Computation of Optimal MEV in Decentralized Exchanges

Mengqian Zhang[*]
Yale University
`mengqian.cs@gmail.com`

Yuhao Li
Columbia University
`yuhaoli@cs.columbia.edu`

Xinyuan Sun
Flashbots
`xinyuan@flashbots.net`

Elynn Chen
New York University
`elynn.chen@stern.nyu.edu`

Xi Chen
New York University
`xc13@stern.nyu.edu`

Abstract

In the ever-evolving blockchain ecosystem, decentralized exchanges (DEXs) have seen significant growth, which, however, also brought challenges of Maximal Extractable Value (MEV). Unlike centralized exchanges, DEXs offer a decentralized platform for cryptocurrency trading. Such trading mechanisms primarily include Constant Function Market Makers (CFMMs) and batch auctions. This paper delves into the optimal MEV strategies across these two DEX frameworks. As our first result, we show it is NP-hard to compute an optimal strategy in *any* CFMMs, when the fraction of swap fee is any constant larger than zero (*e.g.*, 0.3% being commonly used in Uniswap pools).

The second part of our paper examines DEXs utilizing batch auctions, which have gained a lot of attention for their appealing properties. By treating tokens and transactions as goods and traders within a pure exchange market, batch auctions can be formulated as a linear market, allowing exchange rates and trading outcomes in a batch to be derived from the prices and allocation in its market equilibrium. This design ensures fairness and efficiency from the perspective of economics. Despite the general belief that batch auctions are less vulnerable to MEV due to uniform pricing and order independence of transactions, we highlight that the block builder's current ability to rearrange the batch is certainly sufficient to extract MEV in batch auctions, where the strategic behavior is a novel *market equilibrium manipulation.* We further explore the computation and show that: When the transactions in a batch form a Fisher market, an optimal attack can be computed in polynomial time; When the transactions form a general Arrow-Debreu market, it is NP-hard to find such one.

Our results present a complete picture of the complexity of optimizing MEV in DEXs, contributing valuable insights to the ongoing discourse on DEX security and the solutions for addressing MEV therein.

Keywords: Decentralized Exchanges, Maximal Extractable Value, Batch Auctions, Market Equilibrium

[*]Majority of the work was done when the author was at New York University.

1 Introduction

Within the evolving landscape of blockchain technology, the decentralized finance (DeFi) space has witnessed unprecedented growth, with various financial services available on-chain. As the cornerstone of the DeFi movement, Decentralized Exchanges (DEXs) mark a significant shift in the paradigm of asset exchange within the blockchain ecosystem. Leveraging smart contracts on the blockchain, DEXs offer a peer-to-peer marketplace for users to trade cryptocurrencies (also known as *tokens*) without the intermediation of a central authority, which is aligned with the ethos of decentralization. Nowadays, the daily volume of DEXs has reached billions of US dollars [1].

Automated Market Makers (AMMs) are the most widely used type of DEXs. It replaces traditional order books with liquidity pools, each of which is composed of tokens of trading pairs and managed by an on-chain smart contract. In AMMs, users directly swap their tokens against the pool without waiting for a matching order. The exchange rate is algorithmically determined based on the reserves of tokens in the liquidity pool. The most commonly used are constant functions, such as the constant product formula in Uniswap [2], which requires the product of token reserves in the pool to remain constant before and after a trade. Thus, these AMMs are also known as constant function market makers (CFMMs). To avoid unexpected slippage, users are encouraged to set a slippage tolerance, which is equivalent to a limit/threshold of the exchange rate below which their transactions should fail. During a trade, a small fraction (say, 0.3%) of tokens is charged as swap fees to reward liquidity providers for their contribution to the liquidity.

Despite the huge advantages offered by DEXs, they also introduce novel challenges and vulnerabilities. A notable concern is Maximal Extractable Value (MEV) [3], a phenomenon that underscores the potential for block builders (*e.g.*, miners, validators) to exploit their position by transaction insertion, deletion, and reordering for financial gain. As of September 15th, 2022, prior to the Ethereum Merge, the total MEV extracted from the Ethereum network amounted to $675.62 million [4], a figure that has since escalated to $1.81 billion as of February, 2024 [5]. In the context of AMMs, the sensitivity to the processing order of transactions exposes them to various forms of MEV exploitation, notably through front-running [3], where transactions are strategically placed ahead of others for profit, and sandwich attacks [6], which exploit the price impact of a victim trade by placing orders on both sides. In addition to bad user experience, these unexpected MEV behaviors have also caused issues including network congestion, high gas prices, and the risk of blockchain re-organization and consensus instability [3, 7].

To address these vulnerabilities, the novel exploration of *batch auctions* presents a promising DEX design. Instead of processing transactions individually, DEXs utilizing batch auctions such as SPEEDEX [8] aggregate orders over a short period and execute them at a set of uniform prices. Here, "uniform" means that all successfully executed transactions in the same swapping direction, say $\mathcal{X} \to \mathcal{Y}$, are under the same exchange rate $p_{\mathcal{X}}/p_{\mathcal{Y}}$. Then, the design's key lies in setting prices for all involved tokens. This problem turns out to be closely related to the *market equilibrium*. By modeling tokens and transactions as goods and traders in a pure exchange market, respectively, a batch forms a linear market where each player's utility function is linear, and the unique price vector under market equilibrium is exactly what batch auctions need [8]. This feature of uniform exchange rates eliminates the execution sequence in batch auctions and makes such DEXs resistant to front-running, sandwich attacks, and internal arbitrage. Thus, it is a general belief that batch auctions are fundamentally less vulnerable to MEV. However, as we are going to show, it still leaves enough non-trivial space for a block builder to attack batch transactions and obtain MEV.

