Hey all, like these two other excellent threads which I recommend reading -

MEV capturing AMM (McAMM) [

Applications

](/c/applications/18)

MEV capturing AMM (McAMM): A prevailing thought is that the power of transaction ordering is mostly in the hands of block-builders in the current MEV-Boost and PBS specifications. In this write-up, ideas for new AMM designs are presented that would shift the transaction ordering power, at least partly, to AMM designers and liquidity providers. These constructions would allow AMMs to capture part of the MEV that is currently only harvested by block-builders and proposers.

High-level idea: New …

MEV Minimizing AMM (MinMEV AMM) [

Applications

](/c/applications/18)

Towards minimal-MEV-AMM via Direct Elicitation I have been thinking about AMM designs that directly elicit initial prices to offers traders, as a way of reducing the amount of MEV that can be extracted from LPs. Attention conservation notice: very raw work in progress. Passive liquidity providers give away for free a straddle to those who first trade with them. Directly ellicit the price vector that removes this free stradle seems possiblein principle: it is the final price vector in the bloc…

I'm proposing a new MEV Capturing AMM model that, while not quite as efficient at capturing MEV(reducing LVR), is significantly simpler than the versions discussed above. Basic motivation is the same as the other threads, AMM's currently leak a lot of MEV/suffer from Loss-Versus-Rebalancing. I'm proposing using a dynamic fee AMM to address this rather than first right auctions, hence the name Dynamic MEV Capturing AMM(DMcAMM).

High-level idea:

Liquidity pools use a dynamic fee that decays over blocks and resets every time someone swaps through the LP. This mechanism allows LPs to capture backrunning profits, is likely a more profitable fee mechanism than current static fee systems, and has the potential benefit of making sandwich attacks less feasible.

Implementation:

The implementation would follow standard AMM implementations. However, instead of storing the swap fee in state, it would calculate the swap fee on the fly based on how many blocks have passed since the last swap through that pool. The pool would also have to record the block number every time a swap is performed. This would result in slightly more expensive swaps for the additional logic and write per swap.

Note:

This implementation has the fairly significant issue of block number not being an especially granular or accurate measurement of the volume of market activity since the last swap. A potential fix would be adding an additional factor to the fee scaling, such as the number of swaps performed globally by the protocol since the last time the pool in question carried out a swap. However, granularity might not be an issue in low-blocktime environments like rollups, so maybe this is unnecessary.

Discussion:

The main goal for this DMcAMM construction was to capture the majority of important MEV without adding significant complexity to the AMM. To this end, we can break AMM MEV into three types and analyze if/how this construction captures it:

1. Sandwiching -

The DMcAMM construction makes all but the largest sandwich attacks unprofitable because the frontrun swap raises the LP fee to its highest level for both the user and backrun swaps. Thus it doesn't effectively capture sandwich MEV. That said, I don't think it's important to focus on capturing sandwich MEV as tools like COWswap and private builder transaction flow will make sandwiches significantly less common. Verdict: Unnecessary to Capture

1. Internal-Orderflow Arbitrage -

Here I'm referring to arbitrage from a user's swap creating a backrun opportunity. Fees that decay over blocks allow the LP to take a significant cut of the MEV from backruns since the initial swap resets the LP fee to its highest level, effectively initiating a dutch auction between backrunners. Large backrun opportunities may not have large fee sharing since their profits can far exceed the LP's pre-decay fee. However, I'm not convinced we should be concerned with large backrun opportunities as they are uncommon and will become even less common as execution and UX improve. Verdict: Captured

1. Stale-Price Arbitrage -

Here I refer to arbitrage created by the AMM offering an out-of-date swap quote that creates an arbitrage opportunity with a different market. Decaying fees capture this MEV fairly well in high volatility conditions since quotes frequently become stale, meaning frequent arbitrage swaps and consistently high fees. However, they do a very poor job internalizing this MEV in low volatility conditions as quotes will become stale slowly, meaning fees have time to decay quite low. I don't think failure to capture in low-volatility conditions is a significant issue. First decaying fees mean the pool will likely capture organic flow before toxic flow, given that volatility is low. Furthermore, the vast majority of stale-price arbitrage occurs in high-volatility conditions. Verdict: Captured when most important

In conclusion, if parameters can be set correctly, a DMcAMM does a sufficiently good job capturing MEV (or reducing LVR, whichever term you prefer). That being said, there is one other benefit worth discussing. There has recently been a lot of discussion around dynamic fee AMMs, mainly thanks to research like Crocswaps. The main finding of that research is that high fees are better for LPs in high-volatility conditions. The DMcAMM construction raises fees in high-vol conditions since it resets the fee to its highest point every swap, conforming to research recommendations.

There are also a few downsides of this construction worth discussing. Primarily, AMM UX would be worse due to the constantly fluctuating fee. I don't see this as a breaking issue as users will eventually swap mostly through routers which could hopefully handle the fee fluctuations behind the scenes, but it's still worth considering. Secondly, as discussed previously, in slow-blocktime environments setting a starting fee and sane decay factor is likely difficult, if not impossible. Finally, killing sandwich attack viability could be bad as they subsidize low-fee LPs on-chain and may improve pricing.

Anyway, I hope people found this model interesting. Feel free to shoot me a DM or just respond here if you wanna discuss further.

TLDR

Solution:

Make AMM fees dynamic and have them decay over blocks, resetting them every swap through their pool.

Pros:

- Simple
- Captures most relevant MEV
- Conforms to recent dynamic fee research

Cons:

- Parameters are hard to set depending on the environment
- Could make UX worse
- Reducing sandwich attacks is maybe bad

Other Related Work:

Loss-Versus-Rebalancing: https://anthonyleezhang.github.io/pdfs/lvr.pdf

Dynamic Fee Research: Designing a Dynamic Fee Policy that Outperforms All Uniswap ETH/USDC Pools | by CrocSwap | Sep, 2022 | Medium