The Value of Nontoxic Orderflow to the Uniswap Protocol

Max Holloway *†
max@xenophonlabs.com

January 2023

Abstract

We analyze the relationship between nontoxic orderflow and trading fees on the Uniswap Protocol's Ethereum Mainnet deployment. We formalize a notion of orderflow toxicity, and we demonstrate that nontoxic orders' value to liquidity providers (LPs) is expected to be at least as large as the fees paid by the nontoxic orders. Furthermore, we find that nontoxic orderflow creates additional value to LPs in the form of sandwich attacks; from July 1 to Dec 31 of 2022, sandwich attacks compose over 13% of the volume on USDC-ETH-0.05% and over 26% of the volume on ETH-USDT-0.05% pools. Uniswap LPs benefit immensely from these sandwich trades, due to their fees paid and their limited impact on underlying pool reserves; nontoxic orderflow that gets sandwiched is up to 5 times more valuable to LPs than nontoxic orderflow that does not get sandwiched. We also present an opinionated framework for how much the protocol should value nontoxic orderflow, as well as recommendations for how the protocol should incentivize it. We recommend that the protocol wait until it generates revenue before incentivizing nontoxic orderflow.

*Disclosure: The authors do not own `UNI` token, nor are they affiliated with Uniswap Labs or any of its affiliates. This research was funded by a grant from the Uniswap Foundation. Any opinions and results stated here are those of the authors, not of the Uniswap Foundation or its affiliates. Henceforth any mention of "Uniswap" is in reference to the Uniswap Protocol, unless explicitly stated otherwise. Nothing in this paper should be construed as financial advice or trading advice.

†Acknowledgments: @eljhfx, @AlanaDLevin, @thiccythot_, @0x94305, @_julianma, and @raholloway

1 Introduction

The Uniswap protocol. The Uniswap protocol is a collection of smart contracts that enable liquidity providers (LPs) to passively market make on pairs of fungible tokens. LPs deposit tokens into a pool, and traders can place orders against the liquidity in the pool. The price that traders receive is computed in smart contracts based on the state of the pool; importantly, LPs are not required to change their liquidity positions to facilitate trades on the pool. In contrast to limit orderbooks – where market makers generate revenue by quoting higher asking prices and lower bidding prices – Uniswap's LPs earn a fee on each order that is proportional to the order's volume.

Since for many assets, price discovery happens on off-chain trading venues (or on other DEXs), the prices that Uniswap quotes is often different than those venues. In the case of Uniswap V2, the only way of changing the price quoted by a pool is to swap the tokens in the pool [AZR20]. Although this is technically not the case in Uniswap V3, since LPs can set their liquidity positions with different price bounds, the cost of blockspace makes it impractical for LPs to affect price discovery by updating and cancelling liquidity positions [Ada+21]. This marks another difference between Uniswap and the prevailing orderbook-based centralized exchanges, where liquidity providers can remove limit orders to adjust the price without incurring an order cancellation cost.

Under most conditions, taking the other side of an arbitrageur's trade is a very bad deal for LPs. The more general phenomenon of trading between parties with asymmetric information is known in game theory as *adverse selection*. Perhaps obviously, agents with more information about the "true" value of a financial product than their counterparty can use this information to out-trade them. And since bilateral exchange of common-value assets is a zero-sum game, it follows that the agent who gets the better deal – typically the agent with better information – profits at the expense of their counterparty. This is relevant in the case of Uniswap, where LPs' trading strategy is fixed by the automated market maker, and arbitrageurs make money at the expense of LPs.

Intuitively, LPs should be more profitable when they face non-arbitrage volume. To study this, we introduce the notion of orderflow toxicity.

Defining Nontoxic Orderflow. Before determining the value that nontoxic orderflow creates for the protocol, we begin with the simpler problem of determining the value that it creates for Uniswap LPs. We do this by deeming an order nontoxic if the expected future price does not change given the order's existence; that is, the trade provides no predictive signal for the expected price in the future. What follows is the formal definition.

Definition 1 (h-nontoxicity)**.** *Let P_t be a random variable representing the price of an Uniswap pool's token0 at time t. Then for an Uniswap order, we de ne the net trade vector as the amount of the tokens provided to the pool: $\boldsymbol{a} = [a_0, a_1]^T$ 2 R^2, with units of the respective tokens. We can now de ne what it means for an order to be nontoxic.*

For some xed time lag h 2 $\mathsf{R}+$, we say that an Uniswap order $\boldsymbol{a}$ placed at time t is h-nontoxic if

$$\mathsf{E}[P_{t+h} \mid \boldsymbol{a}] = \mathsf{E}[P_{t+h}].$$

We shed more light on this and related definitions in section 2.

Related work. The profitability of Uniswap LPs is not a new topic of research. Angeris et al. provided analytic formulas for the profitability of Uniswap LPs between discrete points in time [Ang+19]. White demonstrated that Uniswap LPs with nearly-zero fees outperform those with higher fees under specific volatility and drift conditions [Whi+20]. A number of reports have shed light on the historical profitability of Uniswap v3 ([AL22], [LR22], [D22b], [D22a], [0xf22b], [0xf22a]) LPs.

This paper. The aim of this paper is to find the value that nontoxic orderflow creates for the Uniswap Protocol on Ethereum Mainnet, as well as discuss the ways through which the protocol can incentivize this orderflow. We begin in section 2 by finding the marginal revenue that nontoxic

orderflow creates for LPs. In section 3, we provide an opinionated framework for how much the protocol should value an increase in liquidity. We then give recommendations for how the protocol should incentivize nontoxic orderflow – if at all – in section 4. Finally, we provide areas for future work in section 5 and conclude in section 6.

2 Nontoxic Orderflow's Value to LPs

Before determining the value that nontoxic orderflow creates for the protocol, we begin with the simpler problem of determining the value that it creates for Uniswap LPs. We present two lower bounds on this value. The first lower bound only counts the value created by a nontoxic order itself, and the second lower bound counts the value created by sandwich attacks that are caused by nontoxic orders.

2.1 Lower Bound I: Nontoxic Orderflow's Direct Value to LPs

How much value does a nontoxic order create for Uniswap LPs? We define the *token0 volume* of a transaction as $|\Delta a_0|$ = abs(a_0), which has units of token0. Let $\gamma \in (0,1)$ be the fee paid by the trader, $\phi \in \{0, 1/4, 1/5, \ldots, 1/9, 1/10\}$ be the protocol fee. Let the price of the pool at time t be P_t, and let the true price of token0 relative to token1 follow a continuous-time stochastic process S_t. To find how much LPs should value this orderflow, we utilize the markout metric.

Definition 2 (h-Markout)**.** *For a given time lag h and trade $\boldsymbol{a}$ at time t, we define the markout for the order $\boldsymbol{a}$ as*

$$m_{\boldsymbol{a}} = d \cdot |\Delta a| \cdot (P_{t+h} - \bar{p}(\boldsymbol{a})),$$

where d is -1 if the LPs are selling token0 and 1 if the LPs are buying token0, and $\bar{p}(\mathbf{a})$ is the execution price of order $\mathbf{a}$, including fees paid to the LP.

Markout embodies the quality of a trade, from the perspective of a market maker, of a trade placed at time t and measured at a time $t+h$ in the future. While a positive value of markout does not imply that an LP's position is profitable long-term, it *does* imply that the LP's trade is profitable at the time h after the trade; a similar statement can be made for negative markout trades. However, this is not a critical flaw to the metric, for the following reason. If we assume that prices follow a zero-drift geometric Brownian motion (GBM), then we can also demonstrate that for any lag $h' > h$,

$$\mathsf{E}[\text{markout at time } t+h'] = \text{profit at time } t+h = m_{\mathbf{a}}.$$

This follows from the fact that the markout is simply a scalar addition and multiplication of the underlying price process P_t, and if P_t follows a zero-drift GBM for times greater than $t+h$, our h-markout is the the same in expectation as h' markout. In fact, it is not even necessary that P_t follow a GBM; any continuous-time stochastic process P_t that obeys $\mathsf{E}[P_{t+h'}] = \mathsf{E}[P_{t+h}]$ for any $h' > h$ suffices for markout to be equal in expectation at and after the lag-time h.

We now demonstrate that for a given nontoxic order $\mathbf{a}$, Uniswap LPs' expected markout is lower bounded by the fee paid to them by $\mathbf{a}$.

Theorem 1 (Expected markout on a single nontoxic order)**.** *Given an h-nontoxic order a and a continuous-time stochastic process P_t that obeys $E[P_{t+h}] = P_t$, its expected markout for LPs, $\mathsf{E}[m_a]$, is lower bounded by the fees that LPs earn on the order. That is,*

$$\mathsf{E}[m_a] > jjajj \quad p_{avgPostFee} \quad (\gamma(1 \quad \phi)).$$

See appendix D for the proof. This result should be rather intuitive: in expectation, Uniswap LPs make at least the LP fee on each unit of nontoxic volume in the pool. Technically, LPs make slightly more than this value in expectation, due to the fact that nontoxic orders move the pool price. Although this approach has given us a lower bound on the h-markout of a nontoxic order, it ignores the h-markout of other orders that are caused by nontoxic orders. In particular, the nontoxic order h-markout ignores sandwich orders. We now create another lower bound on markout value that takes sandwich attacks into account.