As a summary, in this paper, we propose:

With DEXs becoming ever more prevalent and impactful, it is imperative to finally settle the computational complexity of the MEV optimization problems therein.

1.1 Our Model and Contributions

This paper studies the computation of optimal MEV in the two different types of DEXs, *i.e.*, CFMMs and batch auctions. The analysis is under the same framework: both scenarios share the same format of transactions, the same strategy space, and the same utility function.

The Model. Specifically, we consider the exchange transactions among n tokens $\{\tau_1, \cdots, \tau_n\}$. In both scenarios, a transaction is trying to swap one type of token $\mathcal{X}$ for another $\mathcal{Y}$, represented in the format of $(\mathcal{X} \to \mathcal{Y}, \delta_{\mathcal{X}}, r)$ but denoted by Swap and Batch, respectively. In other words, each Swap or Batch transaction is composed of three components, *i.e.*, the exchange direction $\mathcal{X} \to \mathcal{Y}$, the number of tokens willing to sell $\delta_{\mathcal{X}}$, and the threshold for the exchange rate r which means the user should receive at least $\delta_{\mathcal{X}} \cdot r$ amount of token $\mathcal{Y}$ in return.

For a Swap transaction, tokens are exchanged by a certain CFMM, which supports the exchange between two involved tokens. Without loss of generality, we explore the optimal MEV among the Swap transactions between two tokens $\mathcal{X}, \mathcal{Y} \in \{\tau_1, \cdots, \tau_n\}$. Instead, in the batch auction scenario, all Batch transactions involving multiple tokens can be executed in the same batch.

Regarding the strategy space and utility function, we follow the consensus of the community that MEV refers to the *additional* value that can be extracted from block production by *including, excluding, and reordering* transactions in a block. From now on, we use attacker/mediator interchangeably to refer to the role (*e.g.*, block builders, miners) who can extract MEV. Given a set of m user transactions $\left\{\text{Swap}^i\right\}_{i\in[m]}$ or $\left\{\text{Batch}^i\right\}_{i\in[m]}$, the attacker is able to insert some transactions of the same type, select a subset of user transactions, and compute an order for these selected transactions, which together form an MEV strategy. For the batch auction scenario, the attacker can ignore the last reordering step because the order has no influence on the execution outcome of transactions.

Under a set of user transactions, once given an MEV strategy, the transactions' outcomes as well as the attacker's profits are determined. In this paper, the utility of a strategy is measured by the overall value of the attacker's final token holdings. Note that the attacker's profits are all from its own transactions. What utility functions of both scenarios (formally defined in Equation (1) and Equation (2)) do is to enumerate all attacker's newly added transactions, and for each transaction, its utility is the value of tokens finally received minus the value of tokens it brought. Here, the value of a token is measured by its exogenous price, which represents the attacker's self-belief – it may be the price in another DEX, another domain, or even the off-chain information (*e.g.*, the price of tokens in a centralized exchange like Coinbase). Throughout this paper, we assume that the exogenous prices remain the same during the attack, which is around 12 seconds in Ethereum.

Our Contributions. In literature, many excellent works studied the same or similar MEV issue in AMMs, with a focus on empirical approaches [9, 7, 6, 3]. One recent work [10] studies a special case with no swap fees, where they succeed in presenting a polynomial time algorithm. Yet, no polynomial time algorithm for the general setting (where the fraction of swap is a constant larger than 0, say 0.3%) is known. This is not a coincidence, as our first main theorem (Theorem 1) shows that computing a strategy that obtains optimal MEV is, in fact, NP-hard.

In addition, we initial the study of MEV on batch swaps. It's a widespread belief that batch auction is fundamentally less vulnerable to MEV since the outcome doesn't depend on the order of transactions. However, we first observe the ability to insert and delete transactions is already sufficient for the mediator to extract MEV (see Example 1 for a very concrete example)! By adopting the formulation of market equilibrium, such behavior is a *novel market equilibrium manipulation*, where a strategic player in the market has a very strong power – they can arbitrarily kick other participants out of the market and insert several fake identities. Although seemingly unreasonable, this is what could happen with block builders in the current blockchain system. This seeming unreasonableness exactly reflects the potential vulnerability of batch auctions, which, to the best of our knowledge, was not discussed before. One plausible reason is that while one *can* manipulate the batch auction in such a way, it is not clear *how* to manipulate since the outcome is not as easily predictive as that in AMMs. To this end, as our technical contributions, we discover many underlying combinatorial structures of the optimal attacks, the insights of which are going to be shown in the next subsection (Section 1.2).

As our second main contribution, we fully characterize the computational complexity of optimal MEV in batch auctions based on the structure of market: When the transactions in a batch form a Fisher market, an optimal attack can be computed in polynomial time (in fact, in almost linear-time!) (Theorem 2); When the transactions form a general Arrow-Debreu market, it is NP-hard to find such one (Theorem 3).

1.2 Overview of Insights in the Proofs

While optimizing MEV is computationally hard under both popular decentralized exchange scenarios (AMMs and Batch auctions), the routines towards these two theorems, in fact, provide many distinct insights, which we summarize next.

Let's first get intuition on why the mediator could get some MEV in CFMMs and then explain why it is hard to obtain optimal MEV. Imagine that at the latest state of a CFMM pool, the exchange rate of two tokens is exactly the same as the ratio of their exogenous prices. Then, the mediator could execute an arbitrary user transaction, after which the exchange rate in the CFMM must deviate. Thus, this leaves a space for the mediator to back-run and obtain some profits.

