Fair Batch Matching Liquidity Pools for Sandwich-Proof DeFi

January 31, 2025

Coin Fu Master Shifu

Abstract—**We propose the only fair method to jointly match continuous liquidity provided from liquidity pools with discrete liquidity in the form of an order book. In block chain context, our method represents the canonical way of batch matching DeFi swap orders with pool liquidity within a block such that their order is irrelevant. This allows to overcome the infamous front and back running issues that plague today's DeFi platforms, most notably in the form of Sandwich Attacks.**

I. Introduction

Decentralized Finance (DeFi) protocols that have emerged in recent years have taken an important role in the field of crypto asset finance with DeFi trading volume reaching a new all time high of almost \$50b daily trading volume[1] in January 2025. Despite their importance, these protocols suffer from a fundamental flaw that is turning away a large amount of potential high-volume trading activity: transaction ordering manipulation allows third parties to affect an investor's DeFi swaps negatively for profit.

Maximal Extractable Value (MEV) describes the maximal amount of profit that can be extracted from such protocols by adding, omitting and reordering transactions by block creators. Such profit usually comes with a cost to victims' transactions and imposes extra economical friction for today's DeFi users in addition to transaction fees.

The classical example for such practice is the *Sandwich Attack*, in which a victim's order of large volume is enclosed by two orders to buy low and sell hight before and after the victim buys, or sell high and buy low before and after the victim sells respectively. This yields a risk-less arbitrage profit for the attacker but negatively affects the fill price of the victim's order.

There exists substantial academic research on how to protect against MEV attacks [1] ranging from encrypted mempools and permutations [2] to enforcing a specific transaction ordering [3], [4], [5]. Still, in practice DeFi users must rely on primitive best practices such as order splitting, decreasing swap slippage [6] or using premium services that shield users' orders from the public mempool. However these techniques are just band aids and are not addressing the fundamental underlying problem which is the sensitivity of DeFi swaps' fill prices on their arbitrarily modifiable ordering within blocks.

We propose *Fair Batch Matching* (FBM), a new method to match an entire order book against a liquidity pool, which allows to completely drop the concept of transaction ordering. This renders MEV extraction within blocks impossible and opens the door to a new generation of Sandwich-Proof DeFi protocols.

II. Fair Batch Matching

An *Automated Market Maker* (AMM) is a program that runs in a peer-to-peer network acting as a liquidity provider for *Liquidity Takers* (LTs) [8]. When the internal rules of an AMM are determined by a deterministic function $g : \mathbb{R}_+^2 \to \mathbb{R}$ that determines interactions offered to LTs, it is called a *Constant Function Market Maker* (CFMM). In this paper we are considering the most relevant CFMM, which is the *Constant Product Market Maker* (CPMM) with constant g also used by Uniswap v1, v2, and v3 [8].

A. Liquidity Pool

The liquidity of an AMM is stored in a *liquidity pool*, which controls a base asset balance q^B a quote asset balance q^Q. It supports two interactions:

1) Depositing/Withdrawing liquidity
2) Exchanging liquidity

For our purposes only the second type of interaction is relevant, and since g is constant in a CPMM, allowed changes $\Delta^B > -q^B$ and $\Delta^Q > -q^D$ in the pool's base and quote asset reserves are linked by the constant product equation

$$(q^B + \Delta^B)(q^Q + \Delta^Q) = q^B q^Q, \tag{1}$$

such that the pool's exchange price for one unit of the base asset denoted in the quote asset amounts to

$$p_{\text{Pool}} = \lim_{\substack{\Delta^Q \to 0 \\ \Delta^Q \neq 0}} \tfrac{\Delta^Q}{-\Delta^B} = \tfrac{q^Q}{q^B}. \tag{2}$$

B. Order Books

We call a pair $(l, q) \in \mathbb{R}_+^2$ a *swap order* with limit price l and swap quantity q. For each order we can assign a *fill quantity* $f \in [0, q]$ and say the order is *not filled*, *partially filled*, *filled* if $f = 0$, $f \in (0, q)$, $f = q$ respectively.

