Pooling Liquidity Pools in AMMs

Marcelo Bagnulo, Angel Hernando-Veciana, and Efthymios Smyrniotis

Universidad Carlos III de Madrid

Abstract. Market fragmentation across multiple AMMs creates inefficiencies such as costly arbitrage, high slippage, and vulnerability to sandwich attacks. These inefficiencies increase trading costs, reduce liquidity provider profits, and degrade overall market efficiency. We propose a modification of the CPMM pricing mechanism, the *Global Market Maker* (GMM), that incorporates liquidity information from all AMMs to mitigate these inefficiencies. Our theoretical and numerical analyses demonstrate that the GMM increases profits for AMMs and traders by eliminating arbitrage opportunities. It also lowers the profitability of sandwich attacks and minimizes impermanent losses.

Keywords: Decentralized Exchanges (DEXs) · Automated Market Makers (AMMs) · Constant Product Market Maker (CPMM) · Arbitrage · Slippage · Sandwich Attacks · Impermanent Loss

1 Introduction

As cryptocurrentcies' popularity grows, so it does the need to exchange them for other cryptocurrencies and for fiat currencies. Crypto exchanges have emerged to satisfy these needs. A crypto exchange is a platform that allows users to buy, sell, and trade cryptocurrencies. Crypto exchanges can be centralized or decentralized. Centralized EXchanges (CEXs) are owned and operated by a single entity, which holds users' funds and executes trades. Decentralized EXchanges (DEXs) do not rely on a central authority to hold users' funds or execute trades. Instead, they use smart contracts running over a public blockchain such as Ethereum, to automate the trading process. DEXs have become increasingly popular reaching a daily trading volume of billions of dollars [1].

There are two main types of DEXs, namely, order book DEXs and Automated Market Makers (AMMs). Order book DEXs were introduced first but nowadays AMMs are the most common type of DEX. Order book DEXs are similar to traditional order book exchanges. Users place orders to buy or sell assets, and these orders are matched with each other to execute trades. AMMs work by creating liquidity pools, which are essentially large baskets of assets that are held by the exchange. When a user wants to trade an asset, they do not actually trade with another user. Instead, they trade with the liquidity pool. The AMM will then use a predetermined mathematical formula to determine the price of the trade. One of the most widely used AMMs is the Constant Product MM (CPMM). CPMMs use a constant product formula, that ensures that the product

of the reserves of two assets in the liquidity pool remains constant. The CPMM's formula inputs the volume of the trade and the available liquidity and outputs the price of the trade.

A key feature of the CPMM algorithm is that the price it computes reflects the relative scarcity of an asset in the AMM's liquidity pool. This effectively translates traders' demand into prices: when market sentiment about an asset's price rises, traders buy the asset from the AMM, reducing its availability in the liquidity pool until the AMM's price aligns with market sentiment.

Market fragmentation across multiple AMMs presents challenges, the most notable being the reliance on costly arbitrage to equalize prices. Each CPMM independently adjusts swap rates based on its own trading history, discovering asset prices at different paces. If two CPMMs start with the same exchange rate and one executes a trade, their rates will diverge, with larger trades causing greater price differences. These discrepancies create arbitrage opportunities, systematically exploited by arbitrageurs to restore price consistency. A recent study [15] estimates that arbitrage profits on the Ethereum blockchain alone amount to 170 million per year, leading to higher costs for traders and reduced profits for liquidity providers, and thus reducing the efficiency of the blockchain ecosystem.

Another consequence of market fragmentation is increased slippage, i.e. the price's sensitivity to order size. This effect is particularly pronounced in AMMs with small liquidity pools, where large orders significantly alter asset scarcity, causing sharp price impacts in the CPMM. Clearly, aggregating all liquidity pools into a single AMM would substantially reduce slippage if the CPMM algorithm remains in use.

Standard microeconomic analysis suggests that slippage reduces market efficiency as it deviates from the price-taking assumption; see [2]. In practice, however, a more pressing issue is that higher slippage increases the profitability of sandwich attacks. In this front-running strategy, an attacker places a buy order before and a sell order after a victim's trade, manipulating the price to profit from the slippage they induce. As a result, the victim receives a worse price, while the attacker exploits the price difference for profit.

In this paper, we explore how to address these inefficiencies by computing prices in the spirit of the CPMM but based on the relative scarcity of assets in the aggregate liquidity pools of all AMMs rather than in individual AMM pools. This approach allows prices in a fragmented AMM ecosystem to behave as if they were set by a single AMM.

Our first observation is the failure of the naive approach of applying the CPMM formula using aggregate liquidity pools instead of individual ones, what we call the *naive Global Market Maker* (nGMM). This algorithm can be easily manipulated by strategically rearranging transactions, potentially leading to the depletion of the AMM reserves.

Next, we characterize algorithms that fall between the standard CPMM and the nGMM but are not manipulable as described in the previous paragraph.

Among these, we define the *Global Market Maker (GMM)* as the algorithm closest to the nGMM.

In the rest of the paper, we study the properties of the GMM and compare it to the CPMM. We show that the GMM eliminates all arbitrage opportunities and that the profits arbitrageurs would earn when all AMMs use the CPMM algorithm instead are shared between the AMMs and the traders.

We also investigate, both theoretically and numerically, how much the GMM reduces the profitability of sandwich attacks compared to the CPMM. In all simulated examples using real data, we observe reductions of over 50%.

Finally, we analyze the performance of the GMM with respect to impermanent loss. This is defined, see [11], as the loss incurred by a liquidity provider when depositing assets into an AMM, if the relative prices of those assets change. Since the GMM incorporates liquidity information from other AMMs, the price it offers adjusts before reaching the AMM which minimises impermanent losses.

The rest of the paper is structured as follows.

2 Literature Review

Several alternative mechanisms have been proposed to mitigate MEV and the costs of arbitrage. [6] introduce batch trading, a system where orders are executed in batches rather than continuously, thereby reducing arbitrage and sandwich attacks. This mechanism leverages arbitrageurs' competition to eliminate price manipulation but relies on oracle pricing. [20] study verifiable sequencing rules to prevent MEV and find that while some non-zero miner profits remain, their proposed sequencing rule ensures that user transactions remain unaffected. [6] develop a game-theoretic model to differentiate front-running from legitimate trades. Their proposed protocol reduces front-running risk but requires additional message exchanges, which could impact efficiency. [21] proposes a modification of AMM algorithms that incorporate optimal on-chain swap routing and arbitrage.

The literature also provides a detailed analysis of the consequences of arbitrage between AMMs. [17] identifies arbitrage as a rebalancing cost for LPs in a theoretical model of liquidity provision. [16] studies the effect of fees on arbitrage. [14] explores cross-chain arbitrage in decentralized exchanges, providing an empirical analysis beyond the Ethereum ecosystem. [12] examine arbitrage opportunities in Ethereum, identifying $30 million in cross-exchange and triangular arbitrage profits across 63,168 trading pairs between July 2020 and February 2022.

The profitability of sandwich attacks and more generally the issue of Maximal Extractable Value (MEV) has been widely documented. [9] introduce the concept of MEV.[5] provide a game-theoretic analysis of sandwich attacks. [18] estimate that total extractable value over a 32-month period reached $541 million, highlighting the scale of MEV extraction. [7] develop methodologies to identify sandwich attacks, finding that $675 million was extracted before September 2022 and noting that high-volatility tokens are primary MEV targets. [10] provide a

comprehensive literature review on MEV, categorizing extraction strategies and countermeasures. [19] analyze MEV extraction in private pools, quantifying the role of flashbots and other sophisticated arbitrage techniques.

More generally, our work is related to the literature on Automated Market Makers (AMMs). [13] provide a general characterization of AMMs, comparing them to Centralized Exchanges (CEXs). [3] analyze strategic liquidity provision in AMMs. [8] analyse the conditions under which CPMM pricing is an optimal liquidity provision mechanism.

3 The Analysis of Global Market Makers

In this section, we develop the theoretical foundations of the design of algorithms for AMMs that make use of global information available in the blockchain. We first define the algorithms we plan to improve, second, we define the range of improvement and the set of constrains we are considering, and finally we provide the theoretical analysis and conclusions.

We focus our attention on the most popular family of algorithms used by automated market makers (AMMs): the constant product market makers (CPMMs) introduced by Vitalik Buterin in a blog post in 2016 [4] to provide a more efficient and decentralized alternative to the provision of liquidity than traditional market makers. Hayden Adams provided the first implementation of the CPMM in his decentralized exchange, Uniswap, in 2017, and Uniswap quickly became the most popular decentralized exchange on Ethereum, and CPMMs the most popular type of AMM.

The Constant Product Market Maker (CPMM) is an algorithm that manages a pool of reserves consisting of two assets, known as the liquidity pool of the AMM. A trader who wishes to swap one asset for another deposits a certain amount of the first asset into the AMM. The CPMM algorithm then calculates the amount of the second asset to be returned to the trader, ensuring that the product of the reserves of both assets in the liquidity pool remains constant. Formally, let x_i and y_i be the reserves of assets X and Y in the algorithm's liquidity pool. If a trader sends an amount $\Delta x_i > 0$ of asset[1] X into the algorithm, the CPMM returns an amount $\Delta y_i > 0$ of asset Y such that:

$$x_i \cdot y_i = (x_i + \Delta x_i) \cdot (y_i - \Delta y_i). \tag{1}$$

The solution to this equation provides the following characterization of the CPMM:

$$\Delta y_{CPMM}(\Delta x_i; x_i, y_i) = \frac{y_i}{x_i + \Delta x_i} \Delta x_i = \frac{r_i}{1 + \frac{\Delta x_i}{x_i}}, \tag{2}$$

where $r_i \equiv \frac{y_i}{x_i}$ is the ratio of reserves of the AMM. The interpretation of the last equation is that the terms of trade $\frac{\Delta y}{\Delta x}$ of the CPMM are given by a measure of

[1] Since X and Y are arbitrary assets, it is sufficient to describe orders in which the trader sends asset X. The case in which the trader sends asset Y can be handled by relabeling the assets.

the relative scarcity of the assets in the liquidity pool, adjusted by a measure of the order's size relative to the liquidity pool of the AMM. The first term represents the marginal price, as it corresponds to the limit of the terms of trade when Δx_i tends to zero. The second term captures slippage, which quantifies the rate at which the terms of trade deteriorate as the order size increases relative to the liquidity pool.

