackers do not use the full maximum possible attack volume suggested by [HW22]'s study. However, approximately 25% of cases exhibit a ratio exceeding 1. In this case, it can be thought of as an outlier that occurs because the actual slippage value submitted by the victim is unknown.

Our analysis confirms that the optimal attack volume proposed by our theoretical model is consistent with actual data. However, attacks aiming to achieve maximum slippage in a victim's transaction, as described in existing research ([HW22]), constitute less than 10% of all documented successful sand-

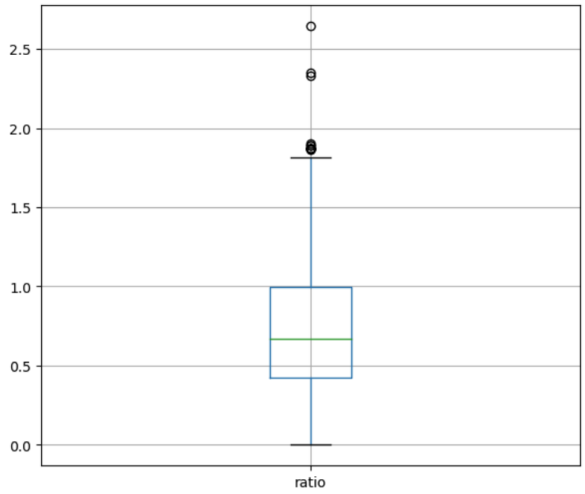

Figure 4: A box plot of the ratio of the actual attack input volume ($\delta^{in}_{a_x}(actual)$) executed by the sandwich attacker relative to the maximum feasible attack input volume ($\delta^{in}_{a_x}(feasible)$)

wich attacks. While there are some inconsistencies between empirical data and our model, refining the model's assumptions could lead to a more accurate reflection of real-world data.

5 Related work

The seminal work of [WPG+22] provides a comprehensive overview of defi industry and highlights a threat of MEV on the industry. MEV is one of the unique characteristics of the cryptocurrency market that arises due to the transparency of the ledger and the fact that the ledger can not be managed by the closed entity. [QZG22, DGK+20] laid the groundwork for MEV by quantifying blockchain extractable value. In decentralized financial markets on public blockchains, most transactions would be openly processed. All information about senders, receivers, and amounts are publicly available in a memory pool until they are included in a block created by responsible entities such as miners or validators. Therefore, the entities, who have the privilege to manage the transaction order in a block, are incentivized by exploiting that privilege for extra financial gain. Main types of MEV are arbitrage, liquidation, and sandwich attack. Arbitrage and liquidation are typically considered benign for enhancing market efficiency. An arbitrage would be executed by anyone who monitors blockchain state changes and founds a discrepancy in prices between different market. In addition, liquidation is also performed by defi participants who have monitored the state of the blockchain, which has the effect of quickly liquidating the insolvent collateral due to price fluctuations [PWXL21].

On the other hand, sandwich attacks in particular are receiving more attention than other MEV elements because they cause cryptocurrency market instability. Recent researches contribute to a broader understanding and preventing of sandwich attacks [ZQT+21, HW22, ZNW21]. Sandwich attacks typically occur on decentralized exchanges that use the AMM(Automated market maker) model using liquidity pools, and [ZQT+21] formalizes this problem. A sandwich attack basically involves the attacker reordering the victim's transaction sequence, including their own. By rearranging the order of transactions that occur in the liquidity pool of a decentralized exchange and performing front-running and back-running attacks based on victim transactions, the exchange price is manipulated and profits are taken in the process. Some studies have been conducted based on cryptography to improve the way these transactions are reordered, thereby improving vulnerability to sandwich attacks [FP22, AASCY23]. [ZNW21] demonstrates that the trading bots performing sandwich attacks is increasingly achieving more efficiency by analyzing the frequency and profitability of sandwich attack from the large-scale empirical study and proposes that sandwich attacks can be mitigated by splitting a large transaction into several transactions with small amounts.

