A Trilemma in AMM Mechanism Design

Yuhao Li$^{1[0000-0001-5281-8742]}$, Elaine Shi$^{2[0000-0002-5605-1048]}$, and Mengqian Zhang$^{2[0000-0003-0247-6152]\star}$

[1] Columbia University, New York, USA `yuhaoli@cs.columbia.edu`
[2] Carnegie Mellon University, Pittsburgh, USA
`{elaineshi,mengqianzhang}@cmu.edu`

Abstract. Blockchains have popularized the Automated Market Makers (AMMs), where users trade crypto-assets directly with a smart contract, governed by a pricing function embedded in the contract's code. Today, users of AMMs are often forced to accept unfavorable prices due to widespread front-running and back-running attacks, commonly known as Miner Extractable Value (MEV). Several earlier works show impossibility results suggesting that completely removing MEV at the consensus layer is impossible, partly because the consensus layer is agnostic of application-level semantics. For this reason, more recent works have advocated mechanism design approaches at the application (i.e., smart contract) level.
We study a natural two-asset AMM mechanism design problem recently initiated and explored in prior work by Chan, Wu, and Shi, in which they proposed a mechanism that satisfies a surprisingly strong notion of incentive compatibility (IC), under the consensus assumption that the underlying blockchain provides sequencing fairness.
In this paper, we investigate the (in)feasibility of simultaneously achieving IC and other desirable properties such as weak local efficiency (wLE) and uniform pricing (UP). At a high level, wLE requires that the mechanism should not leave any unfulfilled demand from users whose asking prices are not overly restrictive, and whose orders could have been executed directly against the pool. UP requires that all orders that get (partially) executed must trade at the same exchange rate.
We unveil the underlying mathematical structure of AMM mechanism design, and our main results can be summarized as a trilemma-style theorem: among the desirable properties IC, wLE, and UP, any two out of three are possible, but no mechanism can satisfy all three.

Keywords: Decentralized Mechanism Design · MEV · Incentive Compatibility · Uniform Pricing · Economic Efficiency.

* Author ordering is randomized. See here.

1 Introduction

Blockchains and cryptocurrencies have enabled decentralized finance (DeFi), with Automated Market Makers (AMMs) [5] being the most popular DeFi application today. As of March 2021, the total crypto assets held by the top six AMMs (including Uniswap and Balancer) was valued at $15 billion [26].

An Automated Market Maker (AMM) is a smart contract backed by a decentralized blockchain. In a standard two-asset AMM mechanism, the contract maintains a liquidity pool (henceforth called the *pool* for short) containing two crypto-assets denoted X and Y respectively. Users can trade with the pool based on the pricing function defined by the smart contract. At any point in time, the pricing function determines an exchange rate for the two assets based on supply and demand. For example, a widely adopted rule is the *constant-product potential function* defined as follows. Let $\mathsf{Pool}(X, Y)$ denote the pool's state meaning that the pool holds $X \geq 0$ and $Y \geq 0$ units of each asset, respectively[3]. A constant product potential requires that $X \cdot Y = C$ for some constant $C > 0$. This means that if a user buys x amount of X from the pool, it needs to pay $-y$ amount of Y such that $(X - x)(Y - y) = C$. Observe that under such a pricing curve, the price of a crypto-asset will rise as more units are purchased from the pool.

The rapid adoption of DeFi applications such as AMMs has led to widespread incentive attacks often referred to as Miner Extractable Value (MEV). Specifically, since users submit their orders in the clear, arbitrageurs can easily launch front-running and back-running attacks to make risk-free profit while forcing the ordinary user to suffer from the worst-possible price [6, 14, 18, 20, 21, 29]. Such front-running and back-running attacks are exacerbated if the arbitrageur colludes with a block producer (also called the miner[4]) who has unilateral control over the next block's contents as well as the sequencing of transactions within the block. MEV-style incentive attacks are harmful in multiple dimensions. Not only do they exploit the victim users, but also undermine the stability of the underlying consensus ecosystem. Specifically, major block producers today have private contracts with arbitrageurs and ordinary users alike, offering favorable positions in the block to them at a price. This has in turn caused the blockchain's ecosystem to evolve towards a high degree of centralization — a recent measurement study showed that more than 85% of the blocks today are built by the top two block producers [27].

These negative externalities of MEV have been widely recognized, and the blockchain community has made it their top priority to mitigate MEV. Unfortunately, several recent works [4, 12] have shown impossibility results that can be interpreted to mean that "complete removal of MEV at the consensus layer (subject to today's architecture) is impossible". Partly this is because the consensus

[3] Whenever the context is clear, we overload the notation X and Y to mean the *quantity* of the two crypto-assets held by the pool.

[4] In this paper, we use the terms "miner" and "block producer" interchangably. Our results do not care whether the underlying consensus protocol is proof-of-work or proof-of-stake.

layer is agnostic of the application semantics of the smart contracts, and yet the utility of any transaction can be an arbitrary function of the blockchain's state (which is why MEV exists in the first place). As a result, more recent works have advocated a combined "consensus + application"-level approach towards mitigating MEV. The idea is for the consensus layer *not* to completely eliminate MEV, but to provide certain desirable properties that lend themselves to MEV mitigation. Then, the application layer (i.e., smart contract layer) can take advantage of such properties in mechanism design to achieve provable notions of MEV resilience.

In particular, the recent work of Chan, Wu, and Shi [9] exemplifies this design principle, where they gave a formal treatment of AMM mechanism design and defined a notion of MEV resilience called *incentive compatible*. Roughly speaking, an AMM mechanism is said to be incentive compatible (IC), if every user (or miner) maximizes its profit by truthfully reporting its intrinsic valuation and demand, and no strategic move or manipulation will lead to positive gains in utility. Because an arbitrageur (possibly colluding with the miner) can be viewed as a special user with zero intrinsic demand, the IC notion of [9] implies arbitrage resilience, that is, no one can make risk-free profit. Chan et al. then showed that if the consensus layer offers *weak sequencing fairness*, one can indeed design an AMM mechanism that satisfies IC. Specifically, the weak fair-sequencing model is meant to capture a new generation of consensus protocols with a *decentralized sequencer* [2,3,15–17], who sequences the transactions based on their (approximate) time of arrival. We stress that this assumption itself does not automatically eliminate MEV or prevent front-running, since a strategic player can still insert orders dependent on others' orders, and even race and preempt the victims' orders if it has a faster underlying network — this is also why the feasibility results in [9] are interesting and non-trivial.