This is an ideal argument which, however, is not always true when there is a constant fraction of swap fees (in particular, $f = 0.3\%$ of tokens is charged in common Uniswap pools). Specifically, when the volume of a user's transaction is small, the profits obtained by back-running may not be able to beat the swap fees! Thus, we should adapt our intuition to try to back-run some large user transactions (*i.e.*, transactions that want to swap a large amount of tokens), and there is where another constraint comes in: the slippage tolerance of these transactions.

Suppose that there is a transaction $\textsc{Swap}$ that wants to swap a very large number of $\mathcal{Y}$ tokens for some $\mathcal{X}$ tokens, but with a non-trivial slippage tolerance requirement. The mediator would meet the following challenge to finish the best back-running: The mediator would like to find a set of users' $\mathcal{X} \to \mathcal{Y}$ transactions to reach a state at which the exchange rate of $\textsc{Swap}$ exceeds but is closest to its slippage tolerance. Thus, the NP-hard problem used in the reduction is the *Partition problem*, which exactly reflects the hardness of achieving the goal above. We will provide more intuition about this argument in Section 3.3 before the formal proof of Theorem 1.

Batch auctions have a more sophisticated structure, as they bring the concept of *equilibrium*. Thus, one small change of the batch (insert, delete, or modify a transaction) may result in dramatically different outcomes for every transaction, making it difficult to analyze compared to AMMs. Nevertheless, we observe many underlying combinatorial structures for optimal attacks and highlight the following two points, which we found insightful:

- In general, the mediator only needs to insert transactions with the same type as user transactions. Here, the "type" refers to which two tokens are involved in a transaction (Lemma 2). This implies there is always an optimal attack that preserves the market structure; in other words, the mediator is never necessary to complicate the market structure. In particular, when the mediator is attacking a set of user transactions that form a Fisher market, it suffices to consider attacks that remain a simple Fisher market, which is crucial to our efficient optimal algorithm for Fisher markets;

- In general, the directed graph of selected user transactions is acyclic (Lemma 3). The proof of this lemma follows a simple trick: If there was some cycle, then the mediator can locate one user transaction that can obtain profits, and replace it with their own transaction of exactly the same content. Despite being simple and intuitive, *this actually illustrates a kind of front-running in batch auctions*! Another consequence is that, under such kind of optimal attacks, there is essentially no "exchange" between two user transactions! The acyclic property also leads to identifying the hardness structure from the *Feedback Arc Set problem*, which we reduce to the MEV optimization problem for the general Arrow-Debreu market.

Finally, we briefly discuss the insights we can obtain from the two NP-hard problems we used. Even though NP-complete problems are all equivalent in terms of worst-case complexity, their corresponding optimization versions can behave dramatically differently in terms of *approximation*. In particular, the Partition problem has a fully polynomial-time approximation scheme (FPTAS), but the Feedback Arc Set problem only has constant approximation algorithm.

We conjecture that there is also a fully polynomial-time approximation scheme for the MEV optimization problem in AMMs, while it remains NP-hard to have even a good approximation algorithm for that in batch auctions. In fact, even for the instances in the reduction (that are constructed from instances of the Feedback Arc Set problem), we couldn't characterize the optimal MEV value, but only its very loose lower and upper bounds in terms of the solution of the Feedback Arc Set (see Lemma 6). Although by carefully chosen parameters, this is enough for us to finish the NP-hard reduction, we suspect the optimal MEV is generally produced by some non-linear programming with discrete decisions. If this conjecture is true, it also reflects the different levels of vulnerability of the two popular DEX systems.

1.3 Organization

The rest of the paper is organized as follows. In Section 2, we provide more background on the MEV, its concrete behaviors in AMMs, and the design of batch auctions. In Section 3, we provide the formulation of the MEV optimization problem in AMMs and show its NP-hardness (Theorem 1). In Section 4, we first provide a light preliminary of market equilibrium and formulate batch auctions in this framework. We then argue that the current strategy is already sufficient for the block builder to extract MEV, and formulate the optimal MEV problem. Then, we provide two general lemmas that reveal combinatorial structures in optimal attacks. Finally, we prove our computational results regarding the Fisher market and the general Arrow-Debreu market. We discuss some future directions in Section 5.

2 Background and Related Work

2.1 Maximal Extractable Value

Although DEXs allow users to directly interact with on-chain smart contracts through a trading transaction when they want to exchange cryptocurrencies, such a transaction only represents the individual's trading intent. The trading is truly executed when the transaction is included in a block on the canonical chain, which is managed by miners (in Proof-of-Work networks) or validators (in Proof-of-Stake networks). In the block-building process, they have the authority to decide which transactions are included in a block and in what order. It's found that block builders are able to extract additional value from block production in excess of the standard block rewards and gas fees by manipulating the block content. This additional value was initially referred to as "miner extractable value" and modified to be "maximal extractable value" since the transition from proof-of-work to proof-of-stake via The Merge. In practice, besides block builders, a large portion of MEV is extracted by independent network participants called "searchers." As the name suggests, they run complex algorithms to search profitable MEV opportunities and have bots to submit MEV-capture transactions automatically. The studies of MEV strategies naturally interest these roles, which are collectively referred to as the attacker or mediator in this paper.

