Pleased to share the outputs of my research on the game theory of exotic blockchain queues of the type exemplified by products such as MEV-Share and MEV-Blocker.

This type of preprocessing layer facilitates the orderly insertion of backrun transactions that can be used to perform useful market functions such as arbitrage, while at the same time making it harder to profit from frontrunning or sandwiching by obscuring information about ordering or transaction contents. Nonetheless, a searcher can still profit in expectation by executing blind sandwiches, afflicting traders with “manipulated” prices.

I defined a probabilistic model for a generalised sort of two-lane “backrun queue” trading on a CFMM and obtained some results about an arbitraging/sandwiching searcher’s dominant strategy. In particular I found a way to quantitatively bound how much prices on the target pool deviate from global equilibrium as a result of sandwiching.

As always, comments and questions welcome!

- Blog post, reproduced below.
- Arxiv paper.

Introduction

A network of footpaths and maintenance access roads has sprung up at the outskirts of Ethereum’s dark forest.

An order dispatched by a trader to a DEX on Ethereum today passes through several layers of preprocessing infrastructure before ultimately landing on the chain and being executed. These layers are populated by sophisticated bots that perform allocative computations such as fulfilling intents and allocating blockspace. Collectively, the infrastructure is known as the MEV supply chain

or [transaction supply network

](https://frontier.tech/infinite-games).

Transaction supply networks provide new ways for algorithmic traders to observe and exploit opportunities created by incoming transaction commitments or the information that they reveal. This exploitation is often termed MEV extraction,

a term that has perhaps accrued some negative connotations. Sure enough, some types of such exploitation — such as local price manipulation or alpha theft by frontrunners — obviously worsen trader experience. But other types of exploitation facilitate essential functions such as arbitrage, i.e. communicating prices between different trading venues, aiding their convergence on a global equilibrium. A bot that carries out such a function by backrunning

a trade doesn’t even directly impact the execution of the trade — essentially, this bot performs a janitorial service, “cleaning up” the trading venue after a user without getting in their way.

An access road for backrunners

Early attempts to formalise, and perhaps legitimise, access to the dark forest provided tools to carry out arbitrary types of exploitation on pending public

transactions, as well as tools to make unconfirmed transactions private

and hence protected from users of the former. To stretch the forest metaphor, generalist bundling tools like MEV-Boost provides single-lane access road to

the forest, and any transaction that might be lost in there; meanwhile pre-confirmation privacy services like Flashbots Protect provide a safe single-lane route through

the forest.

This left untapped a demand for more structured routes that provide a preferred lane for “good” robots to perform restricted maintenance functions on passing transactions while protecting the latter from atomic frontrunning. Apart from clearing the network from congestion caused by unorganised competition, such infrastructure can even provide benefits directly to the traders being backrun: since the ability to reliably backrun an order is valuable to the backrunner, the trader can effectively demand a portion of the proceeds in return for preferred access to his order.

There are services based on this principle already in production: COW DAO’s MEV-Blocker and Flashbots’ MEV-Share.

Does privacy actually prevent frontrunning?

It is probably commonly assumed that pre-confirmation privacy prevents frontrunning[1]

Certainly, it makes it more difficult: it transforms the problem from “fish in a barrel” to “fish in a pond.” If you want to catch fish in a barrel, you can just look in the barrel, reach in and grab one directly. Ethereum’s public mempool is a barrel.

In a pond, you cannot necessarily see the fish. However, under the right conditions, you may be reasonably sure they are there, and you can still arrange to have a good chance of catching one. The more you know about the fish population, the better your chances. You just have to spend a bit of time dangling your lure.

Arbitrage

To put some meat on this metaphor, let’s get into the main example. Suppose a liquidity trader

Lorenzo wishes to buy a certain amount of some asset available in some CFMM pool \mathcal{M}

. Lorenzo is not particularly opinionated about what the price of that asset should be in absolute terms, but naturally he wants to get as close as possible to the best price available in the target timeframe.