To model an order book let $(l_1^B, q_1^B), \dots, (l_m^B, q_m^B) \in \mathbb{R}_+^2$, $m \in \mathbb{N}_0$ and $(l_1^Q, q_1^Q), \dots, (l_n^Q, q_n^Q)$, $n \in \mathbb{N}_0$ be two groups of orders that we call *base swap orders* (or *sell orders*) and *quote swap orders* (or *buy orders*). Within each group we require the limit price to be increasing with index such that we don't allow multiple base or quote swap orders at the same price here.

[1] https://defillama.com/dexs

For a price $p \in \mathbb{R}_+$, a valid *fill configuration* is a collection of fill quantities $f_1^B, \ldots, f_m^B, f_1^Q, \ldots, f_n^Q$ for these orders such that:

1) Sell orders with smaller, and buy orders with higher limit price are filled, i.e. $l_i^B < p \implies f_i^B = f_i^B$ and $l_j^Q > p \implies f_j^Q = f_j^Q$ for $i \le m$, $j \le n$.
2) Sell orders with higher, and buy orders with smaller limit price are not filled, i.e. $l_i^B > p \implies f_i^B = 0$ and $l_j^Q < p \implies f_j^Q = 0$ for $i \le m$, $j \le n$
3) At most one order can be partially filled.

In this context we define the fill sums $F^B := \sum_{i=1}^M f_i^B$ and $F^Q := \sum_{j=1}^N f_j^Q$ for the base and quote swap orders, depending implicitly on the price and the valid fill configuration.

C. Pool Interaction

Since we allow interactions with a liquidity pool, matching of orders is done by deciding on a proportion of quote or base fill sum that is exchanged at the pool, then the remainder of fill sums is directly exchanged between the liquidity takers.

Given a fill configuration of a price p, a *pool interaction* is a pair $\Delta^B \le F^B, \Delta^Q \le F^Q$ satisfying equation (1). If $\Delta^B > 0$ we regard this as the proportion of F^B that is exchanged at the pool for $-\Delta^Q$ in return. Similarly, if $\Delta^Q > 0$, it is the proportion of F^Q that is exchanged at the pool for $-\Delta^B$.

D. Fair Pool Interaction

Buyers and sellers want to act optimally:

1) If $F^B = 0$, buyers will completely swap the quote fill sum at the pool $\Delta^Q = F^Q$ as they don't want to directly exchange non-zero remainder of the quote fill sum $F^Q - \Delta^Q$ for base fill sum 0.
2) Else, if $F^Q = 0$, sellers will completely swap the base fill sum at the pool$\Delta^B = F^B$ as they don't want to directly exchange non-zero remainder of the base fill sum $F^B - \Delta^B$ for base fill sum 0.
3) Else, if the pool price (2) is higher than the fill ratio $p_{Pool} > \frac{F^Q}{F^B}$, base asset sellers can maximize their average selling price by swapping a part Δ^B of the base fill sum F^B at the pool such that
$$\frac{F^Q}{F^B - \Delta^B} = \frac{q^Q + \Delta^Q}{q^B + \Delta^B}. \tag{3}$$
4) Else, if the pool price is smaller than the fill ratio $p_{Pool} < \frac{F^Q}{F^B}$, base asset buyers can minimize their average purchase price by swapping a part Δ^Q of the quote fill sum F^Q at the pool such that
$$\frac{F^Q - \Delta^Q}{F^B} = \frac{q^Q + \Delta^Q}{q^B + \Delta^B}. \tag{4}$$

A pool interaction (Δ^B, Δ^Q) satisfying the above is *fair*, the situation can be also seen as a *Nash equilibrium* because neither buyers nor sellers can improve their average conversion price by interacting differently with the pool. The *final pool price* after the interaction is $\frac{q^Q + \Delta^Q}{q^B + \Delta^B}$.

E. Fair Batch Matching

A fair pool interaction can only exist if the fill configuration is valid for the final pool price, otherwise the orders involved in the fill sums can't be actually filled. We formalize existence as follows.

For a given liquidity pool and order book, a *Fair Batch Matching (FBM)* is a triple consisting of a price p, a corresponding valid fill configuration and a fair pool interaction (Δ^B, Δ^Q) with final pool price p, i.e. $p = \frac{q^Q + \Delta^Q}{q^B + \Delta^B}$.