In reality, several AMMs may be using the CPMM what creates profitable arbitrage opportunities as we illustrate in the next example.

Toy example - Part 1: Arbitrage between CPMMs. Consider two CPMMs, CPMM1 and CPMM2, each one with a liquidity pool containing 100 ETH and 400,000 UST. The price is the same for both CPMMs and equal to 4,000 UST/ETH. All the computations in this example are done using formula 1.
Suppose that a trade for buying 10 ETH arrives to CPMM1. This means that after the trade, CPMM1's pool will contain 90 ETH and 444,444 UST. The price in CPMM1 is now 4,938 UST/ETH. The trader has paid 44,444 USTs. An arbitrage opportunity has emerged.
The arbitrageur will then buy 5 ETH from CPMM2 and sell 5 ETH to CPMM1. After the arbitrage, the reserves of both CPMMs will be equal to 95 ETH plus 421,052 UST. The arbitrageur has made a profit of 2,339 UST.
Suppose at this point that a trader perform the reverse operation, meaning that it sells 10 ETH to CPMM1 After the trade CPMM1 pool will contain 105 ETH and 380,952 UST and the trader has obtained 40,100 UST. Again, an arbitrage operation has emerged which is seized by an arbitrageur buying 5 ETH from CPMM1 and selling them to CPMM1, making a profit of 2,005 UST. After these two arbitrage operations, the reserves of both AMMs wil be restored to 100 ETH and 400,000 UST each.
Overall, in these 6 operations, the CPMMs retained their initial reserves intact, the arbitrageur has made a profit of 4,344 UST which matches to the net loss of the trader(s)-

The CPMM may be subjecto to sandwich attacks. This is a manipulative attack that combines frontrunning and backrunning to exploit a target transaction. The attacker places a trade before the target transaction (frontrunning) to move the price in their favor and then executes another trade after (backrunning) to profit from the price movement caused by the victim's trade. In CPMMs, this involves inserting trades around a large swap to capitalize on slippage, forcing the victim to execute their trade at a worse price while the attacker profits from the price difference. We illustrate sandwich attacks in the next example.

Toy example - Part 2: MEV extraction in CPMM. Consider a CPMM with a liquidity pool containing 100 ETH and 400,00 UST. The marginal price is then 4,000 UST/ETH.
Suppose that a trader submits a transaction for buying 10 ETH. The MEV-extractor performs a sandwich attack by placing a transaction to buy 15 before the victim's transaction is executed and another transaction, in this case selling 15 ETH after the victim's transaction was executed. This would works as follows:
Frontrunning: MEV-extractor transaction 1 that buys 15 ETH. The resulting reserves after this initial transaction are 85 ETH and 470,588 UST. The marginal price price has increased to 5,536 UST/ETH. The MEV-extractor has spent 70,588 UST.
Victim's transaction: THe victim buy 10 ETH. The resulting reserves are then 75 ETH and 533,333 UST. The victim has payed 62,745 UST i.e. a surplus of 18,301 UST compared to what she would have payed without the frontrunning.
Backrunning: MEV-extractor transaction 2 that sells 15 ETH. The CPMM's reserves after the trade are 90 ETH and 444,444 UST. The MEV-extractor obtains 88,858 UST, making a total profit of 18,301 UST (which is exactly what the victim's lost, of course).

Impermanent loss is the temporary loss liquidity providers (LPs) may experience when supplying liquidity to Automated Market Makers (AMMs), particularly Constant Product Market Makers (CPMMs). It arises due to the automatic rebalancing of token holdings in the liquidity pool as the relative price of tokens changes. If one token's price increases relative to the other, LPs end up holding more of the cheaper token and less of the more expensive token than they initially deposited. The loss is termed impermanent because it may be reversed if the price ratio returns to its original state. We illustrate it in the next example.

Toy example - Part 3: Impermanent loss in CPMM. Consider two CPMMs with liquidity pools containing 100 ETH and 400,00 UST each. The marginal price is then 4,000 UST/ETH.
Suppose that one or more traders submit transactions for buying 30 ETH on each CPMM, moving the marginal price to 8,163 UST/ETH in both CPMMs. The resulting reserves for each CPMM afterwards are 70 ETH and 571,429 UST. We can compute impermanent loss IL) as the difference between the value of the CPMM's original reserves valued at the current price and the current reserves valued at the current price i.e.
$InitialReserves = 2 * (100ETH * 8,163UST/ETH + 400,000UST)$
$FinalReserves = 2 * (70ETH * 8,163UST/ETH + 571,429UST)$
$IL = InitialReserves - FinalReserves = 146,922UST$

Our objective is to generalize the CPMM to incorporate additional information available on the blockchain, particularly the liquidity pools of other AMMs that trade the same two assets. To formalize this, we assume a set I of AMMs,

each with liquidity pools described by the vector $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}_+^{2I}$, where

$$x \equiv \sum_{j \in I} x_j, \quad \text{and} \quad x_{-i} \equiv \sum_{j \in I \setminus \{i\}} x_j,$$

with analogous definitions for y and y_{-i}. Let also $r = \frac{y}{x}$ be the global ratio of reserves.

We focus on modifications of the CPMM that determine the exchange rate based not only on the individual liquidity pool but also on the entire vector of liquidity pools across all AMMs. We refer to such algorithms as *global*. The most natural global extension of the CPMM is as follows.

Definition 1. *Given a vector of liquidity pools (x, y), the naive Global Market Maker (nGMM) algorithm swaps $\Delta x_i > 0$ of X for an amount of Y:*

$$\Delta y_{nGMM}(\Delta x_i; x, y) = \frac{y}{x + \Delta x_i} \Delta x_i = \frac{r}{1 + \frac{\Delta x_i}{x}} \Delta x_i, \tag{3}$$

if less than y_i, and $\Delta y_{nGMM}(\Delta x_i; x, y) = y_i$, otherwise.

This algorithm determines the terms of trade as if executing a hypothetical CPMM with reserves equal to the sum of individual reserves. Consequently, the terms of trade reflect both global scarcity and global slippage. Notably, if all AMMs adopt this algorithm, arbitrage opportunities are eliminated, as illustrated in the following example.

Toy example - Part 4: nGMM Consider two market makers GMM1 and GMM2 using nGMM, where the initial distribution of liquidity pools is $(\boldsymbol{x}^0, \boldsymbol{y}^0) = (100ETH, 100ETH, 400000UST, 400000UST)$. Suppose that GMM1 receives a buy-order of 10 ETH. The resulting reserves of UST of GMM1 can be computed as $\frac{(100+100)*(400{,}000+400{,}00)}{(90+100)} - 400,000$ which is roughly 442,105 UST. This means that trader payed 42,105 UST to obtain the 10 ETH. This is less than what the trader has to pay in the CPMM case (see Toy Example 1). Actually, the reduction in what the trader had to pay is exactly the arbitrageur profit computed in Toy Example 1.

However, the nGMM has two significant drawbacks.

The first drawback is straightforward: since the algorithm determines prices based on global scarcity, it may completely deplete the reserves of one asset, forcing the AMM to exit the market. This issue does not arise in the CPMM by design. Withdrawing an amount Δy of Y arbitrarily close to y_i requires supplying the AMM with an unbounded amount of X, meaning the per-unit price of Y diverges to infinity.

The second drawback is more subtle: the nGMM is vulnerable to exploitation through strategically rearranged transactions. Specifically, the algorithm may sell at a low price when an asset is relatively abundant across other AMMs, only to buy it back at a higher price when it becomes scarce in subsequent trades. As a result, the AMM may suffer a net decrease in its holdings of both assets.

We illustrate these two vulnerabilities with an example.

Toy example - Part 5: Exhausting and Exploiting nGMM
Consider two market makers nGMM1 and nGMM2 using nGMM, where the initial distribution of liquidity pools is $(\boldsymbol{x}^0, \boldsymbol{y}^0) = (100ETH, 100ETH, 400000UST, 400000UST)$.

- **Exhausting nGMM**. Suppose that a nGMM1 receives a transaction buying 100 ETH. nGMM1's reserves in UST can be computed as $\frac{(100+100)*(400,000+400,00)}{(0+100)} - 400,000$ which is roughly 1,2M UST. nGMM1's reserves in ETH are zero. This shows how with a finite amount of UST, nGMM1's reserves in ETH can be exhausted.
- **Exploiting nGMM**. We start again with the initial distribution of reserves of 100 ETH and 400,000 UST in both nGMM1 and nGMM2. Consider the effect of the following 3 transactions:
  Exploit transaction 1: Trader buys 10 ETH from nGMM1. The resulting reserves of nGMM1 are 90 ETH and 442,105 UST. The trader has paid 42,105 UST for the 10 ETH. nGMM2 still has 100 ETH and 400,000 UST.
  Exploit transaction 2: Trader buys 10 ETH from nGMM2. The resulting reserves of nGMM2 are then 90 ETH and 446,783 UST. The trader paid 46,784 UST. nGMM2 has more reserves than nGMM1 (same amount of ETH but more UST)
  Exploit transaction 3: Trader sells 10 ETH to nGMM1. The reserves for nGMM1 woult then be 100 ETH and 395,321 UST, which is strictly less than the initial reserves of nGMM1 (same amount of ETH and less UST). nGMM has experiences a net loss of reserves. The trader has obtained 46,784 UST.
  Observe that the last two transactions cancel out exactly and that the result is a net transfer of reserves from nGMM1 to nGMM2 i.e. the trader does not obtain any profit from the attack). This means that the nGMM approach generates the incentives for participating market makers to attack each other to extract each other reserves.