In this paper, we revisit the AMM mechanism design problem with the goal of understanding the composability of incentive compatibility (IC) with other desirable properties. More specifically, we ask the following questions:

- *Is IC at odds with the efficiency of the mechanism?* The mechanism of [9] fails to offer a natural notion of efficiency for orders that demand a more stringent price than the initial market price of the AMM pool denoted by r_0. For example, consider a user who wishes to buy asset X at a maximum price $r < r_0$. In Chan et al.'s mechanism, this user's order will be ignored even if after executing the batch, the pool's ending price is actually lower than r. Therefore, a natural question is whether we can have an IC mechanism with better efficiency.
- *Can we get IC by offering uniform pricing (UP) to the entire batch of orders?* Uniform pricing (UP) has been adopted in many academic publications [7,8,22,23,28] as well as real-world AMM mechanisms [1]. UP is often regarded as a desirable property: not only does it provide fair pricing to all users within the same batch, but it also mitigates MEV by eliminating internal arbitrage and sandwich attacks. However, looking more broadly — and especially in our context, where IC is intended to provide a very strong

incentive guarantee — internal arbitrage elimination can be interpreted as a relatively weak notion: it only prevents those strategic players *with zero intrinsic demand* from making risk-free profit. This, in particular, still permits a user or miner with non-trivial valuation and demand to profit from strategic deviations (see concrete manipulations by strategic users in Section 4.3 and by miners in [28]), which is precisely one of the misbehaviors that IC intends to preclude. Therefore, a natural question is whether UP also lends itself to achieving incentive compatibility.

1.1 Our Results and Contributions

To answer the above questions, we further explore the design space to understand the tension among the various desirable properties. We prove several new feasibility and infeasibility results that jointly characterize the price of IC in AMM mechanism design.

IC precludes strong notions of efficiency. First, we show that incentive compatibility (IC) conflicts with strong notions of efficiency including the standard notion of Pareto optimality (PO) and a more relaxed notion called local efficiency (LE). Specifically, PO means that there should not exist an alternative legal outcome that makes someone strictly happier while leaving everyone else at least as happy as the mechanism's outcome. In other words, at the end of the mechanism, no Pareto improvement can be made for any user or any group of users to incrementally trade among themselves or trade with the pool. LE is a relaxation of PO requiring that the mechanism should not leave any unfulfilled demand that could have been satisfied by trading with the pool at a price desired by the user. Our result is stated in the following theorem:

Theorem 1 (IC precludes LE or PO). *No AMM mechanism can simultaneously achieve IC and LE (or PO), and this impossibility holds even in the weak fair-sequencing model.*

Having established that even LE is too strong to be compatible with IC, it is clear that some notion of efficiency weaker than LE needs to be introduced. Recall that Chan et al. [9] introduced a neat mechanism under the weak fair-sequencing model that explicitly satisfies IC. Their mechanism also implicitly satisfies a natural notion of efficiency that is slightly weaker than local efficiency.[5] We refer to it as weak local efficiency (wLE). In comparison with LE, wLE cares about achieving local efficiency only for "reasonable" orders — those that do not insist on an exchange rate more stringent than the initial market price.

A trilemma among IC, wLE, and UP. We next ask whether uniform pricing (UP) aids the design of incentive-compatible mechanisms with meaningful notions of efficiency. Since we have established wLE as a meaningful efficiency notion

[5] Our Theorem 1 provides a mathematical justification for why [9] only achieved this weaker notion of efficiency.

compatible with IC, we refine the question and ask whether it is possible to achieve IC, wLE, and UP simultaneously. We prove a trilemma-style theorem, stating that one can choose any two out of these three properties, but it is impossible to satisfy all three at the same time. This trilemma theorem can be interpreted to mean that perhaps somewhat counterintuitively, although UP naturally eliminates internal arbitrage, it is actually somewhat at odds with the stronger notion of IC — asking for both UP and IC will result in only very inefficient mechanisms. The intuition why UP does not lend to IC (contrary to common belief) will be explained in more detail in Section 4.3 where we give a natural mechanism that is UP + wLE but not IC.

Our trilemma result is stated in the following theorem:

Theorem 2 (Trilemma among IC, wLE, and UP). *No AMM mechanism can simultaneously achieve IC, wLE, and UP, and this impossibility holds even in the weak fair-sequencing model. On the other hand, it is feasible to achieve wLE + UP or IC + UP in the plain model (i.e., without the weak fair-sequencing assumption), and it is feasible to achieve IC + wLE in the weak fair-sequencing model.*

Clearly, the main open question left by Theorem 2 as well as [9] is whether we can achieve IC + wLE in the plain model, without the weak fair-sequencing assumption. See more discussion on this in Section 5.

Additional results. While our paper is centered around understanding the price of IC, we also explore the tension between UP and LE. We prove an impossibility result showing that no AMM mechanism can simultaneously achieve UP and LE. This demonstrates that LE is a very stringent notion from a different angle — not only is it incompatible with IC, but also incompatible with UP. This again justifies that it is natural to relax LE to wLE, which allows us to overcome this impossibility since Theorem 2 implies the feasibility of UP + wLE.

1.2 Additional Related Work

Verifiable sequencing rules. A line of work has explored mitigating MEV at the application layer. The model we adopt follows directly from Chan et al. [9], which in turn drew inspiration from the elegant work of Ferreira and Parkes [12]. Notably, Chan et al. relax several stringent requirements of Ferreira and Parkes to circumvent impossibility results. While Ferreira and Parkes describe their approach as enforcing "verifiable sequencing rules" at the consensus layer to mitigate MEV, it is in fact more desirable to view their sequencing rules as being enforced by the smart contract application — ideally the consensus layer should be agnostic of application-specific semantics. However, Ferreira and Parkes's model differs in nature from ours and that of Chan et al., since they impose a couple of restrictive constraints: the orders must be fulfilled one after another, and moreover, the pool's state must respect the potential function after executing *every* order, not just at the end of the batch. These requirements lead to very strong impossibility results as demonstrated by Ferreira and Parkes: not only is IC impossible in their model, but even the weaker notion of "arbitrage free" is impossible in their setting. Subsequently, Li et al. [19] studied the miner's profit-maximizing strategy under the greedy sequencing rule proposed in [12], and the implications for users when the miner adopts the optimal strategy.