2.2 MEV in AMMs

In the context of AMMs, the MEV phenomenon arises primarily due to the arbitrage opportunities caused by the price movement within a DEX or price discrepancy among DEXs. *DEX arbitrage* is the simplest and most well-known MEV opportunity. For a pair of tokens, if two DEXs are offering different exchange rates, someone can buy one token in the lower-priced DEX and sell it in the higher-priced DEX in a single transaction. Zhou *et al.* [7] translate the detection of DEXs arbitrage into a negative cycle detection problem, and Wang *et al.* [11] further analyze the profitability conditions and optimal trading strategies of cyclic arbitrages among multiple DEXs. After detecting such a profitable MEV opportunity, someone can submit its own transaction with the same arbitrage logic but a higher gas price to steal the profit by *front-running*. Torres *et al.* [12] perform a large-scale analysis of the real profits made by front-running attacks on Ethereum, providing evidence that front-running is highly lucrative. Qin *et al.* [9] provide a generalized transaction replay algorithm to clone and front-run a victim transaction without the need to understand the underlying transaction logic. *Sandwich attack* [6] is another common MEV behavior where a trader can "manually" create the arbitrage opportunity by exploiting a large order, the profit of which is quantified by Qin *et al.* [9]. Many other behaviors like back-run arbitrage [13] and cross-domain MEV [14] are also discussed in the literature.

The most related to us are the following papers. Bartoletti *et al.* [10] explore the same MEV optimization problem in AMMs but without swap fees, where the optimal attack is a multi-layer sandwich. [15] considers the practical scenario with swap fees and computes the optimal strategy to sandwich a single transaction. Our work fills this research gap by providing the computational hardness of attacking multiple transactions in AMMs with swap fees. Another related work is [16], which studies both with and without fees in a special AMM with greedy sequencing rule.

2.3 Batch Auctions

Batch auctions represent a novel evolution in DEX designs to address certain inefficiencies and MEV challenges inherent in AMMs. This paper mainly follows the mathematical model in SPEEDEX [8], which maps the computation of batch prices to a well-studied equilibrium computation problem of pure exchange markets (details are introduced later in Section 4.1). In SPEEDEX, the batch execution is triggered when the core SPEEDEX engine receives a new block, followed by an algorithm query to compute clearing valuations and thus the execution outcomes. Similarly, CowSwap [17], which uses mixed-integer programming to clear offers in batches, also contains two steps: a centralized entity known as the "driver" aggregates all user orders; these orders are then relayed to specialized centralized actors, termed "solvers", who bid to determine the settlement price (and over half [18] of non-stablecoin CowSwap orders are traded against private liquidity). As we will show later, the need to organize a batch through a mediator brings the risk of MEV. To the best our knowledge, we are the first to study MEV in batch auctions.

3 Optimal MEV in Constant Function Market Makers

In this section, we study the computation of optimal MEV in CFMMs. It is well known that the outcomes of transactions executed on CFMMs are sensitive to transaction-ordering attacks, which leaves space for a mediator (*e.g.*, miner, validator) to extract profits. We start by formalizing the CFMM and the execution of transactions.

3.1 CFMMs Formalization and Execution of Transactions

We consider a CFMM A for two types of tokens $\mathcal{X}, \mathcal{Y} \in \{\tau_1, \cdots, \tau_n\}$, where we use $s = (x, y)$ as a state of A that represents the current reserves of tokens $\mathcal{X}$ and $\mathcal{Y}$. The trading invariant can be modeled by a constant function on two variables $F(x, y)$. Throughout this paper, we will only consider the $\textsc{Swap}$-like transactions and there is no deposit or redemption of liquidity, so sometimes we equivalently use $s = (x, y) \in F$ to represent a reserving point on the curve $F(\cdot, \cdot)$. We assume two natural properties about these constant functions: (1) for any two points $(x, y), (x', y') \in F$, we have $x > x' \Leftrightarrow y < y'$, namely, when the reserve of $\mathcal{X}$ increases, the reserve of $\mathcal{Y}$ decreases and vice versa; (2) $F(x, y)$ is differentiable and the marginal exchange rate $\left|\frac{\partial F/\partial x}{\partial F/\partial y}\right|$ is decreasing with respect to x. We note that most CFMMs satisfy these two properties, including Uniswap v2 and v3. Finally, we denote the fraction of the swap fee in AMM A by $f \in [0, 1)$.

From the two properties above, we know that for any x, there is exactly one y such that $(x, y) \in F$ and vice versa. So for simplicity of notations, we use $F_y(x)$ to denote that y such that $(x, y) \in F$ and similarly define $F_x(y)$.

Suppose we are given a set of $\textsc{Swap}$ transactions $\{\textsc{Swap}^1, \cdots, \textsc{Swap}^m\}$ on AMM A, where each $\textsc{Swap}^i = (\delta_\mathcal{X}, r)$ or $\textsc{Swap}^i = (\delta_\mathcal{Y}, r)$. Without any ambiguity, we omit the swapping direction in this section and use the shortened notation $\textsc{Swap}^i(\delta_\mathcal{X}, r)$ to represent that the i-th transaction $\textsc{Swap}^i$ would like to sell $\delta_\mathcal{X}$ amount of token $\mathcal{X}$ to obtain at least $\delta_\mathcal{X} \cdot r$ amount of $\mathcal{Y}$, *i.e.*, with r as the lowest acceptable exchange rate. The meaning of $\textsc{Swap}^i(\delta_\mathcal{Y}, r)$ is analogous.