Theorem II.1. *There always exists a unique Fair Batch Matching.*

The above theorem allows us to propose FBM as the canonical and only fair way of matching multiple swap orders with an order book.

III. Sandwich-Proof DeFi

A. Reasons for Sandwich Attacks

Current DeFi protocols only allow a single swap order to interact with a liquidity pool, multiple pool swap orders are serially executed against the pool liquidity, which is affected after each interaction. This opens the door for block creators to front-run and back-run, i.e. sandwich, a victim's order for profit worsening the victims price as much as possible. This kind of attack is possible in these systems mainly due to two reasons:

1) Swap are usually publicly visible in the order book and irrevocable before the next block.
2) The order of swap transactions within a block can be controlled by attackers to form a sandwich around a victim's order.

There are some attempts to solve this problem by tackling the first reason above, i.e. by trying to keep orders private from the public. However the real solution is to address Sandwich Attacks and MEV extraction more fundamentally and to eliminate the second reason by removing the order concept of DeFi swap transactions within blocks altogether.

B. Removing the ordering of Swap Transactions

Our proposed Fair Batch Matching can be used to match swap orders in order book with a liquidity pool. For each asset pair, we can exploit this capability by creating an aggregated order book of all swap orders within a block with at most one aggregated order at each price level, matching this order book with the corresponding liquidity pool, and distributing the matched assets in a fair way among all individual orders that made up the filled aggregated orders such that all buyers get the same purchase price and all sellers get the same selling price.

Obviously, this way there is neither an ordering in the execution of DeFi swaps nor any is there way of buying or selling an asset at different prices than everyone else in this setting. In particular DeFi Sandwich Attacks within a block are no longer be possible.

IV. Summary

In this work we have proposed Fair Batch Matching (FBM), a new method for matching discrete liquidity in the form of an order book with continuous liquidity in the form of a liquidity pool. Our method is the only fair and therefore the canonical way to find a matching in this setting and we have shown that there always exists a unique such matching.

Compared to traditional DeFi matching where only a single swap order is applied to a liquidity pool at a time, we can match a multitude of swap orders jointly without the need for an implicit ordering. This eliminates the possibility for the parasitic practice of MEV extractors within blocks as these require an order-dependent execution of swaps that they can abuse to their advantage. In particular the dreaded Sandwich Attack can no longer be executed if our proposed order-less FBM method is applied, thus relieving significant financial burden from DeFi traders. We therefore think that our matching method will become a significant building block for the next generation of DeFi implementations.

Appendix

The following lemma is a direct consequence of the definition of valid fill configurations in Section II-B.

Lemma A.1.
1) *F^Q is non-increasing in p, i.e. when the price p is increased fill sum F^Q of valid fill configurations cannot increase.*
2) *F^B is non-decreasing in p, i.e. when the price p is increased fill sum F^B of valid fill configurations cannot decrease.*

Proof of Theorem II.1. The four conditions in Section II-D can be summarized as

$$0 = \begin{cases} F^Q - \Delta^Q - pF^B & \text{if } F^Q - pF^B \geq 0 \\ F^Q - pF^B + p\Delta^B & \text{else} \end{cases}. \tag{5}$$

The pool interaction in equation (1) can be reformulated as $\Delta^Q = q^Q q^B/(q^B + \Delta^B) - q^Q$ or $\Delta^B = q^B q^Q/(q^Q + \Delta^Q) - q^B$. Plugging this into $p = \frac{q^Q + \Delta^Q}{q^B + \Delta^B}$ and combining with equation (5) shows that an FBM exist if and only if we can find a price $p \in (1, \infty)$ and a corresponding valid fill configuration with

$$q^Q q^B = \begin{cases} \frac{1}{p}\left(q^Q + F^Q - pF^B\right)^2 & \text{if } pF^B \leq F^Q \\ p\left(q^B + F^B - \frac{1}{p}F^Q\right)^2 & \text{else} \end{cases}. \tag{6}$$

By Lemma A.1 it follows that the right hand side of equation (6) is strictly increasing in p, no matter which valid fill configuration is chosen for each p. Since the left hand side is constant, this shows that if an FBM exists, p is uniquely determined. But also the fill configuration is uniquely determined since no two valid fill configurations can have the same fill sums their definition in Section II-B. Finally also Δ^B and Δ^Q are uniquely determined by p by equations (5) and (1). This proves that there can be at most one FBM.