Batch clearing at uniform price. Several works have explored the idea of batch clearing at a uniform price [1,7,8,22,23]. Uniform pricing is a desirable property since it eliminates internal arbitrage, as well as well-known sandwich attacks. Nevertheless, Zhang et al. [28] recently observed that miners can still extract value from batch auctions and investigated profit-maximizing strategies for miners.

Transaction fee mechanism design. A recent line of work on transaction fee mechanism (TFM) design [10,11,13,24,25] explores how to design the blockchain space auction. However, so far, this line of work is agnostic of application-level MEV, since they fail to capture general utility functions that may depend on transaction sequencing. Bahrani et al. [4] showed strong impossibility results for solving the full spectrum of application-level MEV solely at the TFM layer. In this sense, application-level mechanism design [9, 12] as well as our work complement the line of work on TFM design by explicitly capturing application-dependent semantics and utility functions.

2 Model

2.1 AMM Preliminaries

Order. Each order is of the form (t, r, q, α) where

- the first field $t \in \{\mathsf{Buy}(X), \mathsf{Buy}(Y), \mathsf{Sell}(X), \mathsf{Sell}(Y)\}$ specifies the type of the order;
- the second field $r > 0$ specifies the worst exchange rate the user is willing to tolerate. Specifically, the rate r is expressed in terms of the price of each unit of X, measured in units of Y. For a $\mathsf{Buy}(X)$ or $\mathsf{Sell}(Y)$ order, r specifies the maximum units of Y the user is willing to pay in exchange for one unit of X. For a $\mathsf{Buy}(Y)$ or $\mathsf{Sell}(X)$ order, r specifies the minimum units of Y the user wants to get for selling each unit of X;
- the third field q denotes the maximum number of units the user wants to trade. For example, for a $\mathsf{Sell}(X)$ order, it means that the user wants to sell at most q units of X; and for a $\mathsf{Buy}(Y)$ order, it means that the user wants to buy at most q units of Y.
- the last field α is used to encode arbitrary auxiliary information that the mechanism may take into account, e.g., identity information, timestamp of the order, and so on.

AMM and potential function. There is a pool whose state is denoted by a pair (X, Y) where $X > 0$ and $Y > 0$ denote the amount of the assets X and Y remaining in the pool. Without risking ambiguity, we may use X and Y to denote the pool state as well as to name the assets.

The pricing is defined through a potential function $\Phi(\cdot, \cdot)$ such that if (X, Y) and (X', Y') are both valid pool states, it must be that $\Phi(X, Y) = \Phi(X', Y')$. In other words, if the initial pool state is (X, Y), then to buy x units of X from the pool, the payment in Y, denoted $-y$, must satisfy $\phi(X - x, Y - y) = \Phi(X, Y)$. As a concrete example, the *constant-product function* $\Phi(X, Y) = X \cdot Y$ is most commonly adopted in Uniswap contracts.

It is standard to assume that $\Phi(X, Y)$ is *increasing*, *differentiable*, and *concave* [9, 12, 19], which implies the following:

- *Bijective mapping.* Fix some initial pool state (X_0, Y_0) and let $C := \Phi(X_0, Y_0) > 0$. For every $X > 0$, there is a unique $Y := Y(X)$ such that $\Phi(X, Y) = C$. Therefore, for convenience, we can view $Y(X)$ as a function of X, whenever C is clear from the context.
- *No free lunch.* $Y(X)$ is a strictly decreasing function in X, that is, $\forall 0 < X_0 < X_1$, $Y(X_0) > Y(X_1)$. In other words, to buy a positive amount of X from the pool, one must pay a positive amount of Y to the pool.
- *Increasing marginal cost.* Suppose $X_0 < X_1$, then $-\frac{dY}{dX}(X_0) > -\frac{dY}{dX}(X_1)$. In other words, the marginal cost of X increases as one purchases more X from the pool.
  For convenience, we also refer to $-\frac{dY}{dX}(X_0)$ as the *market rate* when the pool has X_0 units of X, i.e., the price per unit of X for purchasing an infinitesimally small amount of X from the pool.

AMM mechanism. We consider *deterministic* AMM mechanisms defined in [9]. The mechanism receives a batch of orders, and *fully* or *partially* executes a subset of the orders. For each order in the batch, its outcome is denoted (x, y) which means its net gain in X and Y, respectively. A negative value of x (or y) means a loss in the asset X (or Y). The mechanism should satisfy the following well-formedness properties:

- *Reasonable fulfillment.* For a $\mathsf{Buy}(X)$-type orders with quantity q, it must be $0 \leq x \leq q$. For a $\mathsf{Buy}(Y)$-type orders with quantity q, it must be $0 \leq y \leq q$. For a $\mathsf{Sell}(X)$-type orders with quantity q, it must be $0 \leq -x \leq q$. For a $\mathsf{Sell}(Y)$-type orders with quantity q, it must be $0 \leq -y \leq q$. In other words, no order should fulfill more than the desired quantity or fulfill a negative amount (i.e., fulfill in the opposite direction of buy or sell).
- *No free lunch.* Either $x \cdot y < 0$, or $x = y = 0$. In other words, no order should strictly gain in one type of asset without losing in the other.
- *Individual rationality.* For a $\mathsf{Buy}(X)$ or $\mathsf{Sell}(Y)$-type order, it must be that $-\frac{y}{x} \leq r$. For a $\mathsf{Buy}(Y)$ or $\mathsf{Sell}(X)$-type order, it must be that $-\frac{y}{x} \geq r$. In other words, the executed exchange rate should be no worse than the rate specified in the order.

– *Conformant to pricing function.* After execution, let x_{tot} and y_{tot} denote the sum of all users' net gain in X and Y, respectively. The end pool state $(X', Y') := (X - x_{\text{tot}}, Y - y_{\text{tot}})$ must satisfy the potential function $\Phi(X', Y') = \Phi(X, Y)$.

Like [9], our definition only requires that the potential function be respected *before and after executing the entire batch.* In comparison, the earlier work of Ferreira and Parkes [12] works in a more draconian model in which the orders are fulfilled one after another, and the pool's state must satisfy the potential function after executing *every* order. For this reason, the impossibility results in [12] are not applicable to our setting.

2.2 Strategy Space

We assume that each user has an *intrinsic type* $T := (t, r, q, \alpha)$ sharing the same form as an order. For a direct revelation mechanism, the honest user strategy is to *truthfully report* its intrinsic type.