Pick an arbitrary permutation $\pi : [m] \to [m]$ as the execution order of all these m transactions and let $s_0 = (x_0, y_0)$ be the initial state of A before processing these transactions. The execution works as follows: Consider the i-th round and $\textsc{Swap}^{\pi(i)} = (\delta_\mathcal{X}, r)$. Let $\Delta \coloneqq F_y(x_{i-1}) - F_y(x_{i-1} + (1 - f)\delta_\mathcal{X})$. If the condition $\Delta \geq \delta_\mathcal{X} \cdot r$ holds, then this swap is successfully executed and $s_i = (x_{i-1} + (1 - f)\delta_\mathcal{X}, F_y(x_{i-1} + (1 - f)\delta_\mathcal{X}))$; otherwise, the transaction fails and $s_i = s_{i-1}$. The case where $\textsc{Swap}^{\pi(i)} = (\delta_\mathcal{Y}, r)$ can be defined similarly.

3.2 MEV Optimization Problem

In this subsection, we formalize the mediator's strategies and define its strategy space. Intuitively, a mediator can potentially delete some users' transactions, insert its own transactions, and pick an arbitrary order of execution of the selected transactions.

Definition 1 (Strategy Space)**.** *Given a set of users' transactions* $\left\{\textsc{Swap}^i\right\}_{i\in[m]}$ *and an initial state* $s_0 = (x_0, y_0)$*, a mediator could create* k *its own transactions* $\left\{\textsc{Swap}^i\right\}_{i\in[m+1:m+k]}$*, select a subset of users' transactions* $S \subseteq [m]$*, and pick an execution order (a permutation) over all these transactions* $\pi : [|S \cup [m+1 : m+k]|] \to S \cup [m+1 : m+k]$*.*

Definition 2 (Utility Function)**.** *Mediator's profit* $U(\{\textsc{Swap}^i\}_{i\in[m+1:m+k]}, S, \pi)$ *is defined as*

$$\sum_{i\in[|S|+k],\pi(i)\in[m+1:m+k]} \frac{x_{i-1} - x_i}{1 - f \cdot \mathbb{1}_{\{x_i > x_{i-1}\}}} \cdot p_x^* + \frac{y_{i-1} - y_i}{1 - f \cdot \mathbb{1}_{\{y_i > y_{i-1}\}}} \cdot p_y^*, \tag{1}$$

where p_x^* *and* p_y^* *are exogenous (or say, off-chain) prices of* $\mathcal{X}$ *and* $\mathcal{Y}$ *respectively.*

This definition generalizes the idea of sandwich attack and can capture a wide range of order manipulation attacks. The goal of the mediator is to find a strategy $(\{\textsc{Swap}^i\}_{i\in[m+1:m+k]}, S, \pi)$ that maximizes its MEV (profits). Sometimes we also refer to such a strategy as an *optimal strategy.*

The following observation can help us simplify the notations when we study Swap transactions. Note that the lemma doesn't hold for Batch transactions, which we are going to study in the next section.

Observation 1 (No-Deleting in AMMs)**.** *Given any set of users' transactions* $\left\{\textsc{Swap}^i\right\}_{i\in[m]}$ *and an initial state* $s_0 = (x_0, y_0)$*, it is without loss of generality to assume that the optimal strategy satisfies* $S = [m]$*, i.e., the mediator will always select the full set of transactions.*

Proof. The proof is easy. For mediator's transactions, it can always create the transactions that it needs. For the users' transactions, if the mediator didn't select some users' transactions in an optimal strategy, it can also equivalently put them at the very end of the sequence, and it will not affect the mediator's profits. □

Based on the observation above, we may omit the parameter S below for simplicity of notations.

3.3 Computational Hardness

In this section, we systematically study the optimal MEV on Swap transactions. Recall that in literature many excellent works studied the same or similar attack, with specific focus on empirical approaches [9, 7, 6, 3], or special cases (*e.g.*, no swap fee [10] or attacking one user's transaction [15]), but no polynomial time algorithm for the general setting is known. This is not a coincidence, as what we are going to show in this section.

In particular, as our main theorem in this section, we show that computing an optimal strategy for MEV is NP-hard when swap fee $f > 0$ is any constant (say $f = 0.3\%$), which is what happens in the real blockchain world. This indicates that if one wants to attract optimal MEV, it is necessary to design efficient heuristic or approximate algorithms, but not hope for a theoretically efficient optimal algorithm.

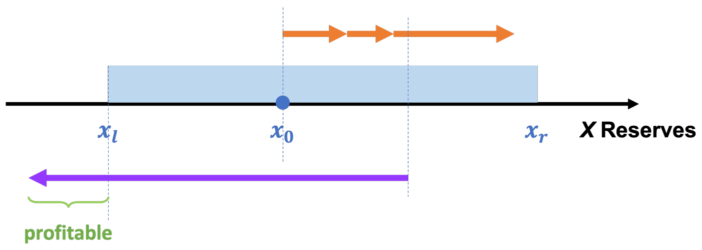

Figure 1: An illustration of the reduction in the proof of Theorem 1. Dark blue line means x-axis which represents the reserved amount of $\mathcal{X}$ tokens. Light blue shadow means the "arbitrage-free" interval $[x_\ell, x_r]$, where any mediator's transaction cannot obtain profits if the execution state is in this interval due to the trade fees. The orange blocks represents the set of $\mathcal{X} \to \mathcal{Y}$ users' transactions; they satisfy that starting from the initial state, after executing all these transactions, the state remains in the arbitrage-free interval (so that the mediator can never use them to obtain profits). The purple block represents a large enough user's $\mathcal{Y} \to \mathcal{X}$ transaction (which is the source where the mediator could obtain profits by back-running), and the purple dotted line represents the state that exactly satisfies its slippage tolerance. The left green part is where the mediator can back-run and obtain profits. The goal of the mediator is to find a subset of users' $\mathcal{X} \to \mathcal{Y}$ transactions such that the sate exceeds and is closest to the dotted purple line, so the green part is as large as possible. This construction can encode any instance of the Partition problem, formally shown in the proof of Theorem 1.