2.2 Lower Bound II: Nontoxic Orderflow's Sandwich Value to LPs

We have demonstrated, for a given order, a lower bound on the LPs' expected markout. Interestingly, some nontoxic orders can *cause* other orders, and in this case, we should ascribe the value of the *caused* order to the nontoxic order that caused it. This is not just a theoretical distinction, since it occurs in real life in the form of sandwich attacks.

A sandwich attack is a type of economic attack that occurs on blockchains that propagate trades via public mempools, such as is the case for orders placed on Ethereum Mainnet through the `app.uniswap.org` trading interface. If a trader *Alice* specifies a worst-case execution price that is far worse than the price she would get from trading on the last block's state, then it is possible for another trader, Bob, to place an order before Alice's order and after Alice's order to "sandwich" her order, such that Bob makes a profit. Henceforth, we refer to Bob's first and second transaction as the front- and back-run transactions, respectively; we will refer to Alice's transaction as the victim transaction. There is extensive existing literature on sandwich attacks and their mitigations ([KDC22], [Züs21], [Zho+21]).

The reason sandwiches play a relevant role in this analysis is due to the fact that sandwiching is typically an attack on nontoxic flow, since nontoxic flow frequently sets looser slippage tolerances and creates sandwich opportunities. Since the front- and back-run orders surrounding the nontoxic victim order also must pay fees, it is clear that those orders also lead to similar fees paid to miners. We observe empirically that these front- and back-run orders are themselves balanced, selling the same amount as that which is purchased – and thus do not demonstrate the directionality that would be needed in order for them to be predictive of future price movements. That is to say, front- and back-run orders appear to be nontoxic, and thus they yield the same markout for LPs as regular nontoxic orderflow.

To find the amount of sandwich value that a nontoxic order creates for LPs, we can utilize either of two distinct approaches: (a) demonstrate optimal sandwich sizes analytically with respect to nontoxic order size and slippage tolerance, and (b) use empirical data to find the amount of sandwich value that historical nontoxic orders have created for LPs. We demonstrate both approaches here.

2.2.1 Quantifying Sandwich Value Analytically

For Uniswap V2, [HW22] derived the optimal sandwich size with respect to a nontoxic order. The resulting analytical solution provides formulas for computing the optimal sandwich input amounts for the front- and back-running trades.

For orders on Uniswap V2 that have no slippage tolerance specified and which are purchasing token1 (without loss of generality), the optimal sandwich order input size is the following:

$$\delta^{o}_{a_x} = \frac{\delta_{v_x}(1 \quad \gamma)^2 x_0 \quad (2 \quad \gamma)\gamma x_0^2}{(2 \quad \gamma)\gamma x_0 \quad \delta_{v_x}(1 \quad \gamma)^2\gamma} + \frac{\sqrt{\delta^2_{v_x}(1 \quad \gamma)^3 x_0(x_0 \quad (1 \quad \gamma)^2\gamma)}}{(2 \quad \gamma)\gamma x_0 \quad \delta_{v_x}(1 \quad \gamma)^2\gamma},$$

where $\delta^{o}_{a_x}$ is the optimal sandwich size, δ_{v_x} is the amount of token0 that the victim is putting into the AMM, and x_0 is the amount of token0 in the pool initially [HW22]. When an order specifies a slippage tolerance s, then the optimal sandwich amount is the following:

$$\delta^{s}_{a_x} = \frac{p\frac{}{n(x_0; \ ; \ v_x; s)} \quad \delta_{v_x}(1 \quad \gamma)^3 \quad (2 \quad \gamma)(1 \quad \gamma)x_0}{1 \quad s \qquad\qquad 2(1 \quad \gamma)^2},$$

where

$$\begin{aligned} n\left(x_0, \gamma, \delta_{v_x}, s\right) = (1 \quad \gamma)^2(1 \quad s)\left(\delta^2_{v_x}(1 \quad \gamma)^4(1 \quad s) + 2\delta_{v_x}(1 \quad \gamma)^2(2 \quad \gamma(1 \quad s))x_0\right. \\ \left. + (4 \quad \gamma(4 \quad \gamma(1 \quad s)))x_0^2\right). \end{aligned}$$

For an algorithm on how we can utilize this method on Uniswap V3, see appendix A.

We can utilize these optimal sandwich order formulas, along with historical on-chain liquidity data, to find the amount that a sandwiched order **a** would create for LPs, in effect allowing us to give a sandwich correction factor that we could add to our markout formula above.

Although this analytical-like method has an appeal, it still requires an algorithm to compute, it requires us to use historical on-chain liquidity data, and it requires us to make gas price assumptions. This makes the method most apt for *ex ante* estimation of the value of nontoxic orderflow. If we can wait to ascribe value to orderflow *ex post*, then we can utilize the empirical approach below in section 2.2.2.

2.2.2 Quantifying Sandwich Value Empirically

As an alternative to determining optimal sandwiching analytically, we can utilize historical sandwich data to measure the amount of value created for LPs as a result of a nontoxic order getting sandwiched.

Data collection methodology. To collect this data, we use the Uniswap V3 subgraph, and the analysis presented here is performed specifically on the ETH-USDC 0.05% fee Uniswap V3 pool. We check that a transaction is sandwiched by utilizing a similar methodology as Johnny Chuang and Anderson Chen in their Uniswap bounty [CC22]. We deem that an order o has been sandwiched if the following conditions hold: (0) there is an address *Addy* that placed two orders in the same block as o, (1) the orders on either side of o were placed by *Addy*, (2) *Addy*'s order before o was in the same direction as o, (3) *Addy*'s order after o was in the opposite direction as o, (4) the orders on either side of *Addy*'s order do not utilize an aggregator or router contract, and (5) the orders on either side of *Addy*'s have a token0 amount within 10% of each other. We apply this filter to all of the swaps in all of the Ethereum blocks from July 1, 2022 - Dec 31, 2022 to create a dataset of sandwiches.

Results. We find that sandwiched orders create an immense amount of value for the LPs of the protocol. The important figures can be found in appendix E, and we present some of the most important figures for the USDC-ETH-0.05% pool below.

Between July 1, 2022 and January 1, 2023, we identified 3,756 sandwich attacks, which attacked over $11B worth of sandwich victim volume [1]. Of the non-front- and back-run transactions, 0.231% were sandwich victim transactions; these victim transactions accounted for 2.38% of the pool's overall swap volume. There were $42,446,355.95 in LP fees paid over this period, and $5,767,261.15 of those fees were paid by sandwich attack front-/back-run transactions. Each dollar of sandwich victim volume led to, on average, over $5 of front- and back-run volume.

[1] We calculate this using the `amount0`, i.e. USDC amount, of the swap. Over this period of time, the price of USDC was stable enough for us to report these metrics without an explicit USDC-USD conversion.

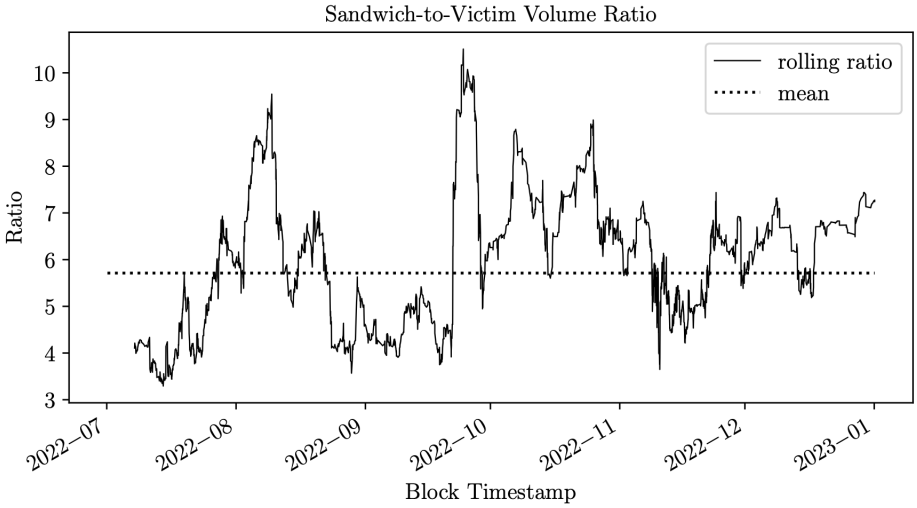

Figure 1: The 100-sandwich rolling ratio of front- and back-run volume to sandwich victim volume for the ETH-USDC 0.05% pool.

We also present plot 2, showing the volume of the sandwich victim and front-/back-run orders. We see that there are two regions of sandwiches, with a cutoff at approximately $10^{4.3} = 30,000$ USDC of front-/back-run volume. This arises due to the two types of sandwich attacks that occur on the USDC-ETH-0.05% pool: single-hop sandwiches and multi-hop sandwiches. Single-hop sandwiches are the simplest type of sandwich attack: user trades A for B, and the sandwicher front-runs with A for B then back-runs with B for A. A multi-hop sandwich is more complicated: user trades A for B for C, and the sandwicher front-runs with A for B for C, then back-runs with C for B for A. In general, most multi-hop transactions occur when one of the user's input or output tokens is relatively illiquid; since these illiquid tokens are easier for sandwich attackers to manipulate the price of, they do not need to trade as much size, and they make up the lower-left-hand-side of the graph. On the other hand, single-hop sandwiches on the USDC-ETH-0.05% pool require significantly more volume, due to the fact that the USDC-ETH-0.05% pool is so liquid. This means that these sandwiches are typically in the 10^5 10^7 USDC range. For example sandwich transactions, see appendix table 1.