These two vulnerabilities render the nGMM unsuitable for practical implementation. Our approach, therefore, is to explore how much we can adjust the CPMM to approximate the nGMM while avoiding these vulnerabilities.

For this purpose, we adopt a highly conservative approach. We restrict our attention to algorithms that, once implemented by an AMM, say i, prevent exhaustion and exploitation for any initial distribution of reserves $(\boldsymbol{x}^0, \boldsymbol{y}^0)$ and any sequence of swaps.

Formally, a sequence of swaps is given by $\{\iota(t), \Delta x_t, \Delta y_t\}_{t=1}^{T}$, where the index t represents the order of transactions, $\iota(t)$ denotes the identity of the AMM handling the swap, and Δx_t and Δy_t represent the quantity of X sent to and the quantity of Y returned by AMM $\iota(t)$, respectively. The case in which the

AMM receives Y and supplies X is handled by allowing Δx_t and Δy_t to take negative values.

We impose only minimal consistency requirements. We say that a sequence of transactions $\{(\iota(t), \Delta x_t, \Delta y_t)\}_{t=1}^{T}$, where $\Delta x_t > 0$, is consistent with the global algorithm of AMM i if, at any t such that $i = \iota(t)$, swapping Δx_t for Δy_t (or vice versa) is compatible with the algorithm of AMM i and the vector of liquidity reserves updated with the preceding swaps $\{(\iota(s), \Delta x_s, \Delta y_s)\}_{s=1}^{t-1}$.[2]

Definition 2. *The global algorithm used by AMM $i \in I$ is non-exhaustible if there exists no distribution of liquidity pools and no sequence of swaps consistent with the algorithm of i that causes AMM i to run out of either asset, i.e., such that $x_i = 0$ or $y_i = 0$.*

Definition 3. *A global algorithm employed by AMM i is exploitable if there exists an initial distribution of liquidity pools (x_i^0, y_i^0) and a sequence of swaps consistent with AMM i's algorithm such that AMM i ends up with a lower liquidity pool, i.e., $(x_i, y_i) \leq (x_i^0, y_i^0)$.*[3]

Finally, we restrict our attention to algorithms whose terms of trade lie between those of the CPMM and the nGMM.

Definition 4. *A global algorithm between the CPMM and the nGMM returns for a swap $\Delta x_i > 0$ of X an amount of asset Y that lies between the minimum and the maximum of $\Delta y_{CPMM}(\Delta x_i; x_1, y_1)$ and $\Delta y_{nGMM}(\Delta x_i; \boldsymbol{x}, \boldsymbol{y})$, for a given vector of reserves $(x, y) \in \mathbb{R}^2$.*

The restriction to algorithms between the CPMM and the nGMM should be interpreted as a minimal departure from existing practices. The CPMM represents the current state of the art, while the nGMM serves as its ideal generalization to mitigate the inefficiencies caused by market fragmentation. Our objective is to approximate the nGMM as closely as possible while deviating as little as possible from the CPMM.

Proposition 1. *A global algorithm in between the CPMM and the nGMM is not exploitable if and only if it satisfies:*

$$\Delta y(\Delta x_i; \boldsymbol{x}, \boldsymbol{y}) \leq \Delta y_{CPMM}(\Delta x_i; x_i, y_i), \tag{4}$$

for all $(x, y) \in \mathbb{R}^2$ and all swaps $\Delta x_i > 0$ of X, and the symmetric condition holds for all swaps of Y.

Proof. By contradiction, suppose a distribution of reserves $(\boldsymbol{x}^0, \boldsymbol{y}^0)$ and a swap $\Delta x_i > 0$ that delivers Δy_i units of Y that violates (4). The definition of CPMM means that the swap Δx_i decreases the product of reserves, this is,

$$x_i^0 y_i^0 > x_i y_i, \tag{5}$$

[2] To simplify notation, we do not explicitly consider liquidity injections. Their inclusion is straightforward and does not affect our characterization.

[3] We denote $(x_i, y_i) \leq (x_i', y_i')$ if $x_i \leq x_i'$ and $y_i \leq y_i'$, with at least one inequality being strict.

for $x_i \equiv x_i^0 + \Delta x_i$ and $y_i \equiv y_i^0 - \Delta y_i$. Now suppose that there is a sequence of swaps submitted to the other AMMs that leads to a new vector of reserves (x_{-i}, y_{-i}) such that:

$$\frac{y_{-i}}{x_{-i}} \leq \frac{y_i}{x_i}, \tag{6}$$

and consider a swap to AMM i that sends Δy_i of Y to AMM i. Then:

$$\begin{aligned}\Delta x_{CPMM}(\Delta y_i; x_i, y_i) &= \frac{\frac{x_i}{y_i}}{1 + \frac{\Delta y_i}{y_i}} \Delta y_i \\ &\leq \frac{\frac{x}{y}}{1 + \frac{\Delta y_i}{y}} \Delta y_i \\ &= \Delta x_{nGMM}(\Delta y_i; x, y),\end{aligned}$$

where the two equalitites follow from (2) and (3), and the inequality from the fact that $x_i < x$ and that (6) implies that $\frac{y_i}{x_i} \geq \frac{y}{x}$. This together with our restriction to global algorithms between the CPMM and the nGMM means that the amount of X that AMM i gives to the trader is at least:

$$\hat{\Delta} x_i \geq \Delta x_{CPMM}(\Delta x_i; x_i, y_i), \tag{7}$$

which by the definition of $\Delta x_{CPMM}(\Delta x_i; x_i, y_i)$ implies that:

$$(x_i - \hat{\Delta} x_i)(y_i + \Delta y) \leq x_i y_i.$$

Since $y_i = y_i^0 - \Delta y_i$, (5) means that $x_i - \hat{\Delta} x_i < x_i^0$ and thus the updated vector of reserves of AMM i satisfies,

$$(x_i - \hat{\Delta} x_i, y_i^0) \leq (x_i^0, y_i^0),$$

as desired.

Thus, global algorithms in between the CPMM and nGMM buy asset X at a price no larger than the CPMM price to avoid exploitation. Intuitively, the CPMM price guaranties that when the AMM buys and immediately sells back the asset the vector of reserves returns to its initial level. Thus, whenever a global algorithm buys one of the assets more expensive than the CPMM it is going to be exploited if it sells it back at CPMM prices or lower. But this will always happen with global algorithms with prices in between the CPMM and nGMM if the asset becomes so abundant among the other AMMs between the initial sale and the sell back that the nGMM price becomes lower than the CPMM price.

Furthermore, since the CPMM is not exhaustible and Proposition 1 says that to avoid exploitation the price at which the algorithm buys must be at most the CPMM price, we can conclude that the algorithms that satisfy Proposition 1 are not exhaustible.

Corollary 1. *A non exploitable global algorithm between the CPMM and the nGMM is also non exhaustible.*

Since CPMM prices put a lower bound to global non exploitable algorithms, the closer one can get to nGMM prices while avoiding exploitation (and thus exhaustability) is to use nGMM prices unless bound by the CPMM.

Definition 5. *The Global Market Maker (GMM) algorithm is defined by:*

$$\Delta y_{GMM}(\Delta x; x, y) = \min\{\Delta y_{CPMM}(\Delta x; x_1, y_1), \Delta y_{nGMM}(\Delta x; x, y)\} \quad (8)$$

To understand the GMM, it is useful to distinguish whether a swap $\Delta x_i > 0$ moves the AMM's reserve ratio away from that of the other AMMs, which we call a *divergent swap*. This occurs when the initial vector of reserves satisfies $\frac{y_i}{x_i} \leq \frac{y_{-i}}{x_{-i}}$. In this case, asset X is relatively less scarce locally than globally, causing the nGMM to pay a higher price Δy_i^{nGMM} for X than the CPMM price Δy_i^{CPMM}. Consequently, the GMM price Δy_i^{GMM} coincides with the CPMM price. This is illustrated in Figure 1.