For defining the strategy space, we consider two models called the plain model and the weak fair-sequencing model respectively introduced in [9].

Plain model. The plain model is meant to capture mainstream consensus protocols today where the block producer (who can be a strategic player in our mechanism) can fully control the block contents and the sequencing of transactions in the block.

In the plain model, we assume that a strategic user (or miner) with intrinsic type (t, r, q, α) may adopt one or more of the following strategies:

– Post zero or multiple arbitrary orders which may or may not reflect its intrinsic type — this captures strategies that involve misreporting valuation and demand, as well as posting of fake orders;
– Censor honest users' orders — this captures a strategic miner's ability to exclude certain orders from the block;
– Arbitrarily misrepresent its own auxiliary information field α, or even modify the α field of honest users' orders — meant to capture the miner's ability to decide the sequencing of transactions within a block, where the arrival-time and position information may be captured by the auxiliary information field α.
– Decide its strategy *after* having observed honest users' orders.

Weak fair-sequencing model. We will also work in a weak fair-sequencing model defined in [9], meant to capture a new generation of consensus protocols that employ a decentralized sequencer and rank orders based on their (approximate) arrival times [2,3,15–17]. We stress that even in the weak fair-sequencing model, it is possible for a strategic user to observe a victim's order, post a dependent order, and have the dependent order race against and front-run the victim's order. Such a front-running attack can succeed especially when the strategic user's network is faster than the victim's. In particular, we stress that the weak fair-sequencing model is sequencing orders based on their *arrival times, not the time of submission of these orders* — for this reason, front-running is still possible in this model.

Recall that a user's intrinsic *type* has the form (t, r, q, α) where (t, r, q) denotes the user's true valuation and budget. In the weak fair-sequencing model, we use the α field to denote the order's arrival time — a smaller α means that the user arrives earlier.

We shall assume that under honest strategy, a user's order should always have the correct α whose value is determined by nature, and equal to the time at which the order is generated plus the user's network delay. A strategic user is allowed to delay the submission of its order. More specifically, a strategic user or miner can adopt the following strategies in the weak fair-sequencing model:

– A strategic user or miner with intrinsic type (t, r, q, α) is allowed to post zero or multiple bids of the form $(\_, \_, \_, \alpha')$ as long as $\alpha' \geq \alpha$. This captures misreporting valuation and demand, posting fake orders, as well as delaying the posting of one's orders.
– The strategic user or miner can decide its strategy *after* observing honest users' orders.

Compared to the plain model, the weak fair-sequencing model imposes certain restrictions on the strategy space. Specifically, in the plain model, a strategic user or miner can arbitrarily modify the α field of its own order or even others' orders, and a strategic miner may censor honest users' orders. In the weak fair-sequencing model, a strategic user or miner can no longer under-report its α, cannot modify honest users' α, and cannot censor honest users' orders, because the sequencing of the transactions is determined by the underlying decentralized sequencer. Importantly, despite these constraints on the strategy space, the weak fair-sequencing model still permits front-running attacks as mentioned earlier, and thus mechanism design remains non-trivial even under this restricted strategy space.

2.3 Utility Ranking

Following the definitional paradigm of [9], we define utility as a partial ranking over outcomes, which expresses a user's preferences among the different possible outcomes.

Recall that we use a pair (x, y) to denote the outcome of a user's order, indicating a net gain of x in X, and a net gain of y in Y (where a net loss is captured as negative gain). Consider two outcomes (x_0, y_0) and (x_1, y_1). The following rules define a natural partial ordering of preferences among outcomes for a user with intrinsic type $T = (\mathsf{Buy}(X), r, q, \_)$:

1. If $x_0 \leq x_1, y_0 \leq y_1$, then $(x_0, y_0) \preceq_T (x_1, y_1)$. In other words, the latter outcome is at least as good as the former, if the latter outcome gains at least

as much as the former in both assets. Further, if at least one $\leq$ is replaced with $<$, then we say that $(x_0, y_0) \prec_T (x_1, y_1)$, i.e., the user strictly prefers the latter outcome.

2. If $x_0 \leq x_1 \leq q$ (or $q \leq x_1 \leq x_0$), and $r \cdot (x_1 - x_0) \geq y_0 - y_1$, then $(x_0, y_0) \preceq_T (x_1, y_1)$. In other words, the latter outcome is at least as good as the former one, if it is closer to satisfying the demand q, and moreover, the user paid at most r *marginal* price for each *extra* unit of X in the latter outcome. Moreover, if $r \cdot (x_1 - x_0) \geq y_0 - y_1$ is replaced with strict inequality, that is, $r \cdot (x_1 - x_0) > y_0 - y_1$, then we say $(x_0, y_0) \prec_T (x_1, y_1)$, i.e., the latter outcome is strictly preferred.
3. The standard transitivity rule holds for $\preceq_T$ and $\prec_T$.

For other types including $\mathsf{Buy}(Y)$, $\mathsf{Sell}(X)$, and $\mathsf{Sell}(Y)$, a partial ordering can be symmetrically defined — the full definition can be found in the full version.

Remark 1 (Why define utility as a partial ordering). As explained in [9], the reason for defining utility as a partial ordering rather than a numerical value is to explicitly respect the intentions of users' intrinsic types.

For example, if a user's intrinsic type is to buy up to q units of X at a maximum price of r. Suppose in one outcome, this user fulfills his demand of q paying $-y$ units of Y. In another outcome, the user buys $q+1$ units of X paying $(-y) + r/2$ units of Y. Then, the latter outcome is incomparable to the former one, because the user obtained the extra unit (which overshoots the demand) but at a favorable marginal price of $r/2$.

In another example, the strategic player could be the miner, who has no intrinsic demand for the assets. However, the miner might be able to get an outcome (x, y) where $x > 0$ and $y = 0$, by for example sandwich attacks. Then this outcome (x, y) is strictly better than the outcome $(0, 0)$.

2.4 Desirable Properties of an AMM Mechanism

Incentive compatibility. In the definition below, we use $HS(T)$ to denote the honest strategy of a user with intrinsic type T — for a direct-revelation mechanism, the honest strategy is simply to reveal the user's true type. Further, we use $\mathsf{out}^u(X_0, Y_0, \mathbf{b})$ to denote the outcome of user u when the mechanism is executed over initial pool state $\mathsf{Pool}(X_0, Y_0)$, and a vector of orders $\mathbf{b}$.