3 Nontoxic Orderflow's Value to the Protocol

How much value does nontoxic orderflow create for the protocol? Here we put forward an opinionated approach based on the expected protocol revenues that would materialize in a world with a nonzero protocol fee.

3.1 A Protocol Fee Based Approach

The simplest approach that we have for determining the value that nontoxic orderflow creates for the protocol is to analyze, under various fee switch scenarios, how much revenue the protocol earns as a result of nontoxic orderflow.

Utilizing the sandwich-adjusted value of nontoxic orderflow, we can calculate a lower bound for the value that would go to the protocol as $v_{proto}(x) = (1+m_{sandwich})\ x\ \gamma\phi$, for an order of volume x and a sandwich multiple $m_{sandwich}$. The sandwich multiple can be computed by analyzing historical sandwich attacks and setting it as

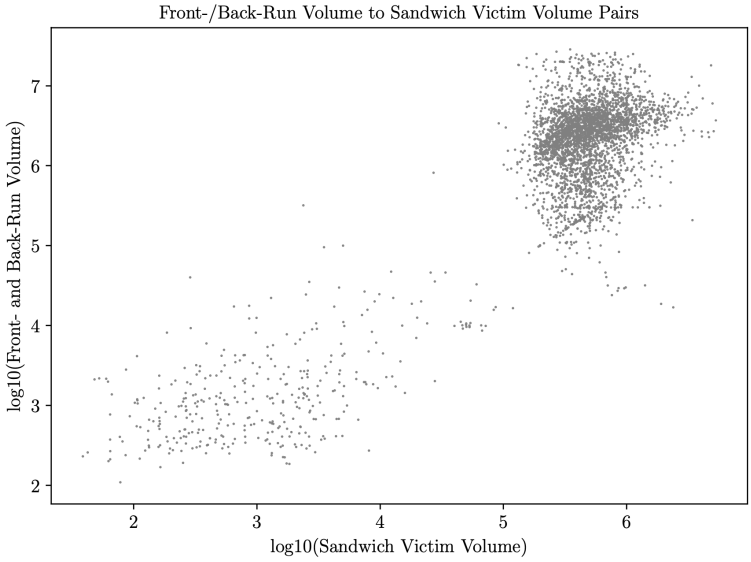

Figure 2: Scatter plot of (sandwich victim volume, front- and back-run) volume pairs. Points in the lower-left are multi-hop sandwiches (sandwiching orders of the form 'A for B for C'), and points in the upper-right are single-hop sandwiches (sandwiching orders of the form 'A for B').

$$m_{\mathit{sandwich}} = \frac{\text{dollars of front-run and back-run volume}}{\text{(dollars of nontoxic volume) - (dollars of front-run and back-run volume)}}. \quad (1)$$

Computing the numerator and the right side of the denominator of equation 1 is straightforward, but computing the left side of the denominator is less obvious. Even if we know an order's observed markout, it is impossible to determine if that order was toxic. With that said, we can create a lower bound on m_{sandwich} by creating an upper bound on the dollars of nontoxic volume, which itself is tractable. See appendix B for further details on orderflow toxicity classification.

For the USDC-ETH-0.05% Uni V3 pool, we estimate the following value of m_{sandwich}:

$$m_{\mathit{sandwich}} = \frac{13.5872\% \text{ of all pool volume}}{(33.0827\% \text{ of all pool volume}) \quad (13.5872\% \text{ of all pool volume})} = 0.6969. \quad (2)$$

That is, each unit of nontoxic volume will beget approximately 0.70 more units of nontoxic volume due to sandwiching. Recall from section 2.2, this method of attributing front- and back-running volume to an order given only the order's size is not accurate for a single order, but is accurate when used across many sandwich orders.

Once we compute the sandwich multiple, we can utilize many potential protocol fees alongside the sandwich multiple to find the value that the protocol would make in the case where there was a protocol fee. See figure 3 for a visual depiction.

With current estimates of the sandwich multiple at 0.70 on the USDC-ETH-0.05% pool, the current value to the protocol is shown for each fee tier in figure 4.

It is important to note that these estimates are contingent on there being roughly the same amount of liquidity in a pool before and after a protocol fee is turned on. Due to the technical and political complexity of the protocol fee, we refrain from analysis of the impact that the protocol fee would have on liquidity. Nevertheless, it is intuitive that the amount of liquidity would decrease in pools that have higher protocol fees, and this would lead to smaller sandwich volumes. Our chart

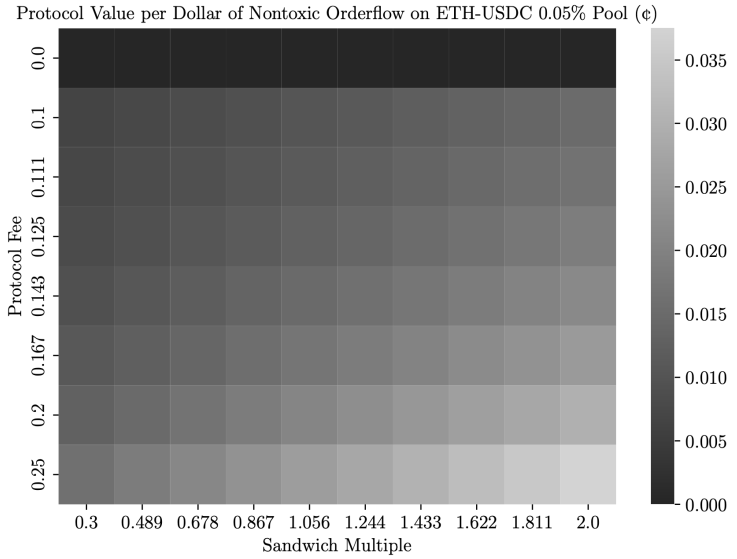

Figure 3: The value per dollar of nontoxic orderflow for various protocol fees and sandwich multiples.

assumes that there is no relationship between the sandwich multiple and the protocol fee, which is not true.

Thus, our dear reader now approaches a fork in the road. If one believes that small values of the protocol fee would not lead to a meaningful decrease in liquidity, then figure 3 demonstrates a lower bound on the average amount of protocol revenue generated from each dollar of nontoxic orderflow. This would allow us to conclude our analysis with a simple description of the value of nontoxic orderflow.

On the other hand, if one believes that an implementation of the protocol fee would lead to a meaningful decrease in liquidity, then the orderflow values here would be over-approximations of the value that nontoxic orderflow creates for the protocol. The next step in this line of research would be to model the relationship between the protocol fee and liquidity, then to model the relationship between the sandwich multiple and liquidity. While this is an excellent opportunity for future research, we do not model the relationship between a protocol fee and liquidity here.

Aside from this approach ignoring the effect of the protocol fee on liquidity, it also possesses the obvious drawback that, at the time of writing, there is no protocol fee. This approach would thus lead us to the unfortunate result that nontoxic orderflow currently has zero value for the protocol. In the following section, we formulate a number of alternative ways that the protocol could value orderflow.

3.2 Non-approaches

We now list a number of intuitive, yet, *in our opinion*, flawed ways of valuing nontoxic orderflow for the protocol.

Non-approach 1: Liquidity is inherently valuable.

Another approach to valuing orderflow is to see how much it would affect liquidity, then ascribe a value on liquidity itself. Accumulating more liquidity would put Uniswap in a position of strength relative to its competitors. For instance, increased Uniswap liquidity leads to an increased share of DEX aggregator volumes, which leads to decreased DEX aggregator volumes for Uniswap's DEX competitors. However, ascribing a number to the value this creates for the Uniswap protocol would be a perilous task, since even in a world where Uniswap defeats all competitors, the protocol must

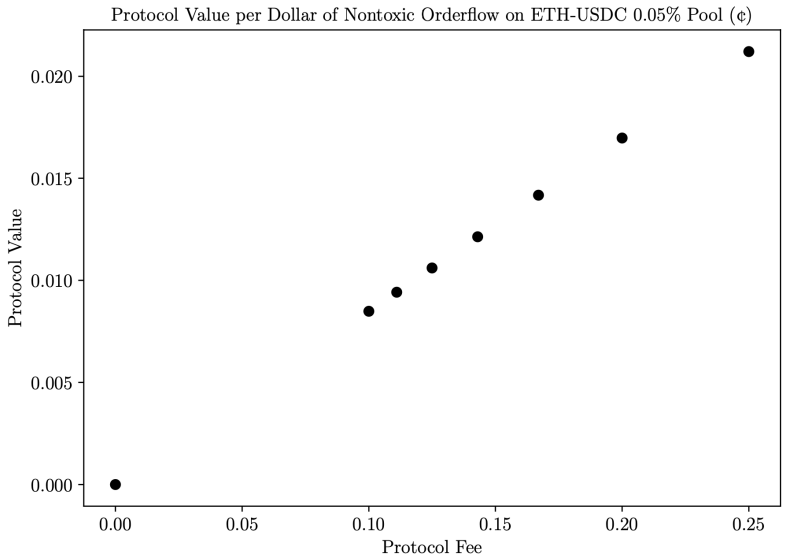

Figure 4: The value per dollar of nontoxic orderflow at the current empirically-observed sandwich multiple of 0.6969.

generate revenue in order for that market dominance value to accrete. Thus we would need to fall back to our previous protocol fee based approach.