Theorem 1. *Let $f \in (0,1)$ be any universal constant. It is NP-hard to solve the problem of finding an optimal strategy on* SWAP *transactions, even with the constant function $F(x,y) = xy$ (i.e., Uniswap v2).*

Before getting into details of the proof, we note that an intuitive illustration of the reduction is provided in Figure 1.

Proof. We reduce the NP-hard Partition problem to our problem. Recall that an instance of the Partition problem contains m positive integers $\{c_1, \cdots, c_m\}$ and asks if it can be partitioned into two subsets $S_1, S_2 \subseteq [m]$ with $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = [m]$ such that the sum of numbers in S_1 equals that in S_2, *i.e.*, $\sum_{i \in S_1} c_i = \sum_{i \in S_2} c_i = \frac{1}{2} \sum_{i \in [m]} c_i$.

Suppose that we are given an arbitrary set of m positive integers $\{c_1, \cdots, c_m\}$ such that the sum of all c_i's is an even number (otherwise the answer to the Partition problem is obviously "no").

To start the reduction, we will construct a CFMM, a set of uses' transactions $\{\text{SWAP}^i\}$, and an initial state. Concretely, it suffices for us to use the product function $F(x,y) = xy$ (other CFMMs that satisfy our two properties defined in Section 2 will also work). Let the initial state $s_0 = (x_0, y_0)$ be such that $\frac{\partial F/\partial x}{\partial F/\partial y} = p_x^*/p_y^*$. We will have $m+1$ users' transactions; m of them will be $\text{SWAP}(\mathcal{X} \to \mathcal{Y})$, namely, they are trying to sell $\mathcal{X}$ to obtain $\mathcal{Y}$, and the other will be $\text{SWAP}(\mathcal{Y} \to \mathcal{X})$.

Let x_r be such that $\frac{\partial F_y(x_r)}{\partial x} = \frac{(1-f)p_x^*}{p_y^*}$ and x_ℓ be such that $\frac{\partial F_y(x_\ell)}{\partial x} = \frac{p_x^*}{(1-f)p_y^*}$. Let K be the smallest positive integer such that $t/K \le x_r - x_0$. Then we let $\{d_1, \cdots, d_m\} = \{c_1/K, \cdots, c_m/K\}$ and $t = \frac{1}{2}\sum_{i \in [m]} d_i$. Our problem is still to decide whether there exists $S_1 \subseteq [m]$ such that $\sum_{i \in S_1} d_i = t$. Obviously this doesn't change the original problem, but makes it easier for us to finish the reduction.

Let $x^* = x_0 + t$. We first construct a huge user's transaction $\text{SWAP}^{m+1} = (\mathcal{Y} \to \mathcal{X}, \delta_y)$, where

$\delta_y = (y_0 - F_y(x^*) + L)/(1-f)$ with a sufficiently large number L. The exchange ratio threshold here is crucial: it is defined as

$$r = \frac{x^* - F_x((1-f)\delta_y + F_y(x^*))}{\delta_y}.$$

Intuitively, this means the transaction $\textsc{Swap}^{m+1}$ can be successfully executed at some state $s = (x, y)$ if and only if $x \geq x^*$. $(\diamond)$

We then construct m users' transactions $\{\textsc{Swap}^i\}_{i\in[m]}$, where each $\textsc{Swap}^i = (\mathcal{X} \to \mathcal{Y}, \delta_{\mathcal{X}} = d_i/(1-f))$ (we'll specify the thresholds of exchange ratios of these transactions later; they will also be crucial). Intuitively, this construction means if we only execute users' transactions, then we can reach the state $s^* = (x^*, F_y(x^*))$ *if and only if* there exists $S_1 \subseteq [m]$ such that $\sum_{i\in S_1} d_i = t$, which corresponds to the answer to the given instance of the Partition problem.

Ideally, we would like the "if and only if" above also extents to the general case where we allow the mediator to insert some of its own transactions *for the purpose of maximizing its MEV*. This is where we need to be careful to construct the thresholds of exchange ratios of the users' transactions.

For each transaction $\textsc{Swap}^i$ for $i \in [m]$, we set the threshold of exchange ratio to be

$$\frac{F_y(x^* - d_i/(1-f)) - F_y(x^*)}{d_i/(1-f)}.$$

Intuitively, this means if a user's transaction $\textsc{Swap}^i$ is executed at state $s = (x, y)$, then the $\textsc{Swap}(\delta_{\mathcal{X}})$ is successfully executed if and only if $x \leq x^* - d_i/(1-f)$. Equivalently, after successfully executing any user's transaction, the state $s = (x, y)$ satisfies $x \leq x^*$. $(\star)$

This finishes the construction of an instance of the optimal MEV problem. Next, we show that if a mediator can find an optimal strategy, then it can solve the given Partition instance.

To this end, we define the mediator's maximal MEV as follows (recall that x_ℓ is such that $\frac{\partial F_y(x_\ell)}{\partial x} = \frac{p_y^*}{(1-f)p_x^*}$):

$$\texttt{MMM} = (y_0 + L - F_y(x_\ell)) \cdot p_y - \frac{x_\ell - F_x(y_0 + L)}{1-f} \cdot p_x.$$

The correctness of our reduction follows from the next lemma.