Definition 1 (Incentive compatibility (IC) [9]). *Given an AMM mechanism, we say that it satisfies IC (w.r.t. some partial ordering relation $\preceq_T$), iff for any initial pool state $\mathsf{Pool}(X_0, Y_0)$, for any vector of orders $\mathbf{b}_{-u}$ belonging to all other users except u, for any intrinsic type T of the strategic player u, for any possible strategic order vector $\mathbf{b}'$ of the player u, either $\mathsf{out}^u(X_0, Y_0, \mathbf{b}_{-u}, HS(T)) \succeq_T \mathsf{out}^u(X_0, Y_0, \mathbf{b}_{-u}, \mathbf{b}')$ or $\mathsf{out}^u(X_0, Y_0, \mathbf{b}_{-u}, HS(T))$ and $\mathsf{out}^u(X_0, Y_0, \mathbf{b}_{-u}, \mathbf{b}')$ are incomparable w.r.t. $\preceq_T$.*

More intuitively, the definition says that *no strategic play can result in a strictly better outcome than the honest strategy.* Note that the "strategic player

u" above could be either a strategic user or miner, and the boldface $\mathbf{b}'$ above is intended to capture the possibility of submitting multiple orders (i.e., Sybil attacks).

Efficiency properties. We may want the mechanism to enjoy certain efficiency properties. First, a very natural notion is Pareto optimality (PO) defined below, where preferences among outcomes are based on the partial ordering notions.

Definition 2 (Pareto optimal (PO)). *We say that an AMM mechanism is Pareto optimal iff the following holds: for any initial pool state and any set of users with arbitrary true types, there does not exist another legal outcome in which some user's outcome is strictly better than the mechanism's outcome, while every other user's outcome is at least as good as the mechanism's outcome.*

PO can also be interpreted to mean that the users cannot achieve a Pareto improvement by further trading among themselves or trading with the pool. We next define a relaxed notion of efficiency, which requires that the mechanism should not leave any unfulfilled demand that could have been satisfied through trading with the pool at a price desired by the user.

Definition 3 (Locally efficient (LE)). *An AMM mechanism is said to be locally efficient, iff after execution, the ending exchange rate is no smaller than the rate of any* $\mathsf{Buy}(X)$ *or* $\mathsf{Sell}(Y)$ *order that is not completely fulfilled, and no larger than the rate of any* $\mathsf{Buy}(Y)$ *or* $\mathsf{Sell}(X)$ *order that is not completely fulfilled.*

We note that LE is naturally desirable in the context of AMMs, since if, after the mechanism executes, some users still wish to continue trading at the pool's ending state, the mechanism has failed to realize obvious gains in social efficiency, which the notion of LE is intended to preclude.

We next define a further relaxed notion of efficiency called weak local efficiency, which places the same requirements as LE but only for "reasonable" users who do not demand a price more stringent than the initial market price.

Definition 4 (Weakly locally efficient (wLE)). *A* $\mathsf{Buy}(X)$ *or* $\mathsf{Sell}(Y)$ *order is said to be eligible iff its rate is no less than the initial market price. Similarly, a* $\mathsf{Sell}(X)$ *or* $\mathsf{Buy}(Y)$ *order is said to be eligible iff its rate is no higher than the initial market price. An AMM mechanism is said to be weakly locally efficient (wLE) iff the same conditions of LE are guaranteed, but now only for eligible orders.*

The following fact follows directly from the definitions.

Fact 1 *PO* $\Longrightarrow$ *LE* $\Longrightarrow$ *wLE.*

Definition 5 (Uniform pricing (UP)). *All orders that get (partially) executed, no matter the type, must all trade at the same exchange rate.*

3 Impossibility Results

We first prove that IC precludes LE.

Theorem 3 (IC + LE $\Longrightarrow$ impossible). *No AMM mechanism can simultaneously satisfy incentive compatibility and local efficiency. Further, this impossibility holds even in the weak fair-sequencing model.*

Remark 2. Chan et al. [9] showed only feasibility results, and therefore they allowed the strategic player to arbitrarily modify anyone's auxiliary information field α in the plain model. This make their feasibility result (for the weaker arbitrage resilience property) stronger. Because we are proving an impossibility result, it will strengthen the result if we make the strategy space in the plain model more restricted. In practice, one natural scenario is that the the auxiliary information field α encodes the user's cryptographic identity such as public key. In this case, although the strategic player cannot arbitrarily modify honest users' public keys, the above impossibility still holds: suppose that honest users sample their keys at random, then each time the strategic player samples a random key, there is a 1/2 probability that it will be favored in the tie-breaking. Therefore, the strategic player can simply perform rejection-sampling till it finds a key that is favored in the tie-breaking. The expected number of trials needed is 2.

We also prove that UP and LE are not compatible — since this impossibility does not involve IC, it holds regardless of the strategy space.

Theorem 4 (UP + LE $\Longrightarrow$ impossible). *No AMM mechanism can simultaneously satisfy uniform pricing and local efficiency.*

Finally, we show that it is not possible to have a mechanism satisfying all three properties simultaneously. Our impossibility proof only uses two $\mathsf{Buy}(X)$ orders, thus for this special case LE and wLE are equivalent.

Theorem 5 (IC + UP + wLE $\Longrightarrow$ impossible). *No AMM mechanism can simultaneously satisfy incentive compatibility, uniform pricing, and weak local efficiency. This impossibility holds as long as the strategic player is allowed to misreport its valuation and quantity, that is, the impossibility holds even in the weak fair-sequencing model, and even in a permissioned model where fake bids are not allowed.*

All the proofs can be found in the full version.

4 Feasibility of Satisfying Any Two Properties

In this section, we demonstrate that for any two properties among IC, wLE, and UP, there exists a mechanism that satisfies them. We examine each pair separately below.

4.1 IC + wLE

Chan, Wu, and Shi [9] proposed a neat mechanism that satisfies IC and wLE in the weak fair-sequencing model. We stress that all our impossibility results (Theorem 3, Theorem 4, Theorem 5) hold in the weak fair-sequencing model. An important open question left is whether we can remove the fair-sequencing assumption and achieve IC + wLE in the plain model.