Non-approach 2: Uniswap could generate revenue from other sources.

While it is also possible that the Uniswap protocol could generate revenue from other sources than the protocol fee, it would be inappropriate for us to speculate on the existence, let alone the size, of those revenue opportunities. The protocol may revisit this approach if there is a tangible proposal for generating protocol revenues in a way other than the protocol fee.

Non-approach 3: Nontoxic orderflow increases the value of the token.

Yet another approach for quantifying the value that nontoxic orderflow creates for the protocol would be to determine how much value nontoxic orderflow would accrete to the token. While on its surface this approach resembles a value-based management approach to valuing nontoxic orderflow, it is clear that nontoxic orderflow itself will not create value for tokenholders unless there is a revenue source for the token. In order to utilize this method of valuing orderflow, we would either need to default back to our initial approach of assuming there is a protocol fee, or we would need to assume there is another source of revenue, which we refute in non-approach 2.

A similar train of thought is to determine the relationship between nontoxic orderflow and the price of the UNI token. An increase in UNI token value would allow us to grow the protocol's liquidity and volume. Still, this suffers from the issues raised in non-approach 1.

If one is instead optimizing not for the UNI token value, but for the UNI token value *for current holders*, then this approach is more coherent, since increasing the UNI token value would allow current holders to sell the token and realize a gain.

3.3 Takeaways

We provide an approach to measuring the value that the protocol would receive from nontoxic orderflow, assuming a future state where protocol fees are collected, and we find the amount of value that would be created for the protocol if a protocol fee were put in place. The main drawback to this approach is that we do not model the effect that an increase in protocol fee would have on pool liquidity.

Although introducing the protocol fee requires us to make assumptions about future governance actions, we argue that this is the only coherent way of placing value on nontoxic orderflow at this time. All other intuitive methods – the inherent value of nontoxic orderflow, Uniswap generating non-protocol fee revenue, and token price – suffer from other issues that make them unsuitable as a method for valuing nontoxic orderflow.

4 How to Incentivize Nontoxic Orderflow

Assuming we know how valuable a dollar of nontoxic orderflow is for the protocol, we can now discuss ways that the protocol might incentivize it. The incentive design space is large, and we focus only on two approaches: direct user incentives and user interface incentives.

4.1 Incentivizing Users

Perhaps the most intuitive way to incentivize nontoxic orderflow is to directly incentivize the users who produce nontoxic orderflow. Here we describe our best attempt at designing a direct-incentivization mechanism, and we then describe all of the issues that make this mechanism difficult to implement in practice.

A Direct Incentive Mechanism. We begin by initializing a smart contract – potentially on a chain with low gas fees – to hold the rewards, and the Uniswap Foundation treasury funds this smart contract with some quantity R of UNI token. There is a function on the contract, `redeemRewards(amount, rewardsSignature)`, that traders can use to redeem UNI rewards, where they redeem an amount that `amount` that is approved and digitally signed by a Uniswap Foundation-owned address. The amount that an address can claim would be calculated as some unitless multiple, α, of the sum of positive markouts that their orders generated for LPs of the protocol; if $\rho : \mathsf{R}^2 \; ! \; \mathsf{R}^+ \; [\; ;$ is the function mapping from order to reward, it would be calculated as follows:

$$\rho(\mathbf{a}) = \alpha p_U \quad \max(m_{\mathbf{a}}, 0), \tag{3}$$

where p_U is the price of token1 denoted in units of UNI (e.g. p_U). This quantity could be calculated off-chain using an open-source markout calculation that operates on publicly-known blockchain data. This calculation would be performed by a trusted, non-censoring operator, and users could query this operator to create a signature for the user to pass into `redeemRewards`.

Issue #1 with Nontoxic Orderflow Direct Incentive Mechanism: wash trading. Since we are rewarding positive LP markout trades, while not disincentivizing negative LP markout trades, we must ensure that the amount paid for wash trading is negative in expectation, otherwise this can be gamed. For instance, if a wash trader places an Ethereum transaction composed of a buy order $\mathbf{a_1}$ and sale order $\mathbf{a_2}$ at time t, then the wash trader's cost of execution is approximately $\gamma p_U P_t(jj\mathbf{a}jj + jj\mathbf{a}_2jj)$, where P_t is the pool price in units of token1-per-token0 at the time of the swaps. The wash trader's revenue is $\alpha p_U(\max(m_{\mathbf{a}_1}, 0) + \max(m_{\mathbf{a}_2}, 0))$. Thus, for them to be profitable[2], they must have

$$\gamma p_U P_t(jj\mathbf{a}jj + jj\mathbf{a}_2jj) < \alpha p_U(\max(m_{\mathbf{a}_1}, 0) + \max(m_{\mathbf{a}_2}, 0)).$$

Since jja_1jj jja_2jj and $m_{\mathbf{a}}$ $m_{\mathbf{a}_2}$, we can derive a similar profit condition, that

$$2\gamma p_U P_t jj\mathbf{a}jj < \alpha p_U j \max(m_{\mathbf{a}}, 0)j = \alpha p_U jj\mathbf{a}jj \;\; jP_{t+h} \quad \overline{p}(\mathbf{a})j,$$

thus

$$\gamma < \frac{\alpha}{2} \;\; \frac{jP_{t+h} \quad \overline{p}(\mathbf{a})j}{P_t}.$$

[2]Notice that this is a necessary condition for wash trader profitability, but it is not sufficient. In reality, the wash trader would need to pay gas fees and pay slippage, and thus would still have negative profit, even if the left- and right-hand sides of the inequality were equal.

Therefore, if the wash trader aims to be profitable in expectation, they must have

$$\gamma < \frac{\alpha}{2} \; \mathsf{E}\left[\frac{jP_{t+h} \quad \overline{p}(\mathbf{a})j}{P_t}\right] \quad \frac{\alpha}{2} \; \mathsf{E}\left[\frac{jP_{t+h} \quad P_tj}{P_t}\right].$$

Rearranging, we get that the wash trader is profitable only when

$$\alpha > \frac{2\gamma}{\mathsf{E}\left[\frac{jP_{t+h} \quad P_tj}{P_t}\right]}.$$

Wash trading is profitable when the rewards portion is high, pool fees are low, and markout variance is high. The markout variance is directly determined by the variance in the price of the underlying assets in the pool. This makes intuitive sense, due to the fact that wash traders are price-neutral, whereas the protocol must pay out a large quantity in a case where price moves by a large percentage.

In order for the protocol to make wash trading unprofitable in expectation, they would need to set the rewards portion α such that

$$\alpha < \frac{2\gamma}{\mathsf{E}\left[\frac{jP_{t+h} \quad P_tj}{P_t}\right]}. \tag{4}$$

This bound should suffice to avert the wash trade attack on a direct incentivization mechanism.

Issue #2 with Nontoxic Orderflow Direct Incentive Mechanism: it's weird. On the one hand, the concept of incentivizing trades based on the markout they create for LPs is quite strange, since it is effectively a consolation prize for bad trading. On the other hand, it is an interesting idea to refund some of a trader's losses, while still making it better in expectation for the trader to make good trades. Whether this is good or bad is largely a matter of protocol and user preferences.

Issue #3 with Nontoxic Orderflow Direct Incentive Mechanism: it is indifferent to trader size. The fact that this mechanism pays proportionally to each trade's markout is not great. In a perfect world, we may want to incentivize smaller trades more, since they are more likely to be nontoxic. This would lead us to give UNI incentives according to a non-decreasing function of the markout that is concave on positive reals. For instance, instead of rewarding an order $\mathbf{a}$ with $\alpha \max(m_{\mathbf{a}}, 0)$, we would reward it with something like $\beta \log\left(1 + \frac{\max(m_{\mathbf{a}}, 0)}{P_t jj\mathbf{a}jj}\right)$ for some $\beta \; 2 \; \mathsf{R}^+$, or $\min(\max(m_{\mathbf{a}}, 0), \omega)$ for some maximum reward $\omega \; 2 \; \mathsf{R}^+$. The issue here is that concave rewards are not sybil-resistant: rational large traders would simply split their trades into smaller orders such that they can be rewarded higher than they would be if they placed large orders.

Issue #4 with Nontoxic Orderflow Direct Incentive Mechanism: no reason to believe it's sticky. Suppose that direct nontoxic orderflow incentivization was a great success, leading to significantly larger quantity of orderflow, with little-to-no gaming of the mechanism. What should lead us to believe that the nontoxic traders using it will stick around once the incentives are turned off? For more information on quantitative approaches to determining the expected stickiness of a nontoxic orderflow incentivization program, see appendix C.