Lemma 1. *There exists a strategy such that $U(\{\textsc{Swap}^i\}_{i\in[m+2:m+1+k]}, \pi) \geq \texttt{MMM}$ if and only if there exists S_1 such that $\sum_{i\in S_1} d_i = t$.*

The reason that this lemma implies the correctness of the reduction is simply that if the mediator can find an optimal strategy, then it can easily compare the profit with MMM and solve the Partition problem. Due to the space limit, we postpone the proof into appendix. □

4 Optimal MEV in Batch Auctions

Different from AMMs discussed above, batch auction is a trading mechanism that computes a uniform exchanging price and executes all transactions simultaneously. So the prices and outcome of the execution do not depend on the order of the transactions. Because of this, batch auctions are believed to be fundamentally less vulnerable to MEV compared to order-dependent mechanisms.

However, even though batch swaps are robust against front-running and sandwich attacks discussed above, we observe new strategy space for a mediator to obtain MEV by manipulating the block content. In the rest of this section, we first introduce the concept of market equilibrium for efficiency of the batch price, which was also introduced in the design of SPEEDEX as the mathematical foundation [8]. Then we *initial and formalize* the study of MEV therein.

4.1 Market Equilibrium Formulation of Batch Auctions

Pure Exchange Market. A pure exchange market consists of n divisible goods, denoted by $\{\tau_1, \cdots, \tau_n\}$ and m traders, denoted by $\{T_1, \cdots, T_m\}$. Every trader T_i initially owns some endowment $\mathbf{w}_i \in \mathbb{R}^n_{\geq 0}$, where w_{ij} represents the amount of τ_j that T_i owns. Every trader i has its own utility function $\mathcal{U}_i : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$, where $\mathcal{U}_i(\mathbf{x}_i)$ means T_i's utility if she gets a bundle of goods with amount x_{ij} for τ_j. In this paper, we focus on linear market, meaning that the utility function $\mathcal{U}_i = \sum_j u_{ij} x_{ij}$, where $u_{ij} \geq 0$ is the utility (or say preference) of trader i for a unit amount of goods τ_j. In economics, it has long been understood that prices are determined by the interplay of supply and demand where under a *competitive price*, supply precisely meets demand. Such an idea was captured by *market equilibrium.*

Definition 3 (Market Equilibrium)**.** *A price vector* $\mathbf{p}$ *along with an allocation* $\mathbf{x}$ *is a market equilibrium if the following conditions meet:*

- *Market Clearance:* $\sum_{i\in[m]} x_{ij} = \sum_{i\in[m]} w_{ij}$ *for all* $j \in [n]$*;*
- *Budget Constraint:* $\sum_{j\in[n]} x_{ij} p_j = \sum_{j\in[n]} w_{ij} p_j$ *for all* $i \in [m]$*; and*
- *Individual Optimality:* $\mathbf{x}_i \in \texttt{OPT}_i(\mathbf{p})$*, where* $\texttt{OPT}_i := \arg\max \mathcal{U}_i(\mathbf{x}_i)$ *among all* $\mathbf{x}_i$ *that satisfies the budget constrain* $\sum_{j\in[n]} x_{ij} p_j \leq \sum_{j\in[n]} w_{ij} p_j$.

The existence of market equilibrium (sometimes called general equilibrium) under mild conditions was proved by Arrow and Debreu [19] and independently by McKenzie, which is regarded by many as the crown jewel of Mathematical Economics. Our world is much easier to navigate: We will show that the batch prices can be computed by a linear market equilibrium.

Consider a set of Batch transactions $\{\text{Batch}^1, \cdots, \text{Batch}^m\}$ among tokens $\{\tau_1, \cdots, \tau_n\}$, where each $\text{Batch}^i = (\mathcal{X}^i \to \mathcal{Y}^i, \delta_{\mathcal{X}^i}, r_i)$ would like to swap $\delta_{\mathcal{X}^i}$ amount of token $\mathcal{X}^i$ for some token $\mathcal{Y}^i$ and the exchange rate between $\mathcal{Y}^i$ and $\mathcal{X}^i$ should be at least r_i (*i.e.*, the amount of received $\mathcal{Y}^i$ should be no less than $r_i \delta_{\mathcal{X}^i}$). Here, tokens naturally correspond to the goods in the market. Every transaction Batch^i can be viewed as a trader, where its endowment is $w_{i,j} = \delta_{\mathcal{X}^i}$ for $\tau_j = \mathcal{X}^i$ and $w_{i,j} = 0$ otherwise. Its utility function is defined as $\mathcal{U}_i(\mathbf{x}_i) = r_i \cdot x_{ij} + x_{ik}$ for $\tau_j = \mathcal{X}^i$ and $\tau_k = \mathcal{Y}^i$. Given any set of Batch transactions, we refer to the corresponding market as an Arrow-Debreu market if there is no more specified structure (this is mainly for comparison of the Fisher market that we will define below).

The market constructed above is a linear market in which every participant's utility function is linear (in fact, it is even more succinct – only two goods have non-zero coefficients). The linear market enjoys many nice properties: First, previous work [20] implies that our batch auction structure always has a *unique* competitive price vector[1], so that we do not face the equilibrium selection problem. In the distributed system where the Batch transactions are eventually processed by an arbitrary node, it releases us from the concern that certain nodes may strategically decide

[1] The uniqueness is in terms of scaling, which means scaling the equilibrium price vector by a constant is also an equilibrium. It does not matter because what is really needed is the ratio of prices, which remains the same no matter how it is scaled.

the exchange rates. Second, it is polynomial-time computable, while for some other classic markets like Leontief or CES (stands for constant elasticity of substitution) functions, the computation of market equilibrium is computationally hard (PPAD-complete [21, 22]). Third, it is generally well understood from the works of past decades by researchers from economics, computer science, and operation research.