4.2 IC + UP

Note that if one is only interested in incentive compatibility and uniform pricing, then the feasibility is trivial since we didn't specify the efficiency guarantee and a naive mechanism could always do nothing. However, our goal in this part is to show an alternative interesting mechanism that we have found, formally shown in Figure 1.

At a high level, Mechanism 1 works as follows: Ignore all $\mathsf{Sell}(X)$ or $\mathsf{Buy}(Y)$ orders. The remaining orders are either $\mathsf{Buy}(X)$ or $\mathsf{Sell}(Y)$. Now, rank the orders based on their declared exchange rate, and let r_1 and r_2 denote the highest and the second-highest exchange rate, respectively. Let x be the number of units of X that must be taken from the pool such that the average price paid is equal to r_2. If the first user can absorb x units of X without exceeding its declared quantity or budget, then allocate x units of X to the first user, and allocate 0 to everyone else. Otherwise, allocate 0 to everyone.

The nice property of Mechanism 1 is that it is a non-trivial mechanism that always satisfies IC and UP. Furthermore, in certain cases, it also achieves wLE.

Theorem 6. *Mechanism 1 satisfies IC and UP. Furthermore, if there are only* $\mathsf{Buy}(X)$ *and* $\mathsf{Sell}(Y)$ *orders, and the user with the highest exchange rate has a sufficiently large quantity or budget, then the mechanism also achieves wLE.*

4.3 UP + wLE

Simple case: all orders are the same type. If all orders are the same type — take $\mathsf{Buy}(X)$ as an example — then there is a simple mechanism that satisfies UP + wLE. Basically, let $r_{\text{avg}}(x)$ denote the average price per unit for purchasing x units of X from the pool. Because of the increasing marginal cost property (see Section 2.1), $r_{\text{avg}}(x)$ is an increasing function in x. Further, as x goes to 0, $r_{\text{avg}}(x)$ goes to the initial market price.

Suppose we are given a batch of orders $\{t_i, r_i, q_i, \_\}_i$ where we may assume that every r_i is at least the initial market price (otherwise, we simply discard the order). Let $D(r) := \sum_i \mathbb{1}(r_i \geq r) \cdot q_i$ denote the total demand that must be fulfilled to respect wLE at any rate $r \geq 0$, where $\mathbb{1}(\cdot)$ is the indicator function. The mechanism finds the $x^* \in [0, \sum_i q_i]$ such that $D(r_{\text{end}}(x^*)) = x^*$, and buys x^* units of X from the pool and allocates them to the orders whose declared rate is at least $r_{\text{end}}(x^*)$. In particular, all orders whose declared rate is strictly larger than $r_{\text{end}}(x^*)$ will be fully fulfilled, and the orders whose declared rate is

Input: The initial pool state (X_0, Y_0), the potential function Φ, and a set of orders $\mathbf{b}$. Since the mechanism does not make use of the auxiliary information field α, each order is simply represented as a tuple (t, r, q).

Output: A net gain in assets X and Y for each order.

Mechanism:

1. Discard all $\mathsf{Sell}(X)$ or $\mathsf{Buy}(Y)$ orders.
2. Discard all $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$ orders whose declared exchange rate $r < -\frac{dY}{dX}(X_0)$.
3. If no orders remain, then do nothing. Otherwise, go to the next step.
4. Rank the remaining orders in descending order of their declared exchange rate. Let (t_1, r_1, q_1) be the order with the highest exchange rate and (t_2, r_2, q_2) the order with the second-highest exchange rate. If multiple orders share the highest exchange rate, which implies $r_1 = r_2$, rank them arbitrarily. If only one order remains, set $r_2 = -\frac{dY}{dX}(X_0)$.
5. Let x be such that $\frac{Y(X_0-x)-Y(X_0)}{x} = r_2$. Let x' be such that $-\frac{dY}{dX}(X_0-x') = r_1$.
   (a) If $t_1 = \mathsf{Buy}(X)$ and $q_1 \geq x$, allocate $x^* = \max(x, \min(x', q_1))$ units of X to the first user (the one with rate r_1), and charge $Y(X_0-x^*)-Y(X_0)$ units of Y. All other users receive zero and pay nothing.
   (b) If $t_1 = \mathsf{Sell}(Y)$ and $\Delta x := X(Y_0) - X(Y_0 + q_1) \geq x$, allocate $x^* = \max(x, \min(x', \Delta x))$ units of X to the first user (the one with rate r_1), and charge $Y(X_0-x^*)-Y(X_0)$ units of Y. All other users receive zero and pay nothing.
   (c) Otherwise, all users receive zero and pay nothing.

Fig. 1. Mechanism 1 satisfying IC + UP

exactly $r_{\text{end}}(x^*)$ might be partially fulfilled. Everyone pays a uniform per-unit price of $r_{\text{avg}}(x^*)$. It is not hard to see that both UP and wLE are guaranteed by design. It remains to argue that an x satisfying the above condition must exist — this can be seen from the combination of the following facts: (1) $D(r_{\text{end}}(x))$ is non-increasing in x; and (2) $D(\max_i \{r_i\} + \epsilon) = 0$ and $D(r_{\text{end}}(0)) = \sum_i q_i$.

We now explain why the above mechanism is not IC, which might be counterintuitive at first sight, because one might be tempted to think that the fair pricing offered by UP facilitates IC. Observe that under truthful reporting, every executed order i always receives positive utility, because their paid price is $r_{\text{avg}}(x^*) < r_{\text{end}}(x^*) \leq r_i$. Suppose that there is some user i with $r_i = r_{\text{end}}(x^*) - \epsilon_1$ and $q_i = \epsilon_2$. Following the description of the mechanism above, this order is not executed and the user gets zero utility under truthful reporting. However, the following strategy will allow i to get positive utility. Instead of truthful reporting, i slightly over-reports its valuation, allowing the order to get executed. As long as q_i is sufficiently small, the induced uniform price will be less than the order's true valuation, making its utility positive.

The more general case: all types of orders. Next, we extend the idea above to the general setting with all types of orders and present the complete mechanism in Figure 2.

Theorem 7. *Mechanism 2 satisfies UP and wLE.*

All the proofs can be found in the full version.

5 Summary and Future Directions

In this work, we reexamine the problem of AMM mechanism design with respect to the widely desired properties including IC, LE/wLE, and UP. Our main technical contribution is the establishment of a trilemma among IC, wLE, and UP. In addition, we also prove that no AMM mechanism can simultaneously achieve IC + LE or UP + LE.