But since the right hand side of equation (6) can attain every value in $(0, \infty)$ for appropriate choice of p and a valid fill configuration, there is always exactly one FBM. □

References

[1] S. Yang, F. Zhang, K. Huang, X. Chen, Y. Yang, and F. Zhu, "Sok: Mev countermeasures," in *Proceedings of the Workshop on Decentralized Finance and Security*, ser. DeFi '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 21–30. [Online]. Available: https://doi.org/10.1145/3689931.3694911

[2] A. Kavousi, D. V. Le, P. Jovanovic, and G. Danezis, "BlindPerm: Efficient MEV mitigation with an encrypted mempool and permutation," Cryptology ePrint Archive, Paper 2023/1061, 2023. [Online]. Available: https://eprint.iacr.org/2023/1061

[3] C. Cachin, J. Mićić, N. Steinhauer, and L. Zanolini, "Quick order fairness," in *International Conference on Financial Cryptography and Data Security*. Springer, 2022, pp. 316–333.

[4] M. Kelkar, S. Deb, S. Long, A. Juels, and S. Kannan, "Themis: Fast, strong order-fairness in byzantine consensus," Cryptology ePrint Archive, Paper 2021/1465, 2021. [Online]. Available: https://eprint.iacr.org/2021/1465

[5] M. Kelkar, F. Zhang, S. Goldfeder, and A. Juels, "Order-fairness for byzantine consensus," Cryptology ePrint Archive, Paper 2020/269, 2020. [Online]. Available: https://eprint.iacr.org/2020/269

[6] L. Heimbach and R. Wattenhofer, "Eliminating sandwich attacks with the help of game theory," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 153–167. [Online]. Available: https://doi.org/10.1145/3488932.3517390

[7] S. Yang, F. Zhang, K. Huang, X. Chen, Y. Yang, and F. Zhu, "Sok: Mev countermeasures," in *Proceedings of the Workshop on Decentralized Finance and Security*, 2024, pp. 21–30.

[8] V. Mohan, "Automated market makers and decentralized exchanges: a defi primer," *Financial Innovation*, vol. 8, no. 1, p. 20, Feb 2022. [Online]. Available: https://doi.org/10.1186/s40854-021-00314-5