The work of [HW22] provides valuable insights into mitigating the sandwich attack problem by considering the optimal decisions of attackers and victims in a game-theoretical framework. This work shows that the proposed algorithm providing effective slippage tolerance outperforms the constant

auto-slippage by the AMM, Uniswap. Building upon the findings of [HW22], our study extends the understanding of optimal decision making for attackers and victims. While this study has advanced our understanding, they leave unanswered questions about the variance of expected return of sandwich attack. The variance stems from the fact that the order of transaction in a block could be different according to the priority fee of the transaction. The attackers attempt to acquire the sandwich attack opportunity by front-running and back-running their victim with higher priority fees to extract MEV [QZG22]. At this time, the priority fee is included in the cost to the attacker, and depending on how much this cost is spent, the profit from the attack varies along with the order of sandwich attack transactions it is executed in the block. Since the probability of attack success varies depending on the order of transactions within the block, the distribution of expected profits from sandwich attacks is determined by the priority fee, and this study seeks to present a new model that takes this into account.

6 Conclusion

In this paper, we have explored the multifaceted nature of MEV in the DeFi ecosystem, with a specific focus on sandwich attacks on DEXs. Our research contributes a novel model for determining optimal attack volumes, which diverges from traditional assumptions about attacker behavior in the DeFi market.

The findings presented challenge the conventional belief that attackers always seek to exploit the maximum slippage available to their victims. Instead, our model reveals a more complex strategy, where attackers must weigh the costs of transaction fees against the unpredictability of potential profits. This strategic decision-making is heavily influenced by the concept of *external slippage*, a key element in understanding how attackers navigate the DeFi environment.

Empirical analysis, drawing on real transaction data from Uniswap, lends credibility to our theoretical model. It confirms that transactions with higher priority fees are more likely to be executed earlier within a block, thus affecting the degree of external slippage and the profitability of the attack. This alignment of theory and practice provides a more accurate understanding of the mechanisms at play in the DeFi market.

Overall, this paper not only enhances our understanding of MEV in decentralized finance but also sets the stage for further research. Future studies might explore defensive strategies against MEV, the impact of regulatory measures, and the broader implications of these dynamics on market efficiency and security. In addition, more accurate assumption about the attack amount can reduce the gap between theoretical model and real transaction data. Our findings offer a solid foundation for such future explorations, emphasizing the need for ongoing research in this rapidly evolving domain.

References

[AASCY23] Orestis Alpos, Ignacio Amores-Sesar, Christian Cachin, and Michelle Yeo. Eating sandwiches: Modular and lightweight elimination of transaction reordering attacks. *arXiv preprint arXiv:2307.02954*, 2023.

[AZR20] Hayden Adams, Noah Zinsmeister, and Dan Robinson. Uniswap v2 core, 2020. *URL: https://uniswap. org/whitepaper. pdf*, 2020.

[AZS+21] Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. Uniswap v3 core. *Tech. rep., Uniswap, Tech. Rep.*, 2021.

[DGK+20] 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)*, pages 910–927. IEEE, 2020.

[FP22] Matheus VX Ferreira and David C Parkes. Credible decentralized exchange design via verifiable sequencing rules. *arXiv preprint arXiv:2209.15569*, 2022.

[HW22] Lioba Heimbach and Roger Wattenhofer. Eliminating sandwich attacks with the help of game theory. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 153–167, 2022.

[PJL+24] Seongwan Park, Woojin Jeong, Yunyoung Lee, Bumho Son, Huisu Jang, and Jaewook Lee. Unraveling the mev enigma: Abi-free detection model using graph neural networks. *Future Generation Computer Systems*, 153:70–83, 2024.

[PWXL21] Daniel Perez, Sam M Werner, Jiahua Xu, and Benjamin Livshits. Liquidations: Defi on a knife-edge. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II 25*, pages 457–476. Springer, 2021.

[QZG22] Kaihua Qin, Liyi Zhou, and Arthur Gervais. Quantifying blockchain extractable value: How dark is the forest? In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 198–214. IEEE, 2022.

[WPG+22] Sam Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William Knottenbelt. Sok: Decentralized finance (defi). In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, pages 30–46, 2022.

[ZNW21] Patrick Züst, Tejaswi Nadahalli, and Ye Wang Roger Wattenhofer. Analyzing and preventing sandwich attacks in ethereum. *ETH Zürich*, 2021.

[ZQT+21] Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc V Le, and Arthur Gervais. High-frequency trading on decentralized on-chain exchanges. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 428–445. IEEE, 2021.