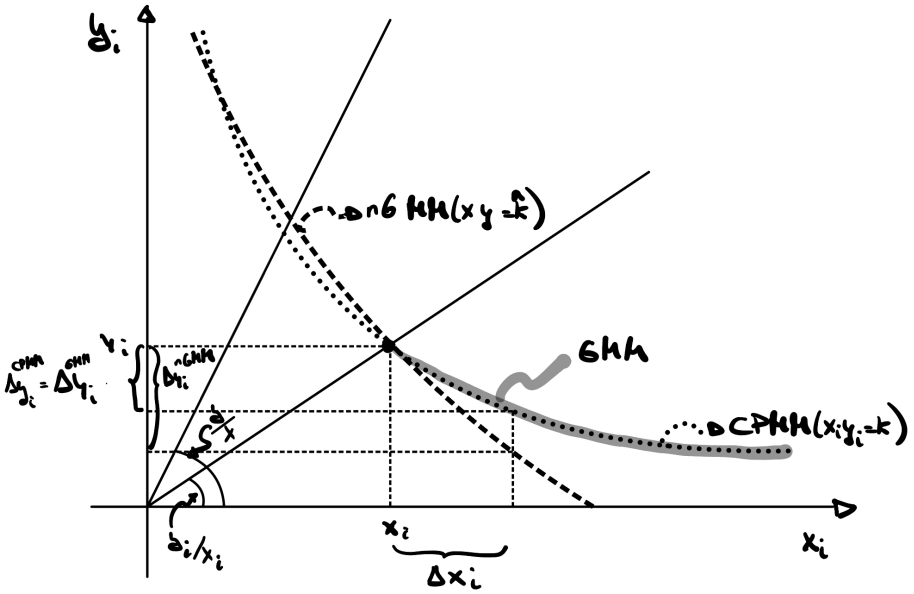

Fig. 1. The graph illustrates a divergent swap where the GMM receives Δx_i units of X and returns Δy_i^{GMM} of Y to the trader, compared to the returns in the CPMM (Δy_i^{CPMM}) and nGMM (Δy_i^{nGMM}) cases. It also shows the possible liquidity reserves of AMM i after a swap, depending on the algorithm used (CPMM, nGMM, GMMM).

A swap $\Delta x_i > 0$ is *non-divergent* when $\frac{y_i}{x_i} > \frac{y_{-i}}{x_{-i}}$. There are two possible cases. The first, which we refer to (with a slight abuse of terminology) as a *convergent swap*, occurs when the nGMM returns less Y than the CPMM, causing the GMM to coincide with the nGMM. This happens when Δx_i is small enough

that the swap moves the reserve ratio toward that of the other AMMs, justifying the name. However, it can also occur if Δx_i is large enough to decrease the reserve ratio below that of the other AMMs but not so large that the nGMM price exceeds the CPMM price. This is illustrated in Figure 2.

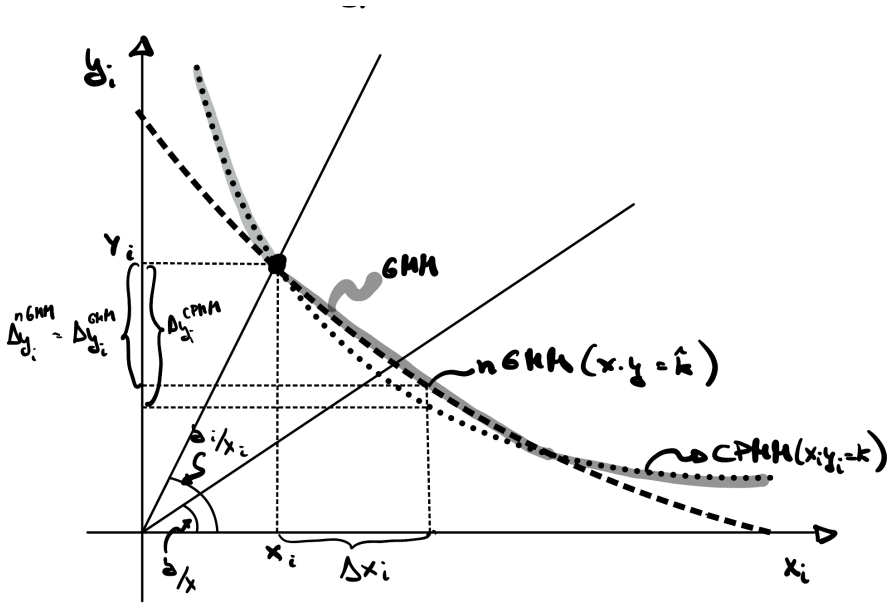

Fig. 2. The graph illustrates a convergent swap where the GMM receives Δx_i units of X and returns Δy_i^{GMM} of Y to the trader, compared to the returns in the CPMM (Δy_i^{CPMM}) and nGMM (Δy_i^{nGMM}) cases. It also shows the possible liquidity reserves of AMM i after a swap, depending on the algorithm used (CPMM, nGMM, GMMM).

The last case, *the overshooting swap*, is when the non divergent swap Δx is so large that the nGMM pays a higher price Δy_i^{nGMM} for X than the CPMM price Δy_i^{CPMM}, causing the GMM price Δy_i^{GMM} to match the CPMM price.[4] We refer to this case as an *overshooting swap* and illustrate it in Figure 3.

Finally, note that the GMM always receives prices that are not worse than the CPMM, strictly better in the case of convergent swaps. This means that the CPMM product of reserves is expected to grow since the CPMM algorithm keeps the product constant. Next corollary formalises this idea.

[4] Formally, the overshooting case occurs at the point where:

$$\frac{y_i}{x_i} \cdot \frac{y_i - \Delta y_{CPMM}(\Delta x_i; x_i, y_i)}{x_i + \Delta x_i} = \frac{y}{x} \cdot \frac{y - \Delta y_{CPMM}(\Delta x_i; x_i, y_i)}{x + \Delta x_i}.$$

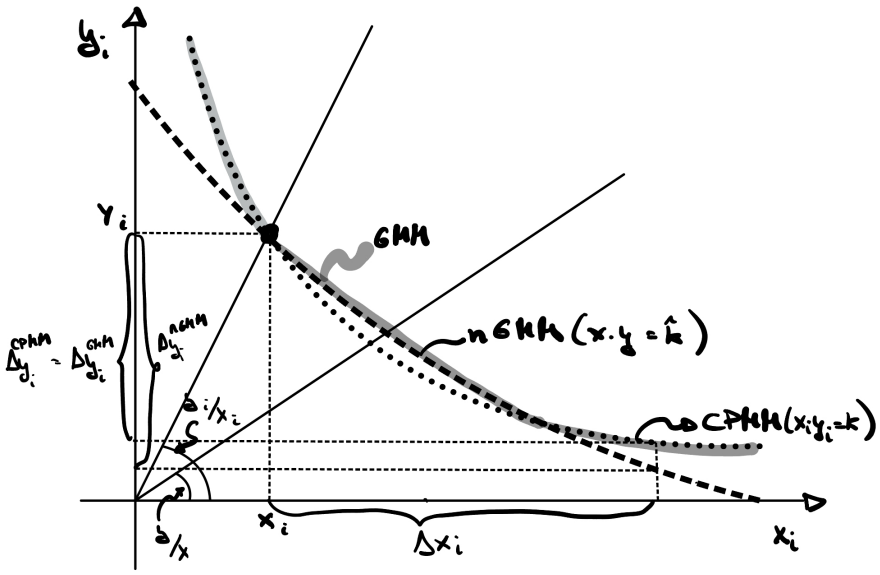

Fig. 3. The graph illustrates an overshooting swap where the GMM receives Δx_i units of X and returns Δy_i^{GMM} of Y to the trader, compared to the returns in the CPMM (Δy_i^{CPMM}) and nGMM (Δy_i^{nGMM}) cases. It also shows the possible liquidity reserves of AMM i after a swap, depending on the algorithm used (CPMM, nGMM, GMMM).

Corollary 2. *The product of reserves of an AMM with the GMM algorithm weakly increases with every swap, strictly if the swap is convergent.*

4 Properties of the GMM

In this section, we show that by exploiting efficiently the aggregation of the information from the liquidity pools of the other AMMs, the GMM improves in the following key dimensions of the design of an AMM: first, it eliminates arbitrage opportunities and splits the efficiency gains between traders and AMMs; second, it reduces slippage and thus the profitability of MEV sandwich trades, and third, it reduces impermanent losses. A second order effect is that the share of arbitrage profits that end in the AMMs increase their liquidity pools and as a consequence further reduce their slippage.

4.1 Arbitrage

We start illustrating arbitrage with an example.

Toy example - Part 6: GMM
In this example, we show that under the same conditions of the example 1, in GMM no arbitrage opportunities appear, and that the profit of the arbitrageurs is split between the AMMs and the traders.
We start with two GMMs (GMM1 and GMM2) that have initial reserves of 100 ETH and 400,000 UST each.
Similarly to Example 1, a trader buys 10 ETH from GMM1. Since both GMMs have the same reserves, GMM1 uses the CPMM formula for this trade (i.e. same as Example 1). After this trade, GMM1's pool contains 90 ETH and 444,444 UST and GMM2's pool is unchanged. The trader has paid 44,444 UST. The resulting prices are as follows:

GMM1 Buy ETH	GMM1 Sell ETH	GMM2 Buy ETH	GMM2 Sell ETH
4,938 UST/ETH	4,444 UST/ETH	4,444 UST/ETH	4,000 UST/ETH

It follows that no arbitrage opportunities emerge, as the most convenient price to sell and buy from the GMMs is the common price.
Suppose now that, similarly to Example 1, the reverse transaction is executed, namely that a trader sells 10 ETH to GMM1. In this case, the GMM determines the prices in the convergent region. This implies that resulting reserves of GMM1 after the trade are 100 ETH and 402,222 UST and that the trader obtained 42,222 UST.
This means that after this trade, the reserves of the GMM have increased in 2,222 UST and that the trader have paid 2,122 UST LESS than in Example 1. In other words, GMM splits almost evenly the profit of the arbitrageur between the trader and the GMM.

The insights of the above example hold true in general.

Proposition 2. *There are no striclty profitable arbitrage opportunities between a set of AMMs that use the GMM.*

Proof. Arbitrage consists of a sequence of swaps initiated by the arbitrageur across different AMMs. Regardless of which AMM executes each swap, the GMM always offers worse terms of trade to the arbitrageur than the nGMM. Moreover, the nGMM preserves the product of aggregate reserves. Consequently, we have:

$$\begin{aligned} xy &\leq (x + \Delta x_1)(y - \Delta y_1) \\ &\leq (x + \Delta x_1 + \Delta x_2)(y - \Delta y_1 - \Delta y_2) \\ &\leq \dots \\ &\leq \left(x + \sum_{t=1}^{T} \Delta x_t\right)\left(y - \sum_{t=1}^{T} \Delta y_t\right), \end{aligned}$$

for any sequence of trades sent by the arbitrageur to the different AMMs, where Δx_i represents the amount of X moved from the arbitrageur to an AMM, and Δy_i represents the amount of Y received by the arbitrageur from the AMM.