The main open question left by this paper and [9] is whether there exists a mechanism that is IC and wLE in the plain model (without the weak fair-sequencing assumption). We conjecture that there is no such mechanism, which, if true, would imply the necessity of a consensus guarantee in AMM mechanism design. We view such an impossibility result as one of the most crucial justifications for the "consensus + application" paradigm for mitigating MEV. On the other hand, a positive answer to this question would yield an exceptionally strong result: the existence of an AMM mechanism that achieves both IC and wLE in the plain model. Note that we consider such a mechanism highly powerful since our notion of IC implies a very strong incentive guarantee with respect to a very general strategy space, including misreporting valuation and demand, splitting real orders, posting fake orders, censoring honest users' orders, and others. Given this "win-win" situation, we believe that studying this problem is of fundamental interest for future work.

On the feasibility side, a natural next step is to investigate mechanism design for multi-token AMMs. In particular, an important open question is whether the IC + wLE mechanism developed for the two-asset setting [9] can be generalized to multi-asset AMMs under the same weak fair-sequencing assumptions. Similarly, one may ask whether it is possible to construct a mechanism that satisfies UP + wLE or IC + UP in the multi-token setting. Exploring the extent to which these feasibility results can be generalized is a crucial step toward a deeper understanding of mechanism design for multi-token AMMs.

Speaking more broadly, it seems unlikely that either the consensus layer or the application layer alone can fully eliminate MEV. This makes combined "consensus + application" approaches particularly appealing. For example, the work [9] demonstrates that AMM mechanism design benefits substantially from even weak consensus guarantees — results that appear difficult to achieve otherwise. It is therefore worth pursuing this line of research in both directions. On the one hand, can weak consensus guarantees also yield improvements for other DeFi applications such as decentralized lending or stablecoins? On the other hand, can insights from DeFi mechanism design help abstract new desiderata that, in turn, advance the study of consensus itself?

Input: The initial pool state (X_0, Y_0), the potential function Φ, and a set of orders $\mathbf{b}$. Since the mechanism does not make use of the auxiliary information field α, each order is simply represented as a tuple (t, r, q).

Output: A net gain in assets X and Y for each order.

Mechanism:

1. Let $r_0 := -\frac{dY}{dX}(X_0)$ be the initial market price. Discard all $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$ orders whose declared exchange rate $r < r_0$, and discard all $\mathsf{Buy}(Y)/\mathsf{Sell}(X)$ orders whose declared rate $r > r_0$. Let $\mathbf{b}'$ be the remaining orders.
2. Let $\mathcal{D}$ denote the set of $\mathsf{Buy}(X)/\mathsf{Sell}(Y)$ orders in $\mathbf{b}'$ which *demands* X, and $\mathcal{S}$ denote the set of $\mathsf{Buy}(Y)/\mathsf{Sell}(X)$ orders in $\mathbf{b}'$ which *supplies* X. More specifically, we let
$$\begin{cases} \mathcal{D}_{>r^*} = \{(t,r,q) \in \mathbf{b}' \mid t = \mathsf{Buy}(X)/\mathsf{Sell}(Y) \text{ and } r > r^*\}, \\ \mathcal{D}_{=r^*} = \{(t,r,q) \in \mathbf{b}' \mid t = \mathsf{Buy}(X)/\mathsf{Sell}(Y) \text{ and } r = r^*\}, \\ \mathcal{S}_{<r^*} = \{(t,r,q) \in \mathbf{b}' \mid t = \mathsf{Buy}(Y)/\mathsf{Sell}(X) \text{ and } r < r^*\}, \\ \mathcal{S}_{=r^*} = \{(t,r,q) \in \mathbf{b}' \mid t = \mathsf{Buy}(Y)/\mathsf{Sell}(X) \text{ and } r = r^*\}. \end{cases}$$
For a set $A \in \{\mathcal{D}_{>r^*}, \mathcal{D}_{=r^*}, \mathcal{S}_{<r^*}, \mathcal{S}_{=r^*}\}$ of orders, denote their demand/supply quantity under an eligible price p by $Q(A,p) = \sum_{(t,r,q)\in A} \beta(t,r,q;p)$, where $\beta(t,r,q;p) = \begin{cases} q, & \text{if } t = \mathsf{Buy}(X)/\mathsf{Sell}(X); \\ q/p, & \text{if } t = \mathsf{Buy}(Y)/\mathsf{Sell}(Y). \end{cases}$
For any rate $r^* \neq r_0$ that corresponds to the pool state (X^*, Y^*), denote $\overline{p}(r^*) = \frac{Y^*-Y_0}{X_0-X^*}$ and $\Delta x(r^*) = X_0 - X^*$. Define $\overline{p}(r_0) = r_0$ and $\Delta x(r_0) = 0$.
3. Find $r^* > 0$ such that one of the following conditions is met:
   (a) $Q(\mathcal{D}_{>r^*}; \overline{p}(r^*)) \leq Q(\mathcal{S}_{<r^*} \cup \mathcal{S}_{=r^*}; \overline{p}(r^*)) + \Delta x(r^*) \leq Q(\mathcal{D}_{>r^*} \cup \mathcal{D}_{=r^*}; \overline{p}(r^*))$ *and* $r^* \geq r_0$;
   (b) $Q(\mathcal{S}_{<r^*}; \overline{p}(r^*)) \leq Q(\mathcal{D}_{>r^*} \cup \mathcal{D}_{=r^*}; \overline{p}(r^*)) - \Delta x(r^*) \leq Q(\mathcal{S}_{<r^*} \cup \mathcal{S}_{=r^*}; \overline{p}(r^*))$ *and* $r^* \leq r_0$.
4. If r^* satisfies condition (a) above, trade $Q(\mathcal{S}_{<r^*} \cup \mathcal{S}_{=r^*}; \overline{p}(r^*)) + \Delta x(r^*)$ units of X at the price $\overline{p}(r^*)$. Specifically, all orders in $\mathcal{S}_{<r^*}$, $\mathcal{S}_{=r^*}$, and $\mathcal{D}_{>r^*}$ are fully fulfilled; orders in $\mathcal{D}_{=r^*}$ are partially fulfilled such that the fulfilled quantity is exactly $Q(\mathcal{S}_{<r^*} \cup \mathcal{S}_{=r^*}; \overline{p}(r^*)) + \Delta x(r^*) - Q(\mathcal{D}_{>r^*}; \overline{p}(r^*))$. $\Delta x(r^*) \geq 0$ units of X are sold by the pool.
   If r^* satisfies condition (b) above, trade $Q(\mathcal{D}_{>r^*} \cup \mathcal{D}_{=r^*}; \overline{p}(r^*)) - \Delta x(r^*)$ units of X at the price $\overline{p}(r^*)$. Specifically, all orders in $\mathcal{D}_{>r^*}$, $\mathcal{D}_{=r^*}$, and $\mathcal{S}_{<r^*}$ are fully fulfilled; orders in $\mathcal{S}_{=r^*}$ are partially fulfilled such that the fulfilled quantity is exactly $Q(\mathcal{D}_{>r^*} \cup \mathcal{D}_{=r^*}; \overline{p}(r^*)) - \Delta x(r^*) - Q(\mathcal{S}_{<r^*}; \overline{p}(r^*))$. $-\Delta x(r^*) \geq 0$ units of X are bought by the pool.