Lorenzo sends a market order along the through lane of our 2-lane route. When it lands on the CFMM, it nudges the pool reserves, creating a local price imbalance. If the market does not consider Lorenzo’s trade to be informative about some future price trend, arbitrage bot operators may profit by correcting this imbalance, buying or selling all liquidity available on \mathcal{M}

at a price better than that available on other markets.

Let’s say some arbitrageur Aya wishes to take this opportunity. She will use the maintenance access road to reliably backrun Lorenzo’s trade, submitting an arbitrage order

that commits to buy/sell all liquidity available on \mathcal{M}

for less/more than a target price p

, which she is free to choose. Aya does not care about how much she trades; she only cares about maximising profit.

If Aya is only looking at this single opportunity, her best bet is to set a target price of p_\mathcal{O}

, the best price she can get on some other highly liquid trading venue \mathcal{O}

. She can then net a pure profit by selling on \mathcal{O}

everything she bought on \mathcal{M}

at p_\mathcal{O}

. This is good news for the next trader to use the pool after Aya: he finds the pool at a global equilibrium, improving predictability and hence his ability to price things!

On the other hand, if Aya thinks she has some chance to backrun both Lorenzo and

whoever uses the through route immediately after him — call him Paul — she can try to capitalise on that chance by using the backrun slot after Lorenzo’s trade to blind frontrun

the unlucky Paul, manipulating the price to an unfavourable value, afterwards extracting a supernormal profit with the subsequent backrun. It is not hard to convince oneself that this strategy is higher in expected payoff for Aya if the probability of landing both backruns is high and if the payoff from a successful sandwich is large compared to that of landing two “myopic” arbitrages.

If Aya knows a little about the population of pending trades on \mathcal{M}

, she can make a more informed decision about how to carry out this speculative sandwich. For example, if she knows that all the market orders in the pool are in the same direction, she knows in which direction she should bias the price in order to sandwich. This could happen in practice for traders on MEV-Blocker, which does not hide the transaction contents from the searcher, or MEV-Share if traders choose not to hide transaction direction.

How bad can manipulation get?

In a paper, I devised a game-theoretic model for an idealised trading venue with a two-lane queue for market orders and arbitrage orders. Orders from these queues are interleaved before being executed in what I’m calling a laminated batch

, with payoffs computed accordingly.[2]

The model supports any set of assumptions about arbitrageurs’ knowledge of order contents, ordering, and backrun position allocation. For given order flow and oracle price p_\mathcal{O}

, what prices will a rational, risk-neutral arbitrageur quote?

When I started this, I was expecting to find that for certain liquidity structures and sufficiently small success probability, this speculative price manipulation is not profitable: a small arbitrageur is better off communicating whatever they think the “real” price is. This turned out to be wrong

In fact, the game theory of a single batch is rather trivial: each arbitrageur has a dominant strategy p^*(b,r)

which is a continuous function of his “market power” b>0

— briefly, his chance of landing a sandwich given he lands the backrun slice — and his beliefs r

about the order flow he is trying to sandwich. He doesn’t need to take into account the strategies of his competitors when he chooses his target price.[4]

To make a more precise statement, we need to make some design choices about the information made available to arbitrageurs and the actions they are allowed to take. A design which approaches the model of the existing solutions MEV-Share and MEV-Blocker is to posit a labelling

of the pending trades and allow arbitrageurs to specify a target price for each label. The arbitrageurs need not know the contents or execution order of the trades associated to the labels, nor which ones they will ultimately be allocated to backrun.

It turns out that in nice cases, we don’t actually need these labels to analyse the backrunning game, as the arbitrageur’s best strategy is to set a single price for the whole batch:

Theorem.

If an arbitrageur has no information about the execution order of trades, his dominant strategy is to set the target price to the same value for all labels.

The basic result is that a dominant strategy exists and is close to the oracle price for small order flow and arbitrageur market power. In particular, a rational arbitrageur sets prices independently of the strategies of his competitors.

Theorem.