This inequality implies that either $\sum_{t=1}^{T} \Delta y_t \leq 0$ or $\sum_{t=1}^{T} \Delta x_t \geq 0$, meaning that the arbitrageur's net position after the swaps,

$$\left(-\sum_{t=1}^{T} \Delta x_t, \sum_{t=1}^{T} \Delta y_t \right),$$

must be either zero or have a strictly negative component. Thus, there are no strictly profitable arbitrage opportunities.

Thus, replacing the CPMM with the GMM could benefit both traders and AMMs by converting strictly profitable arbitrage opportunities into additional profits for them. For any sequence of swaps submitted by traders or arbitrageurs to different AMMs, the total assets transferred to and withdrawn from the AMMs must equal the total variation in liquidity reserves. Arbitrage profits create a wedge between the net flows of traders and AMMs. Eliminating arbitrage removes this wedge, allowing for higher profits for both.

Furthermore, if traders strictly benefit from the switch, their terms of trade must improve, whereas if AMMs strictly benefit, their liquidity pools should grow, reducing slippage and improving terms of trade for future traders. In either case, better terms of trade should lead to increased trading and thus higher potential gains from the switch.

To assess the distribution of gains, we consider a stylized benchmark. First, we assume that all strictly profitable arbitrage opportunities are exploited by arbitrageurs and that traders always submit their trades to the AMM offering the most favorable terms of trade, a strategy we call *optimal routing*. As we shall see, this assumption aligns with empirical evidence.

Second, when all AMMs use the CPMM algorithm, the individual liquidity ratios after maximum arbitrage, $\frac{y_i}{x_i}$, must be equal across AMMs. However, this common ratio may depend on how arbitrageurs extract their profit. For instance, if arbitrageurs structure their swaps to realize profits exclusively in Y, ensuring that the net position of asset X across all arbitrage swaps remains zero, the global liquidity ratio $\frac{\sum_i y_i}{\sum_i x_i}$ will decrease, leading to a corresponding decline in the common individual liquidity ratios $\frac{y_i}{x_i}$.

To abstract from this complication, we assume that arbitrage profit extraction preserves the global liquidity ratio, ensuring that the resulting individual liquidity ratios remain equal to the original global liquidity ratio. We refer to this assumption as *balanced arbitrage*.

Proposition 3. *Consider an initial distribution of liquidity* $(\boldsymbol{x}, \boldsymbol{y})$ *and a swap* $\Delta x > 0$ *of* X *that is optimally routed and compare two scenarios:*

1. *All AMMs use the CPMM, and all strictly profitable arbitrage opportunities in the initial liquidity distribution are exploited by balanced arbitrage before the swap* Δx *is executed.*
2. *All AMMs use the GMM algorithm.*

1. provides the trader strictly better terms of trade than 2. if and only if:

$$\frac{r}{1+\frac{\Delta x}{x}} > \frac{r_i}{1+\frac{\Delta x}{x_i}} < \frac{r}{1+\frac{\Delta x}{\sqrt{\frac{\max\{x_j y_j\}_{j\in I}}{r}}}}, \forall i \in I. \tag{9}$$

Proof. We begin with some remarks that apply to both parts of the proof. In 1., the CPMM mechanism and balanced arbitrage imply that AMM j's reserves (x'_j, y'_j) satisfy the equations:

$$\begin{aligned} x'_j y'_j &= x_j y_j, \\ \frac{x'_j}{y'_j} &= r, \end{aligned}$$

which yield the solutions: $x'_j = \sqrt{\frac{x_j y_j}{r}}$ and $y'_j = \sqrt{r x_j y_j}$. Consequently, the terms of trade for a swap $\Delta x > 0$ sent to AMM j are:

$$\frac{\sqrt{r x_j y_j}}{\sqrt{\frac{x_j y_j}{r}} + \Delta x} = \frac{r}{1+\frac{\Delta x}{\sqrt{\frac{x_j y_j}{r}}}}, \tag{10}$$

where the optimal strategy for the trader is to swap with the AMM j that maximizes $x_j y_j$. This implies that the last term of (9) represents the most favorable terms of trade for the trader in 1.

In 2., the terms of trade for a swap Δx sent to AMM i are:

$$\min\left\{\frac{r}{1+\frac{\Delta x}{x}}, \frac{r_i}{1+\frac{\Delta x}{x_i}}\right\}. \tag{11}$$

We now use (10) and (11) to prove the necessary and sufficient conditions, beginning with the "if" direction. The first inequality in (9) implies that the terms of trade in the 2. are determined by the second term in (11). Combining this with the second inequality in (9) and (10) yields the desired result.

For the "only if" direction, we proceed by contradiction starting with the first inequality in (9). Suppose there exists some $i \in I$ such that:

$$\frac{r}{1+\frac{\Delta x}{x}} \leq \frac{r_i}{1+\frac{\Delta x}{x_i}}.$$

This inequality, along with (10) and (11), implies that to complete the contradiction argument, it suffices to show:

$$x \geq \sqrt{\frac{x_j y_j}{r}} \quad \forall j \in I,$$

To check that this inequality holds true it is replace $r = \frac{y}{x}$ to get:

$$\sqrt{xy} \geq \sqrt{x_j y_j} \quad \forall j \in I,$$

which is clearly satisfied as desired.

To conclude the proof, suppose now that

$$\frac{r}{1+\frac{\Delta x}{x}} > \frac{r_i}{1+\frac{\Delta x}{x_i}} \geq \frac{r}{1+\frac{\Delta x}{\sqrt{\frac{\max\{x_j y_j\}_{j\in I}}{r}}}}, \forall i \in I.$$

The first inequality and (11), mean that the most convenient terms of trade for the trader in 2. are:

$$\max_{i\in I} \frac{r_i}{1+\frac{\Delta x}{x_i}},$$

which by the second inequality (10) are more beneficial to the trader than the most convenient terms of trade in 1., as desired.

To provide an intuitive interpretation of the Proposition, note that, in both the CPMM and GMM algorithms, the terms of trade depend on the ratio of reserves and slippage. For instance, in the case of the CPMM, when there is no arbitrage or non-convergent GMM orders, a swap Δx delivers:

$$\frac{r_i}{1+\frac{\Delta x}{r_i}}\Delta x.$$

In the case of convergent orders under the GMM (priced as in the nGMM algorithm), the swap delivers:

$$\frac{r}{1+\frac{\Delta x}{r}}\Delta x.$$

Similarly, one can show that the CPMM after balanced arbitrage delivers:

$$\frac{r}{1+\frac{\Delta x}{\sqrt{\frac{x_j y_j}{r}}}}\Delta x.$$

Thus, the first inequality in (9) requires that all AMMs price GMM orders using CPMM prices, meaning there are no possible convergent swaps for Δx. If this holds, the second inequality ensures that GMM prices are worse for traders than CPMM prices with arbitrage. This occurs because arbitrage improves the best CPMM price for traders. Clearly, while these conditions can be met in reality, they are very demanding.

The conditions will not hold in the following cases:

– All AMMs have the same reserve ratio. In this case, there is no room for arbitrage in 1. and no convergent orders in 2., meaning there is no difference between the prices offered by the GMM and the CPMM.[5]

[5] Formally, if $r_i = r$, then $\sqrt{\frac{x_j y_j}{r}} = \sqrt{\frac{x_j y_j}{r_j}} = x_j$, which violates the second inequality in (9) for any i.

- All AMMs have different reserve ratios, but Δx is small. Heterogeneous reserve ratios and a small order size mean that some AMMs will display a convergent swap in 2. But convergent swaps always offer better prices to traders than CPMM prices with balanced arbitrage: the reserve ratios remain the same, but slippage is lower because the GMM uses as a reference aggregate reserves.[6]
- All AMMs have the same product of reserves, i.e., $x_j y_j$ is constant across j. Again, convergent swaps offer better prices to traders than CPMM prices with balanced arbitrage, so 1. can be better only when 2.´s best prices for traders are CPMM's prices. But in this case and in both scenarios, AMMs differences can only arise because of differences in reserve ratios. Arbitrage eliminates them in 1., whereas the heterogeneity of reserve ratios in 2. allows traders to find better deals.[7]

For cases where (9) applies and thus worse terms of trade may impact trading activity, there is a natural solution, which we describe next.

First, an important aspect of the GMM that we have not discussed but is now relevant is how swaps between AMMs using the GMM algorithm are priced. Up to this point, we have implicitly assumed that these trades do not occur. While such orders do not require special consideration under the CPMM, they exhibit a particular property in the nGMM: they do not affect the global aggregate (x, y). Thus, for any swap Δx of X sent to an AMM, any corresponding Δy is compatible with maintaining a constant aggregate product.

Here, we adopt the natural exchange rate given by the ratio of reserves, $\frac{y}{x}$, so that $\Delta y_{nGMM} = \frac{y}{x}\Delta x$ ensures that each AMM preserves its value, computed at the current aggregate marginal prices, i.e.,

$$(y_i - \Delta y_{nGMM}) + \frac{y}{x}(x_i + \Delta x) = (y_i - \frac{y}{x}\Delta x) + \frac{y}{x}(x_i + \Delta x) = y_i + \frac{y}{x}x_i.$$

Once the nGMM is defined for these inter-AMM swaps, the GMM can be defined as in Section 3, and the properties discussed there apply in the same manner to these swaps. Indeed, as we shall show later, these swaps are equivalent to other swaps with zero net trade implemented by an external agent. From now on, we assume that the GMM incorporates these specific pricing rules for inter-AMM trades.