Coin Fu Master Shifu prefers to stay anonymous at this time. Should he ever want to claim credit for this work may here be his PGP public key:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQGNBGUzn5YBDACio8JQd9kKAoKEi137Pmn4E3b65WykoK7tYg3PNeIg6EAS+OuZ
G1u6I//iv2YBuzV1Jl3W0f9uMqJodZVhkWn/CLurUg7zyGXFgN8qkgI/pkq7o/JD
txAwwuD+kTy++NNsa7+OMa5PgV/q1d+ieeYjM0XLXTb8Kk3hUQx5us/jIrAavm35
LyDK0yHqG0HesAuowk0pjPI0zU2JUQbrYSbKbh0Zmuvy4QB78bgDt+Q4eFI+cg+D
Qf4BuMKOFsYHSu9+OmOb5sFpI8U3bjWN7izhJcsUYMouu9AttY3nWptG1Bpg2/nk
1iHoEwq7j3XKaO6doG8BY6IN+YrR3a37kXlj+StIqvdOAqaDVrIrgG1+BbvGjeU+
NbaZWWMbArLGdeyyPMaCxI0thYkqyXYNku2pTGTZUNGqYVyzoG3T81MVEkHTHGZ4
LTdGYqlsVf/E1+iPxZ66mRl9LkE7RBa+huWk+h5OW10RrpBVcJuvTeefGacBESvp
Ho5YhjQvPCGy3TEAEQEAAbQUQ29pbiBGdSBNYXN0ZXIgU2hpZnWJAc4EEwEKADgW
IQTR/W1/HXMAZ+gJgX3YQBkptMgPVQUCZTOflgIbAwULCQgHAgYVCgkICwIEFgID
AQIeAQIXgAAKCRDYQBkptMgPVWJqC/wIACt8XGShBcj5n6Mhswq7oIDhMvHhiCSl
0grOGDKLbqSnnXv5MiZb8XW60qFMW1RXzVWXpQSXeSugnDPuvYEFwSKwbTb3x9xW
sW0WNpyLNJqnKAkomwaiPimCX/V00sNWE7c/SSGGPjEGk7UQx4xMcUwAf0QoPRXO
dPeTQ6R1CtuSbZpa40ytQ0ywbxQJ0AKteH4anEt6AwNuKNFmD4/Et19hB4PcCeLG
VHUjqjuTFCK8BEnWgq4fPpuAtCQjhXOY9Ynnwq+P88/7iIs14z/QS+5gclmYQ/R2
5mof/v7xBE1Ol/cU8KHdbdYPzLUgxfX5ewZnRx5hjMGzAPmjhAzuKnVnIHuuO7BY
q9JXagH1ZFYO1vT6T/R5+TNmaaVdZfU2vL231CI+EZ6Q6191NTWzdBQn7Ly+wpEU
/axyVZ6NQ4uQuwowX1otcVwcjVCsZvV5ZVTgq8hWXXWTKtrOpb70Vow/fdxy0XwC
AhBCCtBYFqscyzNr8j33QvYHF2cpCf25AY0EZTOflgEMAMX+DH0JqK3L6WKaNmll
YOf2v5D0rByQySaRfzZ5nAkmLHjwUXuok1WGh/23ZX5GIHxPlhcfaAYJflzWe1z6
U8EEetw8g0mgzZ6+sEFrs6i74vlm36Cz0zJ+YxDiaxy+dbPCTOHJg8ZR74wEhc8y
IaCeKnNHX5oK15cBFRBtYKVgs4WBLE6bg2Yt0eE8kUDhReiWvoPczxZg9bBdGdXI
Z0lgIBe+F6fIP6UW7jEQN71O3K83mNbb97Mb13DW6hjhw6OGJnCj+R6Z0+paOL0G
kPV8gTpkTtJhXSBTiG6Br7uXlB8DWR2GDMcKwr2QjKmIisrQuQvQT9cUgFCLSXf2
OcKlUQwWHA+8VTVw9s/HgF4ghfegsidmesDqtg8IUnVUbXhVl3Ta4nGoOgKf4um3
AlP2wcR0l1eD9wQtsczqUssjhfqYr56JI/hAkgm//aW4rpy1Hxd3APJhRPm2y2w6
sXhwvmzEabV1qU7fqX9hXP8cyRCzLmVi7jj55Xqe4GbnEQARAQABiQG2BBgBCgAg
FiEE0f1tfx1zAGfoCYF92EAZKbTID1UFAmUzn5YCGwwACgkQ2EAZKbTID1VqWgv/
T310IhZzRX9lzo9UgpJmoiCo5F9b4JCXf1S9O2srIRBjHzzoikWmqRUpHK1iTYnr
bODFdyW20+l/b60vzWjfotgZM7oFtHK8TNNhzx/p8judAsp6RLffPz+c7c92eOZj
4UoeQGndvIZM8cDyNEPj9VKLI2Tnk6vAgcR/fEkrGFinMCKwJ1s7Ct6QNZAPhy3+
1xgHw5hsQARv9V/xEGP14CkPvjW5m2HdFUiHL602xQ2VTzVeV3VW/Pl3sVnbMq4N
4zNpHWNFeqHKyjs2UhXPWF4lqoC7/EQDFYdFqyVqT0CVMl6sD6CGmGrklm3HnDgQ
OgeSoUOOEbyg3q01XrR5mHzubDFt8nY7EA7+AEdSxXBXbHxx7MPtqZGlYUg1dQLi
l7R7mQHykxKBwWRFqn5RoQFO+xIc0jYt+agKpZ86k+GbXxPcNsYPfs7lzyZcS3c6
AACd6gaZGLO6Z3T+pdElVvvqqG247tpLJDmYSyRC2T/544e25VDnhiwH8ntQZlMU
=yyqJ
-----END PGP PUBLIC KEY BLOCK-----
```