4.2 Incentivizing User Interfaces

Instead of incentivizing users who create nontoxic orderflow directly, we could instead incentivize the user interfaces that connect users to the protocol. Examples of potential interfaces include Robinhood, Coinbase Wallet, 1Inch Network, Yearn Finance, and Perp Protocol. These interfaces, and others, could provide a gateway for nontoxic orderflow onto Uniswap protocol, and the Uniswap protocol could incentivize these interfaces to send nontoxic orderflow to the protocol. In this section, we review two paradigms that the Uniswap protocol could use to incentivize these interfaces.

4.2.1 Incentivizing Orders

Similar to section 4.1, we could incentivize interfaces based on the quantity of nontoxic orderflow that they generate. The incentives program would closely resemble that of the direct user incentivization, except we would calculate the empirical markout of all of the orders placed through the interface. Interface operators would have a mechanism through which they can signal which orders they processed; this could be done by requiring users sign an attestation that they used the interface. Interface operators would specify an address that should be eligible to withdraw rewards. Aside from these details, we could create an interface incentivization mechanism that mirrors the direct incentive approach in section 4.1.

Benefits of interface order incentivization. There are benefits to incentivizing interfaces based on the orders they process, rather than directly incentivizing users. First of all, if we trust the interface operator to not artificially push their own volume through the interface in order to generate rewards, then we do not need to worry about wash trading. While this trust assumption is unreasonable under the direct rewards mechanism, it may be more reasonable depending on the interface operators involved.

Furthermore, incentivizing interface orderflow retains many of the benefits apparent in the direct incentivization mechanism, chiefly among them the alignment between orderflow producers (users, interfaces) and the protocol's LPs.

Incentivizing interface orderflow is operationally much simpler. The Uniswap Foundation would not need to go through the hassle of creating a rewards claiming interface and the potential security vulnerabilities that would entail. Instead, they would only need to manage contact with a small number of interfaces.

Downsides of interface order incentivization. There are two main downsides to incentivizing interface orderflow: interface incentives would not necessarily accrete to interface users, and these integrations are not necessarily sticky.

The rewards that are paid to interfaces will likely accrete in value for those interfaces, and there is no reason to believe that interfaces will route those rewards back to the users who created in the nontoxic orderflow. While it is possible that the protocol could stipulate that a portion of the rewards go to users of the interface, this would introduce operational complexity. Paying interfaces for nontoxic orderflow is obviously suboptimal, since it effectively leads to worse pricing for users unless the interface gives all of the rewards to its users. However, if this orderflow would not have existed in the absence of the interface, then it may be fair – at least, fair in the Shapley sense [Sha97] – to give a cut of the nontoxic orderflow value to the interface. Nevertheless, the only situations in which the protocol should be willing to pay interfaces are those situations where the orderflow would not have existed without the interface.

The second, larger issue with interface order incentivization is that it does not necessarily lead to sticky orderflow. Once the incentive program halts, so do the rewards for the interface. Unless interface operators agree otherwise, this means that the interface has no further incentive to route orderflow to Uniswap. Of course, they may continue to route orderflow to Uniswap, but there is no reason to believe that they would do so. Since the topic of interface orderflow stickiness relies so much on the particular interface being incentivized, we refrain from further speculation on interface orderflow stickiness.

4.2.2 Incentivizing Integrations

Yet another approach for incentivizing interface orderflow is by incentivizing interfaces to integrate with the Uniswap protocol. For instance, if a team is building an interface, the Uniswap Foundation could sponsor the development work needed for the interface to integrate with the protocol. Or, similarly, the protocol could hire developer relations person or small team to help evangelize the Uniswap protocol and help customers to integrate. By sponsoring an interface integration, the protocol would effectively invest in the future orderflow created by that interface.

4.3 Incentivizing Sandwiches

In sections 4.1 and 4.2, we utilize the observed markout when determining the rewards that an order should receive, despite the fact that nontoxic orders also occasionally lead to value from the front- and back-run orders that surround them. Why should the protocol not also pay for the value created by the front- and back-run orders?

We offer no answer to this question, but we do provide two perspectives.

Why the protocol should incentivize sandwich orders. In a world where the Uniswap protocol generates revenues on its volume, it clearly benefits from the volume generated by sandwich attacks. As we demonstrated in section 3, sandwich volume accounts for approximately 13% of total volume on the USDC-ETH-0.05% pool, and each dollar of fees paid by nontoxic volume generates an additional 70¢ of sandwich fees, on average. Ignoring sandwich fees when incentivizing the pool would be to ignore almost half of the value created by nontoxic orderflow.

Why the protocol should not incentivize sandwich orders. On the other hand, incentivizing sandwiched orders leads to worse user outcomes in the case of an interface orderflow incentivization program. Sandwich revenues come directly out of the pockets of users (see Theorem 2 in [HW22] for the proof).

Final word on sandwiches. Whether the protocol should give additional incentives for sandwiched orders is a decision for the protocol's tokenholers. While we are happy to provide as much supporting information as possible for the protocol to make this decision, we do not have a recommendation regarding sandwich incentivization.

4.4 When to Give Incentives

Since the Uniswap Protocol currently has the protocol fee set to zero on all pools, it is natural to ask: should the protocol incentivize nontoxic orderflow today? We argue that the protocol should *not* run an incentive program that rewards based on nontoxic orderflow, and we recommend that the protocol also not pursue interface incentives except for cases with time-sensitive strategic appeal.

Suppose, to the contrary, that the protocol should incentivize nontoxic orderflow while the protocol fee is set to zero. Then, if we utilize the aforementioned revenue-based framework for valuing this orderflow, we would need to reconcile the incentive program spending by an expectation of future revenues. If we assume that future revenues would take the form of a protocol fee, we would need to make an assertion that (a) the protocol fee will be turned on in the future, and (b) nontoxic orderflow today would lead to an increase in revenues after the protocol fee is turned on. For (b) to be true, it would need to be the case that nontoxic orderflow today results in increased volumes in the future (since revenue is proportional to volume). The best theory for how this would occur is via a liquidity-orderflow feedback loop, whereby increases in nontoxic orderflow today lead to greater exchange liquidity, which leads to increases in nontoxic orderflow, etc. We analyze this feedback loop in appendix C, and we find that due to the fact that nontoxic orderflow is not price sensitive, this feedback loop's total addressable nontoxic volume is capped by DEX aggregator volumes. This is not a strong feedback loop. We believe that nontoxic orderflow would not lead to a meaningfully large feedback loop between orderflow and liquidity, and thus would not lead to increased future revenues. Thus, the Uniswap Protocol should not incentivize nontoxic orderflow prior to charging a protocol fee.

Of course, there are exceptions to this argument. For instance, if there were a high profile interface (e.g. Robinhood) interested in an integration, it could be worthwhile to incentivize that interface, in anticipation of growing that partnership in advance of charging protocol fees or other protocol revenues.

4.5 Recommendation

We recommend that the protocol wait until it generates revenue before implementing a nontoxic orderflow incentivization program. Without revenue, we have no coherent way of valuing the nontoxic orderflow that the protocol would generate via a nontoxic orderflow incentives program.

Once the protocol generates revenue, we believe that interface integration incentives are the most cost-efficient route to growth of nontoxic orderflow. If these opportunities are sparse, then we recommend that the protocol advance on interface orderflow incentives. We also believe that direct incentives have the potential to create substantial value, however we believe that they require further research on their game theoretic properties.

5 Future Work

Sandwich Attack Data Analysis. We queried 6 months of Uniswap swaps for a number of top Uniswap pools. When we started analyzing the sandwich data, we were surprised by the magnitude and informational properties of sandwich attack. It would be great to see more work in the following areas: finding the source of the sandwich volume, explaining the massive variance in sandwich attack sizes, and exploring sandwich attack mitigations at the protocol or interface level.

Optimal Interface Orderflow Procurement. When procuring orderflow from interfaces, how much should Uniswap be willing to pay? One approach, which we provide in our paper, is for Uniswap to state its price on a per-interface basis, then negotiate with each interface; this is akin to running multiple posted-price auctions. Instead, Uniswap could procure interface orderflow by publishing a request for orderflow, then allowing interfaces to bid in advance to determine how much Uniswap protocol would pay. Utilizing a multi-unit Dutch-style auction may allow Uniswap to procure this orderflow at a deep discount, based on interfaces' willingness to route at lower prices.

Alternative Revenue Streams. The protocol fee is the canonical built-in revenue generation mechanism, but are there other ways that the protocol could generate revenue? One idea, akin to Skip protocol's Osmosis solution, is for the Uniswap Foundation to sponsor the creation of a block builder that back-runs users' transactions and give the proceeds to the protocol's treasury; see also A^2MM for a similar idea [ZQG21].

Behavioral Economics of Retail DeFi Users. Our analysis was based in financial economics and on-chain data, but we do not analyze the actual people creating nontoxic orders. What consumer incentive programs have worked – and not worked – historically? Can we apply those learnings to Uniswap? Also, what is the profile of an Uniswap user who generates nontoxic orderflow, and how can we find more customers who fit their same customer profile?