Next, we introduce a modification of the GMM designed to guarantee traders an improvement over the CPMM.

Definition 6. *The GMM algorithm with rebalancing is a global algorithm that applies the GMM algorithm to the resulting updated vector of reserves from executing the following algorithm iteratively until one of the condition fails (assume wlog $\Delta x > 0$).*

[6] In the limit as $\Delta x \to 0$, (9) fails, as it converges to $r > r_i < r$ for all $i \in I$.

[7] A constant product $x_j y_j$ means that for any $l \in I$, $\sqrt{\frac{\max\{x_j y_j\}_{j\in I}}{r}} = \sqrt{\frac{x_l y_l}{r}} = x_l\sqrt{\frac{r_l}{r}}$, which is strictly less than x_l for any $r_l < r$. Thus, the second inequality in (9) fails.

1. *Check whether* $l \in \arg\max\{x_j y_j\}_{j \in I}$.
2. *Check whether* (9) *is met.*
3. *Check whether* $\frac{\tilde{y}_l}{\tilde{x}_l} < r$, *for the (updated) vector of reserves* $(\tilde{x}_l, \tilde{y}_l)$.
4. *Update reserves using a swap from AMM* l *to* $j \in \arg\max_{k \in I \setminus \{l\}} \frac{\tilde{y}_k}{\tilde{x}_k}$ *of*

$$\min\left\{\frac{r\tilde{x}_j - \tilde{y}_j}{2r}, \frac{\tilde{y}_l - r\tilde{x}_l}{2r}\right\}$$

   units of X.
5. *Go to 3.*

The GMM algorithm with rebalancing is a variation of the GMM algorithm that, before applying it, checks whether (9) holds. It is under this condition that Proposition 3 states that the AMM provides better terms of trade for the trader under 1. than 2. Specifically, the trader benefits from trading with the AMM that has the largest product of reserves (and thus the lowest slippage) after all arbitrage opportunities are exhausted and all AMMs have a reserve ratio equal to the global ratio.

In this case, the GMM algorithm with rebalancing redistributes swaps to the other AMMs until the AMM's reserve ratio aligns with the global ratio. Consequently, the GMM with rebalancing offers marginal terms of trade at least as favorable as those in the CPMM with arbitrage. Additionally, slippage is lower since rebalancing increases reserves because the AMM captures a portion of the arbitrageurs' profits.

Proposition 4. *Consider an initial distribution of liquidity* $(\boldsymbol{x}, \boldsymbol{y})$ *and a swap* $\Delta x > 0$ *of* X *(wlog) that is optimally routed. Compare the following two scenarios:*

1. *All AMMs use the CPMM, and all strictly profitable arbitrage opportunities in the initial liquidity distribution are exploited by balanced arbitrage before the swap* Δx *is executed.*
2. *All AMMs use the GMM algorithm with rebalancing.*

2. provides better terms of trade for the trader.

Proof. Proposition 3 and the definition of GMM with rebalancing means that we can restrict to the case where (9) is satisfied. Suppose from now on that this is the case. In 1., arbitrage equalises the ratio of reserves of all AMMs. Solving that the product of reserves of each AMM remains constant and that each ratio of reserves must equal to r, one can use the CPMM formula to show that AMM j would respond to the swap of Δx sending back

$$\frac{r}{1 + \frac{\Delta x}{\sqrt{\frac{x_j y_j}{r}}}}, \tag{12}$$

so that the most profitable trade for the trader is the AMM with largest product of reserves, say l. To prove the proposition we shall show that the same AMM

in 2., offers strictly better terms of trade. Since $l = \arg\max_{j \in I} x_j y_j$, the second inequality of (9) implies that

$$\frac{r_l}{1 + \frac{\Delta x}{x_l}} < \frac{r}{1 + \frac{\Delta x}{\sqrt{\frac{x_l y_l}{r}}}} = \frac{r}{1 + \sqrt{\frac{r}{r_l}\frac{\Delta x}{x_l}}}, \tag{13}$$

which implies that $\frac{y_l}{x_l} = r_l < r$, and thus there are other AMMs with ratios of reserves larger than r. The algorithm GMM with rebalancing iterates sending swaps from l to each of other AMMs in step 4. At each iteration in which the updated ratio of reserves of l and j, $\tilde{r}_l \equiv \frac{\tilde{y}_l}{\tilde{x}_l}$ and $\tilde{r}_l \equiv \frac{\tilde{y}_j}{\tilde{x}_j}$ respectively, satisfy $\tilde{r}_l < r < \tilde{r}_j$, there are two possibilities. Consider first that,

$$\frac{r\tilde{x}_l - \tilde{y}_l}{2r} > \frac{\tilde{y}_j - r\tilde{x}_j}{2r}, \tag{14}$$

then the swap sent to j is equal to $\tilde{\Delta}x = \frac{\tilde{y}_j - r\tilde{x}_j}{2r}$ units of X. Applying the GMM with rebalancing formula, the amount of Y obtained with the swap by l is equal to:

$$\begin{aligned}
\min\left\{\frac{\tilde{y}_j}{\tilde{x}_j + \tilde{\Delta}x}, r\right\}\tilde{\Delta}x &= \min\left\{\frac{\tilde{y}_j}{\tilde{x}_j + \frac{\tilde{y}_j - r\tilde{x}_j}{2r}}, r\right\}\tilde{\Delta}x \\
&= \min\left\{\frac{\tilde{r}_j}{1 + \frac{\tilde{r}_j - r}{2r}}, r\right\}\tilde{\Delta}x \\
&= \min\left\{\frac{\tilde{r}_j}{r + r_j}, 1\right\} r\tilde{\Delta}x \\
&= r\tilde{\Delta}x.
\end{aligned}$$

Thus, the swap sent to j increases the ratio of reserves of l to:

$$\frac{\tilde{y}_l + r\tilde{\Delta}x}{\tilde{x}_l - \tilde{\Delta}x} = \frac{\tilde{y}_l + \frac{\tilde{y}_j - r\tilde{x}_j}{2}}{\tilde{x}_l - \frac{\tilde{y}_j - r\tilde{x}_j}{2r}} < \frac{\tilde{y}_l + \frac{r\tilde{x}_l - \tilde{y}_l}{2}}{\tilde{x}_l - \frac{r\tilde{x}_l - \tilde{y}_l}{2r}} = r.$$

Thus, the algorithm goes back to 3. and then 4. again, repeating the process. Note that there are other AMMs with reserve ratios strictly larger than r since the updated ratio of reserves of l is strictly less than r and r is the aggregate ratio of reserves. Consider now the case:

$$\frac{r\tilde{x}_l - \tilde{y}_l}{2r} \leq \frac{\tilde{y}_j - r\tilde{x}_j}{2r}. \tag{15}$$

A similar argument as above implies that the ratio of reserves of l increases to r and then the algorithm stops at 3. once it revisits it again. Since this is the only way the iterative part of the algorithm stops. The amount of Y that the algorithm returns to the trader must be the minimum between the nGMM quantity and the CPMM quantity that corresponds to the updated liquidity reserves. The former is equal to:

$$\frac{y}{x + \Delta x}\Delta x = \frac{r}{1 + \frac{\Delta x}{x}}\Delta x, \tag{16}$$

and the latter to:

$$\frac{\tilde{y}_l}{\tilde{x}_l + \Delta x}\Delta x = \frac{r}{1 + \frac{\Delta}{\tilde{x}_l}}\Delta x = \frac{r}{1 + \frac{\Delta}{x_l\left(\frac{r+r_l}{2r}\right)}}\Delta x, \tag{17}$$

where $(\tilde{x}_l, \tilde{y}_l)$ denotes the final vector of liquidity reserves of l after running the iterative part of the algorithm. Since this part only stops at the point in which $\frac{\tilde{y}_l}{\tilde{x}_l} = r$ and it satisfies $r(x_l - \tilde{x}_l) = y_l - \tilde{x}_l$ we have that $(\tilde{x}_l, \tilde{y}_l) = \left(x_l\frac{r+r_l}{2r}, x_l\frac{r+r_l}{2}\right)$. Since $x_l\frac{r+r_l}{2r} \leq x_l \leq x$, (17) is less than (16), and thus the amount of Δy returned by the algorithm is equal to (17). To finish the proof we show that (17) is greater than (12) for $j = l$. This is equivalent to show that $\frac{r+r_l}{2r} \geq \sqrt{\frac{r_l}{r}}$, which can be easily checked.

One concern, however, is that rebalancing may introduce perverse effects. We argue that this is not the case by showing that rebalancing produces the same outcome for the AMMs as a specific sequence of trades with zero net trade applied to the standard GMM. For simplicity, we illustrate this claim with an example, though the argument is general.