The dominant strategy p^*(b,r)

is defined for small b

or r

, and converges to the price p_\mathcal{O}

on external markets as b\rightarrow0

or r\rightarrow 0

.[5]

In other words, when one of the following two conditions are satisfied:

- Order flow is identically zero;
- The probability of landing a sandwich is zero;

then the dominant strategy is to quote the oracle price p^*=p_\mathcal{O}

, and small perturbations of these conditions only lead to small changes in p^*

. OK, but how small are we talking?

For simplicity, we assume that a given arbitrageur has the same chance w>0

to be allocated each backrun slot, and that these allocations are independent (i.e. they areBernoulli trials). Then w

stands as a proxy for the market power of the arbitrageur — if you like, imagine that slots are allocated to arbitrageurs by a stake-weighted lottery, and w

is the proportion of stake held. We also assume that the pending orders r_1,\ldots,r_k

are identically distributed \sim r

and that the price impact of a trade of size and direction r

is well approximated by an exponential, i.e.

\phi(x+r) \approx \phi(x)e^{-\lambda r}

for some \lambda > 0

, where \phi

is the marginal price function of the pool and x

is the equilibrium balance of the risky asset[6]

Such approximations hold up pretty well in practice.

Theorem.

Under the above assumptions, the price quoted by a rational arbitrageur is approximated by a “zeta function”

p^*(w,r) \approx Z_{\phi,r}(w) := \frac{1-w}{1-M_r(\lambda)w}

where M_r

is the moment generating function of the order flow distribution r

.

How could such a formula be used in practice? The quantity \lambda

is an easily computed invariant of the underlying CFMM. Given a tractable model for the order flow r

, which can easily be fit to historical data, one can compute M_r(\lambda)

.

The tricky part is bounding w

. If

one can bound w

to some moderate value, say 10\%

, then one can use a power series expansion to get an estimate for p^*

which might be quite reasonable in practice — say, small enough that general market frictions make the error indistinguishable from noise.

However, while it may well be possible to get some idea of the market share of large MEV actors, particularly if these represent known real-world entities such as trading firms, it is generally not possible to rule out behind-the-scenes cooperation that would allow multiple actors to act as one large manipulator, as is well-known to occur in real life scenarios

Conclusion

Clearly, the two-lane combination of pre-confirmation privacy and atomic backruns provide a UX improvement over the public mempool status quo while still allowing robots to perform valuable janitorial services. The benefits are most pronounced when the searcher market is highly decentralised so that the probability of any individual agent landing a sandwich is small, net order flow is small, and searchers have little information about the execution order or contents of individual pending market orders.

Exchange or transaction queueing infrastructure designers can use these results to set parameters and quote (probabilistic) bounds on trade execution quality to their customers. Bounds would be updated live based on up to date short term order flow forecasts and estimates of arbitrageur market power. That said, to ensure realism of the model, more work needs to be done to expand its scope to include the incentives associated to bidding for backrun positioning and dynamic strategies that may include smart contract commitments.

The features of bipartite queues alone do not completely prevent the exploitation of liquidity traders by sophisticated market participants, and the possibility of profiting from price manipulation presents an incentive to collude. Yet, the problem is probably no worse than cartelisation and oligopoly found elsewhere in finance and industry. Somehow, we still keep trying to build competitive markets and decentralised systems, and sometimes we succeed. Let’s keep building.

1. The False Narrative of MEV Protection: How Private Transactions Can Result in a Poorer Settlement Than Sending Publicly ↩
2. We assume that no changes to liquidity structure occur within the batch.↩
3. Assuming zero transaction fees. ↩
4. This breaks down when considering strategies with a horizon of multiple batches.↩
5. Since r

is itself a random variable, we should understand the limit r\rightarrow 0

in terms of convergence in probability.↩

1. A natural candidate for this approximation is obtained by linearising \log\phi

, whence \lambda

is given by the logarithmic derivative of \phi

. If \phi

is the price function of a Balancer-style weighted CPMM, then \lambda\in(0,1)

is the pool share of the numéraire asset. In practical situations of reasonable liquidity depth, this approximation is very good. ↩