The Orderflow-Liquidity Feedback Loop. Feedback loops are interesting, as they can tend toward exponential growth or decay. We briefly outline a feedback loop between orderflow and liquidity in appendix C, but there is still ample greenfield work to be done on this topic. What is the relationship between nontoxic orderflow and liquidity? The relationship between liquidity and DEX aggregator volume? Can Uniswap dominate DEX aggregator volume by dominating an orderflow-liquidity feedback loop, and if so, is there a low-cost strategic move that would catalyze this?

6 Conclusion

In this paper, we formalize an Uniswap-specific notion of orderflow toxicity, as well as propose an Uniswap-specific markout metric to measure it.

We were surprised to find that the value of nontoxic orderflow to LPs may be over 1.5x that of the fees paid by nontoxic traders, due to the large quantity of sandwich volume. This leads the protocol to an ethical dilemma: should the protocol give boosted incentives to interfaces that

generate sandwiched trades, or should the protocol avoid sandwich incentives and generate less volume?

Irrespective of the sandwich dilemma, we advise that the protocol refrain from any incentive program until the protocol generates revenue. If and when it does generate revenue, we suggest that the protocol incentivize interface integrations, and potentially also interface orderflow. These incentives, if carefully implemented, could meaningfully increase the quantity of nontoxic orderflow, helping the exchange to grow sustainably in the long run. We hope that the Uniswap community can harness this research to grow the protocol.

A Computing an Optimal Sandwich

Here we give a simple algorithm for computing optimal sandwich attacks on Uniswap V3. In words, we do roughly the following.

We check if the optimal sandwich order size surpasses the current tick; if it does, then the order size is the amount needed to get to the next tick, plus the result of the optimal arbitrage formula recursed on the next optimal sandwich solution; if it does not, then use the optimal sandwich solution. If in the recursive step or the base case the trade is unprofitable, then set the optimal amount to 0. This algorithm's correctness comes from the fact that sandwich profits w.r.t. trade size are single-peaked, and thus we can optimistically cross as many ticks as necessary to get to the optimal sandwich that ignores gas costs, and if we are ever unprofitable, we can simply lower our sandwich input amount.

See the following figure for python-like pseudocode implementation.

```
def optimal_sandwich_input(victim_order, curtick, pool_data, gas_fee_per_tick,
                                          gas_fees_incurred=0):
    """
    Assuming there's a victim order that's buying token0, here's
    an algorithm for computing the optimal sandwich input size of
    token1. A similar algorithm can be used to compute the optimal
    sandwich input when the victim is selling token0.
    """
    # optimal sandwich size if trading on Uniswap V2
    v2_size = v2_optimal_sandwich_size(victim_order, pool_data)
    curtick_max_size = get_size_until_tick_crossing()

    if v2_size > curtick_max_size:

        # find the optimal rest of the trade to perform, under the
        # optimistic assumption that it's worth it to trade on this
        # tick
        gas_fees_incurred += gas_fee_per_tick
        rest_of_order_size = optimal_rest_of_sandwich(victim_order, curtick+1,
                                                      pool_data, gas_fee_per_tick,
                                                      gas_fees_incurred)

        if trading_this_tick_itself_is_profitable(gas_fee_per_tick,
                                                  curtick_max_size):
            return curtick_max_size + rest_of_order_size
        else:
            return 0

    else:
        if trading_this_tick_itself_is_profitable(gas_fee_per_tick, v2_size):
            return v2_size
        else:
            return 0
```

B Identifying Nontoxic Orders

At various points in this analysis, it has been useful for us to classify historical swaps as toxic or nontoxic. However, since our definition of nontoxic orderflow depends on probabilistic statements, we cannot verify with certainty whether an order was nontoxic based just on historical data like markout. Here, we share a few approaches that can be utilized to determine if an order is toxic.

Approach I: where the order came from. Most Uniswap swaps are not placed directly on the Uniswap Core smart contracts, and instead they go through an order router. The Uniswap protocol has an order router, and other DEX aggregator protocols also offer order routers. These routing protocols cause orders to pay additional gas fees that they would not incur if they went directly through the Uniswap core routers. Since CEX-DEX and DEX-DEX arbitrage are (presumably) very competitive, and thus sensitive to gas fees, swaps that go through routers are likely not toxic. Furthermore, it is straightforward to determine if an order went through a router by checking the Ethereum transaction's `msg.sender` field. This provides us with a quick and easy classifier for orderflow toxicity.

Approach II: observed markout + Bayesian classifier. For each swap, calculate the avg execution price, then look at the price after h, and if the swapper lost money by time h, count the order as nontoxic. Conversely, if the swap is profitable, then we do not necessarily know that it is toxic, since we would expect that approximately half of the nontoxic orders are profitable (although possibly a bit less than half, depending on the pool fees). Then for an order o, we can use Bayes' rule to calculate

$$\begin{aligned}\Pr(o \text{ is nontoxic} \mid o \text{ is profitable}) &= \frac{\Pr(o \text{ is profitable} \mid o \text{ is is nontoxic}) \quad \Pr(o \text{ is nontoxic})}{\Pr(o \text{ is profitable})} \\ &\quad \frac{.5 \quad p}{(1 \quad p) \quad 1 + (p) \quad .5} \\ &= \frac{.5p}{1 \quad .5p},\end{aligned}$$

where p is the proportion of swaps that are nontoxic, which it unknown. However, we do know the proportion of observed swaps that lose money, and let us call this quantity s. Let s be the proportion of unprofitable swaps, then since we expect nontoxic orders to lose money about half of the time, we have that $p = 2s$, and thus we have

$$\Pr(o \text{ is nontoxic} \mid o \text{ is profitable}) = \frac{s}{1 \quad s}.$$

Then, to compute the sandwich multiplier using this number, we would look at all of the profitable swaps and assign them 'toxic' with probability $\frac{s}{1 \; s}$. This would allow us to calculate a single value of $m_{sandwich}$, and then we could re-run this again, and again, etc until we converge on a value of $m_{sandwich}$.

The reason we do not use this approach is that it relies too heavily on the definition of profitable trades, which itself relies on the markout interval h. We leave it as an interesting area of future work to see if $m_{sandwich}$ derived from this approach is the same as the $m_{sandiwich}$ that we derive using approach I.

C The Orderflow-Liquidity Feedback Loop

The model we provide for valuing orderflow in the paper is meant to give a lower bound on the orderflow value, and we do so by looking only at the amount of value created nearly immediately following a nontoxic order, where liquidity is held constant. While we assume that this model makes sense for small additions of orderflow, it is clear that if a sufficiently large amount of nontoxic orderflow comes in, then liquidity will increase. Here, we present a rough model for how one might model these interaction effects between nontoxic orderflow and liquidity. This section is intended for nerd-sniping purposes only, and should be read as a first-pass on the interaction effects between nontoxic orderflow and liquidity; we do *not* endorse this methodology.

C.1 The model

Suppose there are types of of orderflow that go to Uniswap pools: arbitrage volume, DEX aggregator volume, and retail volume. Let t denote time, and initially $t = 0$. Let L_t be a metric for liquidity at time t; we do not define what this metric should be. Let $v_{arb;t}$ be the rate of arbitrage volume at time t; $v_{agg;t}$ be the rate of DEX aggregator volume at time t; let $v_{ret;t}$ be the rate of retail volume at time t. Assume the following relationships hold:

$$\begin{aligned}\frac{\partial L_t}{\partial v_{arb;t}} &= 0,\\ \frac{\partial L_t}{\partial v_{agg;t}} &= \beta_{agg}, \text{ and}\\ \frac{\partial L_t}{\partial v_{ret;t}} &= \beta_{ret},\end{aligned}$$

for some β_{agg}, β_{ret} 2 R^{+}.

These β values should be interpreted as the elasticities of liquidity, with respect to the different types of volume. Implicit here is that an increase in arbitrage volume will not cause an increase in liquidity. Assume that all of the liquidity is explained by the volume, such that

$$L_t = L_{agg;t} + L_{ret;t} = \beta_{agg} v_{agg;t} + \beta_{ret} v_{ret;t}, \tag{5}$$

where $L_{agg;t}$ is the liquidity due to aggregator volume, and $L_{ret;t}$ is the liquidity due to retail volume.

We assume that there exist $L_{rest;t}$ units of liquidity across other DEX aggregator-reachable venues; we also assume that the the total rate of volume coming from a DEX aggregator at time t is $V_{agg;t}$. Now we make what is perhaps our most egregious assumption, that the rate of DEX aggregator volume given to the Uniswap pool is proportional to the Uniswap pool's liquidity share:

$$v_{agg;t} = \frac{L_t}{L_{rest;t} + L_t} V_{agg;t}. \tag{6}$$

Combining equations 5 and 6, we get that if retail volume changes between time 0 and time t, and causes a change in liquidity, but the rest of the DEXs' liquidity stays the same, then the rate of aggregator volume can be represented as the following:

$$\begin{aligned} v_{agg;t} &= \frac{(\ \beta_{agg} + L_{ret;t}) \quad \sqrt{(\beta_{agg} \quad L_{ret;t})^2 + 4\beta_{agg}(V_{agg;0} L_{ret;t} \quad L_{rest;0})}}{2\beta_{agg}} \\ &= \frac{(\ \beta_{agg} + \beta_{ret} v_{ret;t}) \quad \sqrt{(\beta_{agg} \quad \beta_{ret} v_{ret;t})^2 + 4\beta_{agg}(V_{agg;0} \beta_{ret} v_{ret;t} \quad L_{rest;0})}}{2\beta_{agg}}. \end{aligned} \tag{7}$$

Given the changes in aggregator volume, we get that the new arbitrage volume is

$$\begin{aligned} v_{arb;t} &= \omega L_t \\ &= \omega \left(\beta_{agg} v_{agg;t} + \beta_{ret} v_{ret;t} \right). \end{aligned} \tag{8}$$

We can then calculate the quantity of fees generated by the protocol as follows:

$$F_t = \gamma \phi (v_{arb;t} + v_{agg;t} + v_{ret;t}), \tag{9}$$

where F_t is the rate of protocol fees per unit time.