Toy example - Part 7: Rebalancing
Suppose a set of two AMMs with initial vector of reserves

$$(\boldsymbol{x}, \boldsymbol{y}) = (90ETH, 210ETH, 440000UST, 760000UST),$$

and a swap of 1ETH sent by a trader to AMM 2. In this case, the GMM with balancing applies the iterative algorithm sending 10ETH to AMM 1 to get 40000UST in return ($\frac{1200000UST}{300ETH} * 10ETH$). Then, the vector of reserves is updated to

$$(100ETH, 200ETH, 400000UST, 800000UST),$$

and AMM 2 returns to the trader $3980,10UST \approx \frac{800000UST}{200+1} ETH * 1ETH$ since it is a divergent order.
Consider next, the standard GMM and that the swap sent by the trader is preceded by the following two trades sent by an additional trader. The additional trader sends 10ETH to AMM1 to get 38709.67 UST in return ($\approx \frac{1200000UST}{300+10ETH} * 10ETH$ since it is a convergent trade). This updates the vector of reserves to

$$(100ETH, 210ETH, 436.129,03UST, 760000UST),$$

and then sends 38709.67 UST to AMM2 to get 10ETH in return ($\approx \frac{310ETH}{1.161.290,32+38709.67UST} 38709.67UST$ since it is again a convergent trade). After these two trades, the vector of reserves is updated to

$$(100ETH, 200ETH, 400000UST, 800000UST),$$

and now when the original trade is submitted to AMM 2 it returns to the trader $3980,10UST \approx \frac{800000UST}{200+1} ETH * 1ETH$ since it is a divergent order.

Thus, rebalancing neither enables additional manipulative strategies nor alters the cost of manipulation compared to using phony trades. The only caveat is that manipulation with phony trades requires additional asset holdings (e.g., 10 ETH in the example).

4.2 MEV Sandwich Attacks

Next, we show MEV sandwich are strictly less profitable when the AMM uses the GMM algorithm than when it uses the CPMM.

Definition 7. *Given a swap of $\Delta x > 0$ units of X submitted to AMM i, a MEV sandwich $\widehat{\Delta x} > 0$ consists of two swaps submitted before and after Δx. The first one of $\widehat{\Delta x}$ units of X and the second one of the units of Y obtained with the first swap.*

A MEV sandwich may be profitable due to slippage: the first order buys X more cheaply than the price at which the second order sells because there is

an order in between that makes X more scarce at the AMM. This is clearly the case when the AMM uses CPMM in which prices reflect the scarcity at the AMM but it is less so in the case of the GMM. In this later case, the first buy order is satisfied at a price that reflects the scarcity created by the order but the subsequent sell order is satisfied at a price that reflects the much smaller scarcity existing in the set of AMMs. The reason for this is that one would expect that AMMs should start from an equilibrium situation in which all (marginal) prices are the same (and thus local levels of scarcity) and thus the first buy order moves the AMM away from the other AMMs whereas the sell order moves the AMM towards the other AMMs. Next proposition formalises this argument.

Proposition 5. *The profit of a MEV sandwich is strictly lower when the AMM uses GMM than when the AMM uses CPMM.*

Proof. In this proof, we shall use the property that the GMM weakly increases the product of reserves, strictly if in the convergent region. This is a consequence of the definition of the CPMM, that keeps constant the product of reserves, and the definiiton of the GMM, that returns either the same amount than the CPMM or strictly less in the case of convergent orders.

Suppose the same order $\Delta x > 0$ (the case $\Delta x < 0$ is symmetric) and MEV sandwich $\widehat{\Delta x}$ submitted to AMM i operating under two alternative protocols, CPMM and GMM. The corresponding profits for the trader issuing the MEV sandwich are equal to her final net positions in Y, this is:

$$-\left(\widehat{\Delta y}_s + \widehat{\Delta^F y}_s\right),$$

for $s \in \{CPMM, GMM\}$ and where:

$$\left\{(\widehat{\Delta x}, \widehat{\Delta y}_s), (\Delta x, \Delta y_s), (-\widehat{\Delta x}, \widehat{\Delta^F y}_s)\right\},$$

is a sequence of feasible trades in AMM i with protocol s.

We first note that $-\Delta y_{CPMM} \geq -\Delta y_{GMM}$, with the inequality strict if $\widehat{\Delta x}$ is not price by the GMM in the divergent region. This follows from the fact that Corollary 2 implies that

$$(x_i^0 + \widehat{\Delta x})(y_i^0 + \widehat{\Delta y}_{GMM}) \geq (x_i^0 + \widehat{\Delta x})(y_i^0 + \widehat{\Delta y}_{CPMM}),$$

where the inequality is strict if the order is not in the divergent region.

Next, we show that $-\Delta y^F_{CPMM} \geq -\Delta y^F_{GMM}$. Since the two consecutive orders $\widehat{\Delta x}$ and Δx are priced in both algorithms as a single order $\widehat{\Delta x} + \Delta x$, we can again conclude from Corollary 2, that

$$y_i^0 + \widehat{\Delta y}_{GMM} + \Delta y_{GMM} \geq y_i^0 + \widehat{\Delta y}_{CPMM} + \Delta y_{CPMM},$$

with the inequality strict if either of the two orders $\widehat{\Delta x}$ and Δx is not priced by the GMM in the divergent region. From this inequality and the fact that

$$\widehat{\Delta^F y}_{CPMM} = \frac{y_i^0 + \widehat{\Delta y}_{CPMM} + \Delta y_{CPMM}}{x_i^0 + \Delta x}\widehat{\Delta x}$$

means that,

$$\widehat{\Delta^F y}_{CPMM} \leq \widehat{\Delta^F y}_{AUX} \equiv \frac{y_i^0 + \widehat{\Delta y}_{GMM} + \Delta y_{GMM}}{x_i^0 + \Delta x} \widehat{\Delta x},$$

with the inequality strict if either of the two orders $\widehat{\Delta x}$ and Δx is not priced by the GMM in the divergent region. Keep this inequality in mind, and not that the definition of $\widehat{\Delta^F y}_{AUX}$ also means that,

$$\left(y_i^0 + \widehat{\Delta y}_{GMM} + \Delta y_{GMM} + \widehat{\Delta^F y}_{AUX}\right)\left(x_i^0 + \Delta x\right) = \left(y_i^0 + \widehat{\Delta y}_{GMM} + \Delta y_{GMM}\right)\left(x_i^0 + \widehat{\Delta x} + \Delta x\right).$$

Applying again Corollary 2, we can conclude that the right hand side of the last equation is less than:

$$\left(y_i^0 + \widehat{\Delta y}_{GMM} + \Delta y_{GMM} + \widehat{\Delta^F y}_{GMM}\right)\left(x_i^0 + \Delta x\right),$$

and thus $\widehat{\Delta^F y}_{AUX} \leq \widehat{\Delta^F y}_{GMM}$, with the inequality strict when $-\widehat{\Delta x}$ is not priced by the GMM in the divergent region. Since at least one of the orders $\widehat{\Delta x}$, Δx and $-\widehat{\Delta x}$ must not be priced by the GMM in the divergent region, we have that $\widehat{\Delta^F y}_{CPMM} < \widehat{\Delta^F y}_{GMM}$ as desired.

Next, we redo the MEV extraction example computed for CPMM in Example 2 in the case of GMM and we observe that the extracted MEV is less in the GMM case.

Toy example - Part 7: MEV extraction in GMM Consider 2 GMMs (GMM1 and GMM2) with initial reserves of 100 ETH and 400,000 UST each. Suppose that a trader issues an order to buy 10 ETH 10 from GMM1 and that the MEV extractor launches a sandwich attack of 15 ETH. The transactions of the MEV-extractor and victim will be processed as follows:
Frontrunning: MEV-extractor transaction 1 that buys 15 ETH. The resulting reserves after this initial transaction are 85 ETH and 470,588 UST. The marginal price price has increased to 5,536 UST/ETH. The MEV-extractor has spent 70,588 UST. This is the same as Example 2.
Victim's transaction: The victim buy 10 ETH. The resulting reserves are then 75 ETH and 533,333 UST. The victim has payed 62,745 UST i.e. a surplus of 18,301 UST compared to what she would have payed without the frontrunning. Again, this is the same as Example 2.
Backrunning. The MEV-extractor sells the 15 ETH. In this case, GMM1 uses the convergent formula, which yields that the MEV-extractor will obtain 73,684 UST, making a net profit of 3,096 UST, which is 15,205 UST less than in Example 2.

4.3 Impermanent Losses

Finally, we show that impermanent losses are also lower when the AMM uses the GMM algorithm instead of the CPM. The impermanent losses refer to the losses associated to liquidity provision in in an AMM and are defined as as the difference between the value of the initial reserves (x_i^0, y_i^0) and the final reserves (x_i^1, y_i^1) after a change in the market price. In the CPMM, there is only one way to determine how the initial reserves translate into current reserves, but in the GMM we need to determine how the transition occurs. As it turns out, these transitions are irrelevant if the transition in monotone in the sense that the marginal prices in all the AMMs move monotonically to the new price. We model this transition assuming that a competitive trader submits orders in an exogenous sequence (potentially, the optimal one to the competitive trader) to each of the AMMs equalizing each of the marginal prices to the new market price.

Definition 8. *Suppose an initial price of X in terms of Y of r^0 and a set of AMMs I, each with a ratio of reserves $\frac{y_i}{x_i} = r^0$, $\forall i \in I$. The impermanent loss of AMM i associated to a change of prices from r^0 to r^1 is equal to:*

$$\left(r^1 x_i^0 + y_i^0\right) - \left(r^1(x_i^0 + \Delta x_i) + y_i^0 + \Delta y_i,\right)$$

where $(\Delta x_i, \Delta y_i)$ is determined by an exogenous sequence of trades of a competitive trader.