Fig. 2. Mechanism 2 satisfying UP + wLE

Acknowledgments

This work is in part supported by NSF awards 2212746, 2044679, a Packard Fellowship, a gift from the Ethereum Foundation, and a generous gift from the late Nikolai Mushegian.

References

1. https://cow.fi/cow-protocol.
2. Challenging periods reimagined: The key role of sequencer decentralization. https://ethresear.ch/t/challenging-periods-reimagined-the-key-role-of-sequencer-decentralization/15110.
3. The espresso sequencer. https://hackmd.io/@EspressoSystems/EspressoSequencer.
4. Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden. Transaction Fee Mechanism Design in a Post-MEV World. In *6th Conference on Advances in Financial Technologies, AFT 2024, September 23-25, 2024, Vienna, Austria*, volume 316 of *LIPIcs*, pages 29:1–29:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.
5. Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch-Lafuente. A theory of Automated Market Makers in DeFi. *CoRR*, 2021.
6. Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch Lafuente. Maximizing Extractable Value From Automated Market Makers. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*, 2022.
7. Andrea Canidio and Robin Fritsch. Batching Trades on Automated Market Makers. In *AFT*, 2023.
8. Andrea Canidio and Robin Fritsch. Arbitrageurs' profits, LVR, and sandwich attacks: batch trading as an AMM design response. 2024.
9. T-H. Hubert Chan, Ke Wu, and Elaine Shi. Mechanism Design for Automated Market Makers. In *AFT*, 2025.
10. Hao Chung, Tim Roughgarden, and Elaine Shi. Collusion-Resilience in Transaction Fee Mechanism Design. In *EC*, pages 1045–1073. ACM, 2024.
11. Hao Chung and Elaine Shi. Foundations of Transaction Fee Mechanism Design. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3856–3899. SIAM, 2023.
12. Matheus Venturyne Xavier Ferreira and David C. Parkes. Credible Decentralized Exchange Design via Verifiable Sequencing Rules. In *STOC*, 2023.
13. Aadityan Ganesh, Clayton Thomas, and S. Matthew Weinberg. Revisiting the Primitives of Transaction Fee Mechanism Design. In *EC*, page 703. ACM, 2024.
14. Lioba Heimbach and Roger Wattenhofer. Eliminating Sandwich Attacks with the Help of Game Theory. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022.
15. Mahimna Kelkar, Soubhik Deb, and Sreeram Kannan. Order-Fair Consensus in the Permissionless Setting. In *APKC '22: Proceedings of the 9th ACM on ASIA Public-Key Cryptography Workshop*, 2022.
16. Mahimna Kelkar, Soubhik Deb, Sishan Long, Ari Juels, and Sreeram Kannan. Themis: Fast, Strong Order-Fairness in Byzantine Consensus. 2021.
17. Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. Order-Fairness for Byzantine Consensus. In *CRYPTO*, page 451–480, 2020.
18. Kshitij Kulkarni, Theo Diamandis, and Tarun Chitra. Towards a Theory of Maximal Extractable Value I: Constant Function Market Makers. *CoRR*, abs/2207.11835, 2022.
19. Yuhao Li, Mengqian Zhang, Jichen Li, Elynn Chen, Xi Chen, and Xiaotie Deng. MEV Makes Everyone Happy under Greedy Sequencing Rule. In *Proceedings of the 2023 Workshop on Decentralized Finance and Security*, DeFi '23, page 9–15. Association for Computing Machinery, 2023.

20. Kaihua Qin, Liyi Zhou, and Arthur Gervais. Quantifying Blockchain Extractable Value: How dark is the forest? In *43rd IEEE Symposium on Security and Privacy, SP*, 2022.
21. Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais. Attacking the DeFi Ecosystem with Flash Loans for Fun and Profit. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part I*, 2021.
22. Geoffrey Ramseyer, Ashish Goel, and David Mazières. SPEEDEX: A Scalable, Parallelizable, and Economically Efficient Decentralized EXchange. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 849–875, 2023.
23. Geoffrey Ramseyer, Mohak Goyal, Ashish Goel, and David Mazières. Augmenting Batch Exchanges with Constant Function Market Makers. In *EC*, 2024.
24. Tim Roughgarden. Transaction Fee Mechanism Design. In *Proceedings of the 22nd ACM Conference on Economics and Computation (EC)*, pages 792–792, 2021.
25. Elaine Shi, Hao Chung, and Ke Wu. What Can Cryptography Do for Decentralized Mechanism Design? In *ITCS*, volume 251 of *LIPIcs*, pages 97:1–97:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.
26. Jiahua Xu, Krzysztof Paruch, Simon Cousaert, and Yebo Feng. SoK: Decentralized Exchanges (DEX) with Automated Market Maker (AMM) Protocols. 55(11), 2023.
27. Sen Yang, Kartik Nayak, and Fan Zhang. Decentralization of Ethereum's Builder Market. In *2025 IEEE Symposium on Security and Privacy (SP)*, page 1512–1530. IEEE, May 2025.
28. Mengqian Zhang, Yuhao Li, Xinyuan Sun, Elynn Chen, and Xi Chen. Maximal Extractable Value in Batch Auctions. In *Proceedings of the 26th ACM Conference on Economics and Computation (EC)*, page 510, 2025.
29. Patrick Zust. Analyzing and Preventing Sandwich Attacks in Ethereum. Bachelor's thesis.