We could then use this fee function to calculate the change in fees, ΔF_t, that we would expect given a change in retail trades, $\Delta v_{ret;t}$. This $\Delta F_t / \Delta v_{ret;t}$ quantity would account for the interaction effects between arbitrageurs, DEX aggregators, and retail traders; by comparison, the lower bound we establish in the paper ignores these interaction effects.

C.2 Discussion

Issues Although this model of the interaction effects is interesting, there is no indication that it is correct. We ascribe continuous functions everywhere: a continuous relationship between liquidity and DEX aggregator volume rate, a constant increase in liquidity per unit of aggregator volume rate, an constant increase in liquidity per unit of retail volume rate, etc. These models are clearly false in real life; all of these codependencies between liquidity and volume must be implemented by economic agents, and we have no indication that this is a realistic assumption. Furthermore, even if this model were sufficient, it would be quite challenging to measure the β and ω parameters.

Learnings Despite its issues, we believe this model provides a helpful mental framework for assessing the interactions between orderflow and liquidity. For instance, we would not be surprised if the relationship between aggregator volume rate and liquidity followed a similar function to that provided in equation 6 for large changes in liquidity L_t. But this is speculation.

Interestingly, if this model's assumption that liquidity is not responsive to arb volume, that retail volume is not responsive to liquidity, and that DEX aggregators' rate of volume is fixed, is correct, then it would seem that this is a negative feedback loop that reaches equilibrium at a finite value of DEX aggregator and arbitrage volume. Of course, this should be trivial, given the fact that arbitrage volume is the only infinitely scalable volume term in this model, and we assume that it does not lead to increases in liquidity. In fact, we hypothesize that this would be a feedback loop under much weaker conditions, for any setup where the (increase in retail volume) $\rightarrow$ (increase in DEX liquidity) $\rightarrow$ (increase in retail volume) loop has a finite sum over all increases in retail volume. Weakening our model's assumptions would be an excellent further area of research on this problem.

D Proof of Markout Lower Bound

Proof. We begin by obtaining an expression for $\bar{p}(\mathbf{a})$, where x represents an amount of token0 and y represents an amount of token1.

$$\bar{p}(\mathbf{a}) = \frac{y_{intoAmm} \quad (1 \quad \gamma\phi)}{x_{outOfAmm}} = \frac{(y_{reserveChange}/(1 \quad \gamma)) \quad (1 \quad \gamma\phi)}{x_{reserveChange}} = \frac{1 \quad \gamma\phi}{1 \quad \gamma} \quad p_{avgPostFee}.$$

This allows us to represent markout in the following way

$$\begin{aligned} m_{\mathbf{a}} &= (\quad 1) \;\; jj\mathbf{a}jj \;\; (P_{t+h} \quad \frac{1 \quad \gamma\phi}{1 \quad \gamma} \;\; p_{avgPostFee}) \\ &= jj\mathbf{a}jj \;\; (\quad P_{t+h} + \frac{1 \quad \gamma\phi}{1 \quad \gamma} \;\; p_{avgPostFee}) \\ &\quad jj\mathbf{a}jj \;\; (\quad P_{t+h} + (1 + (1 \quad \phi)\gamma) \;\; p_{avgPostFee}) \\ &= jj\mathbf{a}jj \;\; p_{avgPostFee} \;\; \left(\gamma \quad \gamma\phi + \frac{p_{avgPostFee} \quad P_{t+h}}{p_{avgPostFee}}\right), \end{aligned}$$

using the Taylor approximation of $\frac{1}{1 \quad}$ around $\gamma = 0$ for the approximate-equal step.

With this definition for markout, we may now move on to determining the value that a nontoxic order $\mathbf{a}$ would create for the LPs of a pool. When $d = \quad 1$, we would have

$$\begin{aligned} \mathsf{E}[m_{\mathbf{a}}] &= \mathsf{E}\left[jj\mathbf{a}jj \;\; p_{avgPostFee} \;\; \left(\gamma \quad \gamma\phi + \frac{p_{avgPostFee} \quad P_{t+h}}{p_{avgPostFee}}\right) \; j \; \mathbf{a}\right] \\ &= jj\mathbf{a}jj \;\; p_{avgPostFee} \;\; \left(\gamma \quad \gamma\phi + \frac{p_{avgPostFee} \quad \mathsf{E}\left[P_{t+h} \; j \; \mathbf{a}\right]}{p_{avgPostFee}}\right). \end{aligned}$$

Since $\mathbf{a}$ is h-nontoxic, it follows that $\mathsf{E}[P_{t+h} \; j \; \mathbf{a}] = E[P_{t+h}]$, and we find that

$$\begin{aligned} \mathsf{E}[m_{\mathbf{a}}] &= jj\mathbf{a}jj \;\; p_{avgPostFee} \;\; \left(\gamma \quad \gamma\phi + \frac{p_{avgPostFee} \quad \mathsf{E}\left[P_{t+h} \; j \; \mathbf{a}\right]}{p_{avgPostFee}}\right) \\ &= jj\mathbf{a}jj \;\; p_{avgPostFee} \;\; \left(\gamma \quad \gamma\phi + \frac{p_{avgPostFee} \quad \mathsf{E}\left[P_{t+h}\right]}{p_{avgPostFee}}\right). \end{aligned}$$

If we assume that $\mathsf{E}[P_{t+h}] = P_t$, then we find

$$\begin{aligned} \mathsf{E}[m_{\mathbf{a}}] &= jj\mathbf{a}jj \;\; p_{avgPostFee} \;\; \left(\gamma \quad \gamma\phi + \frac{p_{avgPostFee} \quad \mathsf{E}\left[P_{t+h}\right]}{p_{avgPostFee}}\right) \\ &= jj\mathbf{a}jj \;\; p_{avgPostFee} \;\; \left(\gamma \quad \gamma\phi + \frac{p_{avgPostFee} \quad P_t}{p_{avgPostFee}}\right) \\ &> jj\mathbf{a}jj \;\; p_{avgPostFee} \;\; (\gamma(1 \quad \phi)). \end{aligned}$$

This demonstrates the desired result for $d = \quad 1$, the case where the pool is selling token0. A similar result can be shown for the case of $d = 1$. □

E Additional Data and Graphs

	Transaction Hash
USDC-ETH-0.05% Front-run Transaction	0x388f697fe428b4d3124c943673e64712a22d244548ff111e528b609add5a3bdd
USDC-ETH-0.05% Victim Transaction	0xc89645dca26308fb5c02a5ba884ba37e3a8105e427b1c33ce9a835be79e02210
USDC-ETH-0.05% Back-run Transaction	0x3bd54dc1de6f8de4f50862249df8a4b1ab5020d0c3ae9d0ebcfe80650b3465f4
ETH-USDT-0.05% Front-run Transaction	0x459c3ac6fac91eaa3cfae16195aafc9b5025428efe300b0f4050426fe89a683f
ETH-USDT-0.05% Victim Transaction	0x1cdae41c0ecc86858fe3b72e88b9f61067b000bc4cbcf8e28ddb3f7d86643496
ETH-USDT-0.05% Back-run Transaction	0xeecb545a2512fafb9bb0d64962708fe162a45d3f7cb07127cdccb2a4e9c55f08
DAI-USDC-0.01% Front-run Transaction	0xe069c3cd0805e38bb1a56417107eda097b0edac8bd091c983d9680cc86c21005
DAI-USDC-0.01% Victim Transaction	0xbf8cd2a37d7ec6ad9ec4b405d33bc31c4a796d28880a647d69f49bad7d6ab854
DAI-USDC-0.01% Back-run Transaction	0xed873ada8a3b448edfd4806378ce238bbe7be2f0594340815bc05b6133818efb
LDO-ETH-0.30% Front-run Transaction	0x27c388e5eb45b6397e9d0f6e56bcde79b6e40034796d74722e3029babf31f11b
LDO-ETH-0.30% Victim Transaction	0xd269227266ea5fa237612547596fe825fbe81d142a1fa33b07562fd3aa108b6a
LDO-ETH-0.30% Back-run Transaction	0x1bcb98a5e82456063c2b600b3927806c86be200cba1b4d57d04285a0e02db997

Table 1: Some example sandwich attacks on Uniswap V3 pools.