Proposition 6. *The impermanent losses of AMM i use GMM are lower than the impermanent losses when AMM i uses CPMM, strictly if AMM i is not the first one in the exogenous sequence of trades use by the competitive trader.*

Proof. We shall use in our proof that $x * y = k$ and $\frac{y}{x} = r$ has as unique solution $x = \sqrt{\frac{k}{r}}$ and $y = \sqrt{r * k}$. Thus, the value of the reserves of an AMM with reserves satisfying $x_i * y_i = k$ and with a marginal price $\frac{y_i}{x_i} = r$ at price r' is equal to:

$$r' * x_i + y_i = \left(\frac{r'}{r} + 1\right)\sqrt{k * r}.$$

This formula and the property that the CPMM has a constant size means that the impermanent loss of a CPMM that starts with reserves (x_i^0, y_i^0) after a price increase from r^0 to r^1 is equal to:

$$\left(\frac{r^1}{r^0} + 1\right)\sqrt{x_i^0 * y_i^0 * r^0} - 2\sqrt{x_i^0 * y_i^0 * r^1},$$

whereas the same price increase in the GMM gives impermanent losses:

$$\left(\frac{r^1}{r^0} + 1\right)\sqrt{x_i^0 * y_i^0 * r^0} - 2\sqrt{x_i^1 * y_i^1 * r^1},$$

where $x_i^1 * y_i^1$ is the product of reserves after the price increase. By Corollary 2, $x_i^1 * y_i^1 \geq x_i^0 * y_i^0$, strictly if the AMM is not the first one in the exogenous sequence, which implies the proposition.

The next example illustrate this claim.

Toy example - Part 8: Impermanent losses in GMM Consider two GMMs with liquidity pools containing 100 ETH and 400,00 UST. The marginal price is then 4,000 UST/ETH.
Similarly than in Example 3, suppose that one or more traders submit transactions for buying 30 ETH on each GMM.
We can compute impermanent loss IL) as the difference between the value of the GMM's original reserves valued at the current price and the current reserves valued at the current price.
To make a simple example, we assume that a first order for 30 ETH arrives to GMM1 and a second order for 30 ETH arrives to GMM2. The final reserves for GMM1 are the same as in one of the CPMMs in Example 3, namely 70 ETH and 571,429 UST.
The final reserves of GMM2 resulting from the second trade of 30 ETH can be computed using the convergent formula for 28 ETH, at point which the relative prices are equalized for the two GMMs and the divergent formula for the remaining 2 ETH.
Overall, this implies that the impermanent loss for GMM1 and GMM2 are equal to 129,023 UST, which is 17,899 UST less than in Example 3.

The next section provides simulations using real life transactions and provides a quantitative assessment of how large this reduction of impermanent losses is.

5 Numerical Evaluation

In our data set, we consider transactions in two AMMs in the Ethereum network, Uniswap[8] V2 and Sushiswap, and one of the most traded pairs USDT/WETH, and three other less traded pairs ALX/WETH, BLOCKS/WETH and VSP/WETH. For the USDT/WETH pair we consider three different periods: two periods with high price variation and one period with low price variation.

First, we shall consider our results with respect to arbitrage. For this, we first illustrate that our assumptions of optimal routing are consistent with the data. In Table 1, we estimate the optimality of routing by computing the unrealised profits from not optimally routing over the volume of trade.

Next, we illustrate the potential benefits of replacing the CPMM with GMM. For that we compute in the different periods the amount of arbitrage profits with respect to the fees revenues of the AMMs. Our results show that the gains for a traded pair are modest ranging from 0.23% to 5.82%. For the other less traded pairs, the gains are significant which suggests that the GMM can be particularly effective for less traded pairs.

To assess the effectiveness of the GMM algorithm in reducing the profitability of sandwich operations, we analyze all such operations that occurred in selected

[8] We chose Uniswap V2 instead of V3 due to the similar fee structure that it has compared to Sushiswap for ease of exposition.

Pairs WETH/	Profits from Routing / Volume of Trade
USDT (11565020-11640849)	0.0003
USDT (12390294-12467913)	0.0007
USDT (13317339-13406729)	0.00016
ALX (1948971-12050004)	0.0015
BLOCKS (14356780-14497019)	0.017
VSP (13528195-13917471)	0.023

Table 1. Routing Errors

trading pairs between 2021 and 2023. We use reserves at the beginning of each block where a sandwich operation appears and simulate the entire block of transactions. For the counterpart AMM, we use its reserves at the start of the same block but keep them constant, disregarding trades that may have occurred in parallel with the sandwich operation.

We exclude operations that result in an "imbalanced" position[9] and, for this purpose, equate the frontrun operations with the backrun. If an operation remains unprofitable after this adjustment, we discard it. Our simulations suggest that this exclusion does not introduce significant bias, as the simulated MEV operators' profits closely match those reported by zeromev.

Finally, we simulate the same trades using a GMM design. Our results, presented in Table 3, show that in all cases, the GMM design significantly reduces the profitability of sandwich operations and, in many cases, renders them unprofitable.

Pairs WETH/	Profitable Mev operations(CPMM)	Sandwich Profits (CPMM)	Profitable Mev operations(GMM)	Sandwich Profits (GMM)
USDT (uni)	446	988,254$	177	260,000$
USDC (uni)	240	853,872$	90	140,000$
DAI (uni)	443	922,269$	94	84,311$
WBTC (uni)	123	4 WETH/20 BTC	18	0.1 WETH/0.78 BTC
LINK(uni)	464	8,949 LINK/85 WETH	200	1,479 LINK/9.86 WETH
LDO (Sushi)	1,152	139,336 LDO/260 WETH	1,114	132,183 LDO/245 WETH
CRV (Sushi)	659	140,345 CRV/197 WETH	473	21,102 CRV/90 WETH

Table 3. Simulations for Sandwich operations

[9] Measuring the profits of imbalanced sandwich positions is an open question in the literature and beyond the scope of this paper.

References

1. Dex tracker - decentralized exchanges trading volume. https://www.defiprime.com/dex-volume (2023), [Online; accessed 12-August-2023]
2. Aldridge, I.: Slippage in amm markets. SSRN Electronic Journal (2022). https://doi.org/10.2139/ssrn.4133897, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4133897
3. Aoyagi, J.: Liquidity provision by automated market makers (2023), https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3674178, last updated April 2023
4. Buterin, V.: Let's run on-chain decentralized exchanges the way we run prediction markets. Reddit post (2016), https://www.reddit.com/r/ethereum/comments/55m04x/lets_run_onchain_decentralized_exchanges_the_way/
5. Canidio, A., Danos, V.: Commitment against front-running attacks. Management Science **70**(7), 4429–4440 (2024). https://doi.org/10.1287/mnsc.2023.01239, https://arxiv.org/abs/2301.13785
6. Canidio, A., Fritsch, R.: Arbitrageurs' profits, lvr, and sandwich attacks: Batch trading as an amm design response (2024)
7. Chi, T., He, N., Hu, X., Wang, H.: Remeasuring the arbitrage and sandwich attacks of maximal extractable value in ethereum (2024)
8. Cong, L.W., Li, X., Tang, K., Yang, Y.: Tokenomics: Dynamic adoption and valuation. Management Science **67**(7), 3919–3945 (2021). https://doi.org/10.1287/mnsc.2021.02802, https://pubsonline.informs.org/doi/full/10.1287/mnsc.2021.02802
9. Daian, P., Goldfeder, S., Kell, T., Li, Y., Zhao, X., Bentov, I., Breidenbach, L., Juels, A.: Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges (2019)
10. Gramlich, V.: Maximal extractable value: Current understanding, categorization, and open research questions. Electronic Markets (2020)
11. Hafner, M., Dietl, H.: Impermanent loss conditions: An analysis of decentralized exchange platforms. arXiv preprint (2024), https://arxiv.org/abs/2401.07689
12. Hansson, M.: Arbitrage in crypto markets: An analysis of primary ethereum blockchain data. Available at SSRN 4278272 (2022)
13. Lehar, A., Parlour, C.A.: Decentralized exchange: The uniswap automated market maker. Journal of Finance **79**(6), 1234–1278 (2024). https://doi.org/10.1111/jofi.13405
14. Mazor, O., Rottenstreich, O.: An empirical study of cross-chain arbitrage in decentralized exchanges. Cryptology ePrint Archive, Paper 2023/1898 (2023), https://eprint.iacr.org/2023/1898, https://eprint.iacr.org/2023/1898
15. McLaughlin, R., Kruegel, C., Vigna, G.: A large scale study of the ethereum arbitrage ecosystem. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 3295–3312. USENIX Association, Anaheim, CA (Aug 2023), https://www.usenix.org/conference/usenixsecurity23/presentation/mclaughlin
16. Milionis, J., Moallemi, C.C., Roughgarden, T.: Automated market making and arbitrage profits in the presence of fees (2023)
17. Milionis, J., Moallemi, C.C., Roughgarden, T., Zhang, A.L.: Automated market making and loss-versus-rebalancing (2024)
18. Qin, K., Zhou, L., Gervais, A.: Quantifying blockchain extractable value: How dark is the forest? (2021)

19. Weintraub, B., Torres, C.F., Nita-Rotaru, C., State, R.: A flash(bot) in the pan: Measuring maximal extractable value in private pools. In: Proceedings of the 22nd ACM Internet Measurement Conference. p. 458–471. IMC '22, ACM (Oct 2022). https://doi.org/10.1145/3517745.3561448, http://dx.doi.org/10.1145/3517745.3561448
20. Xavier Ferreira, M.V., Parkes, D.C.: Credible decentralized exchange design via verifiable sequencing rules. In: Proceedings of the 55th Annual ACM Symposium on Theory of Computing. p. 723–736. STOC '23, ACM (Jun 2023). https://doi.org/10.1145/3564246.3585233, http://dx.doi.org/10.1145/3564246.3585233
21. Zhou, L., Qin, K., Gervais, A.: A2mm: Mitigating frontrunning, transaction reordering and consensus instability in decentralized exchanges (2021), https://arxiv.org/abs/2106.07371