	Nontoxic Volume %	Front- Back-run Volume %	Victim Volume %	Sandwich Count	Sandwich Multiple
USDC-ETH-0.05%	33.0827	13.5872	2.3787	3756	0.6969
ETH-USDT-0.05%	44.869	26.8018	1.44971	2765	1.4834
DAI-USDC-0.01%	46.2412	0.506031	0.0577408	9	0.0111
LDO-ETH-0.30%	29.0025	1.87641	1.58727	85	0.0692
USDC-ETH-0.30%	14.2243	0.0872036	0.0533355	4	0.0062
USDC-USDT-0.01%	44.695	2.87004	0.391578	513	0.0686
WBTC-ETH-0.05%	32.3354	14.5738	2.64851	831	0.8205

Table 2: Pool-level nontoxic volume and sandwich data.

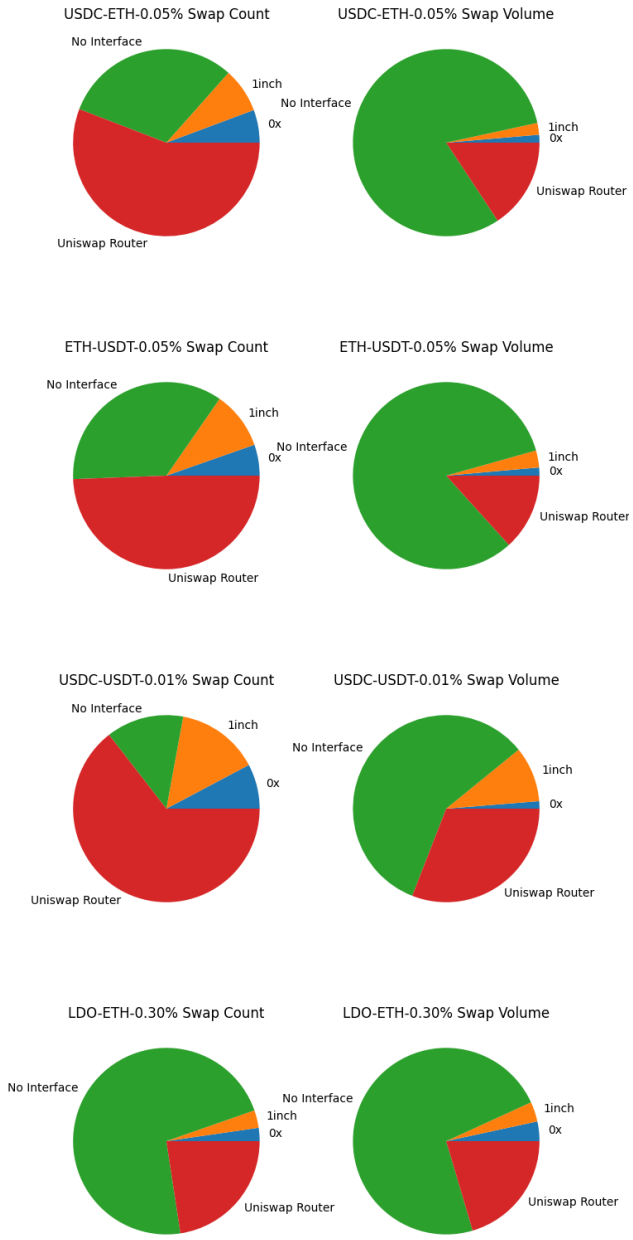

Figure 5: The counts (left column) and volume (right column) of swaps that originate from each interface. We see that although most *orders* come from the Uniswap router in each of these pools, most of the *volume* does not go through any known router at all.

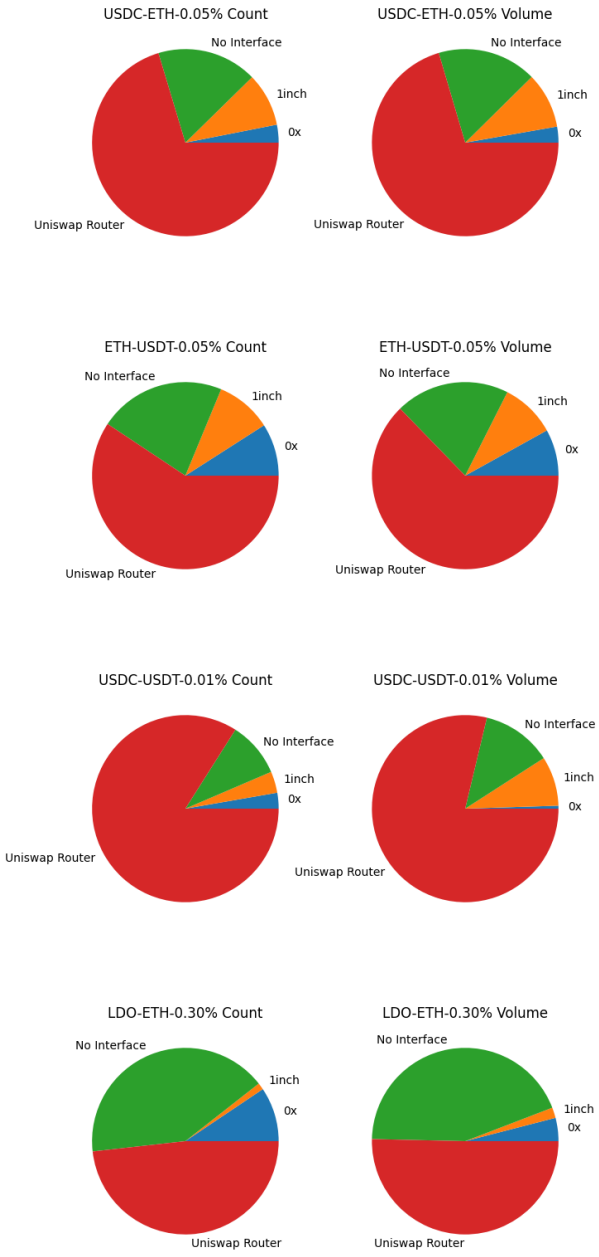

Figure 6: The counts (left column) and volume (right column) of sandwich victim orders that originate from each router. We see that most sandwiched orders and sandwiched volume come from the Uniswap router for each of these pools.

References

[0xf22a] 0xfbifemboy. *Follow-Up Analyses of LP Pro tability in Uniswap V3*. Accessed 10 Jan 2022. 2022. URL: https://crocswap.medium.com/follow-up-analyses-of-lp-profitability-in-uniswap-v3-2cfc8c5e014e.

[0xf22b] 0xfbifemboy. *Usage of Markout to Calculate LP Pro tability in Uniswap V3*. Accessed 10 Jan 2022. 2022. URL: https://crocswap.medium.com/usage-of-markout-to-calculate-lp-profitability-in-uniswap-v3-e32773b1a88e.

[Ada+21] Hayden Adams et al. "Uniswap v3 Core". In: *Tech. rep., Uniswap, Tech. Rep.* (2021).

[AL22] Austin Adams and Gordon Liao. *When Uniswap v3 returns more fees for passive LPs*. Accessed 10 Jan 2022. 2022. URL: https://uniswap.org/blog/fee-returns.

[Ang+19] Guillermo Angeris et al. "An analysis of Uniswap markets". In: *arXiv preprint arXiv:1911.03380* (2019).

[AZR20] Hayden Adams, Noah Zinsmeister, and Dan Robinson. "Uniswap v2 Core". In: (2020).

[CC22] Johnny Chuang and Anderson Chen. *Uniswap Bounty #19 - MEV, Sandwich Attacks and JIT*. Accessed 4 Jan 2022. 2022. URL: https://coinomo.notion.site/Uniswap-Bounty-19-34df5d8d69e54b2ba10f5b5799758d26.

[D22a] Alex D. *DeFi Deep Dive: Uniswap Part 2*. Accessed 10 Jan 2022. 2022. URL: https://medium.com/friktion-research/defi-deep-dive-uniswap-part-2-8be77a859f47.

[D22b] Alex D. *Uniswap V3 Toxicity Report*. Accessed 10 Jan 2022. 2022. URL: https://dune.com/thiccythot/uniswap-markouts.

[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*. 2022, pp. 153–167.

[KDC22] Kshitij Kulkarni, Theo Diamandis, and Tarun Chitra. "Towards a theory of maximal extractable value i: Constant function market makers". In: *arXiv preprint arXiv:2207.11835* (2022).

[LR22] Gordon Liao and Dan Robinson. *The Dominance of Uniswap v3 Liquidity*. Accessed 10 Jan 2022. 2022. URL: https://uniswap.org/TheDominanceofUniswapv3Liquidity.pdf.

[Sha97] Lloyd S Shapley. "A value for n-person games". In: *Classics in game theory* 69 (1997).

[Whi+20] Dave White et al. *Uniswap's Financial Alchemy*. Accessed 10 Jan 2022. 2020. URL: https://research.paradigm.xyz/uniswaps-alchemy.

[Zho+21] Liyi Zhou et al. "High-frequency trading on decentralized on-chain exchanges". In: *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2021, pp. 428–445.

[ZQG21] Liyi Zhou, Kaihua Qin, and Arthur Gervais. "A2mm: Mitigating frontrunning, transaction reordering and consensus instability in decentralized exchanges". In: *arXiv preprint arXiv:2106.07371* (2021).

[Züs21] Patrick Züst. *Analyzing and Preventing Sandwich Attacks in Ethereum*. 2021.