Proposition 1 ([8])**.** *By modeling the tokens $\{\tau_j\}_{j\in[n]}$ and transactions $\{\text{Batch}^i\}_{i\in[m]}$ as goods and traders, under the (unique) competitive price vector $\mathbf{p}$ and arbitrary equilibrium allocation $\mathbf{x}$, it satisfies*

- *Internal arbitrage-free: The prices are internal arbitrage-free (simply because we have a uniform price vector);*
- *Soundness: Any exchange (allocation) follows the price vector $\mathbf{p}$ and meets its threshold requirement for the exchange ratio;*
- *Completeness: For any Batch^i with its threshold r_i, $\tau_j = \mathcal{X}^i$ and $\tau_k = \mathcal{Y}^i$, if $p_j/p_k > r_i$, then Batch^i sells all its $\mathcal{X}^i$ and gets $\delta_{\mathcal{X}^i} \cdot p_j/p_k$ amount of $\mathcal{Y}^i$.*

4.2 MEV Optimization Problem

The feature of uniform price in batch auctions makes it resistant to several DEX-related MEV behaviors like sandwich attacks and internal arbitrage. As a result, MEV seems impossible in batch auctions. We first note that this is not the case – we observe new strategic behavior, which is high-levelly in line with block content manipulation, but different from well-known MEV strategies like front-running or sandwich attack. The point is the execution result of a Batch transaction depends on which other transactions are included in the batch. Therefore, a strategic mediator can affect the outcome of exchanges by manipulating the batch contents (inserting and deleting transactions). The mediator still respects the market equilibrium outcomes, but that of the *manipulated block content*, during which it may gain profits. We give an example to provide more intuition.

Example 1 (MEV in Batch Auctions)**.** *Consider the scenario with three tokens $\{A, B, C\}$ and three user transactions $\{\text{Batch}^i\}_{i\in[1,3]}$ where $\text{Batch}^1 = (A \to B, 2, 0.5)$, $\text{Batch}^2 = (B \to C, 1, 4)$, and $\text{Batch}^3 = (C \to A, 4, 0.5)$. Assume the exogenous prices of tokens are all one, namely, $p^*_A = p^*_B = p^*_C = 1$. Given this batch of transactions, an honest mediator makes them exchange with each other (i.e., three users receive $1B, 4C, 2A$, respectively) under the price equilibrium $p_A = 0.5, p_B = 1, p_C = 0.25$. Nevertheless, a strategic mediator can extract some additional value by re-organizing the batch. One method is to exclude Batch^2 and insert two attacking transactions $\{\text{Batch}^4 = (B \to A, 1, 2),\ \text{Batch}^5 = (A \to C, 2, 2)\}$. By executing them in the batch $\{\text{Batch}^i\}_{i\in\{1,3,4,5\}}$, the attacker receives $2A$ and $4C$ at the cost of $1B$ and $2A$, obtaining a net benefit of 3 (recall that their exogenous prices are all 1). Another way is to directly replace the Batch^2 with the same attacking transaction $\text{Batch}^{2'} = (B \to C, 1, 4)$. In this way, the attacker can also get a profit of 3.*

Next, we give formal definitions of an attacker's strategy space and utility function.

Definition 4 (Strategy Space)**.** *Given a batch of user transactions $\{\text{Batch}^i\}_{i\in[m]}$, a mediator could select a subset of all these transactions $S \subseteq [m]$, create k its own Batch transactions $\{\text{Batch}^i\}_{i\in[m+1:m+k]}$, and execute them in a same batch.*

Without loss of generality, we assume that all mediator's newly inserted transactions will be successfully executed; otherwise, removing failed ones from the set $\{\text{BATCH}^i\}_{i\in[m+1:m+k]}$ makes no difference in results.

Definition 5 (Utility Function)**.** *Mediator's profit $U(S,\{\text{BATCH}^i\}_{i\in[m+1:m+k]})$ is defined as*

$$\sum_{i\in[m+1:m+k]} -w_{ij}\cdot p_j^* + x_{i\ell}\cdot p_\ell^*, \tag{2}$$

where $\text{BATCH}^i = (\tau_j \to \tau_\ell, \delta_{\tau_j} = w_{ij}, r_i)$, $\mathbf{x}$ is an allocation under equilibrium, and vector $\mathbf{p}^*$ represents the exogenous prices of tokens.

This section asks the following research question: Given a batch of user transactions, what's the optimal strategy for a mediator to maximize its utility? Based on the structure of the given user transactions, we discuss this problem for the Fisher Market (Section 4.4) and the general Arrow-Debreu market (Section 4.5), respectively.

4.3 Combinatorial Structures in Optimal Attacks

In this section, we present two general lemmas that provide more intuition about how to tackle the MEV Optimization Problem in Batch auctions, and are very useful to simplify the analysis of optimal attacks. Before that, from a technical perspective, we recall the definition of the economy graph of a market, which was defined by Maxfield [23]. In our context, we will use the terminology of transactions, and define both the directed and undirected versions of economy graphs for our purpose.

Definition 6 (Economy Graph)**.** *Given a set of transactions $\{\text{BATCH}^i\}_{i\in[m]}$, we define a directed graph as follows. Each vertex corresponds to a token τ_i. For two tokens τ_i and τ_j, we add a directed edge from τ_i to τ_j if there is a transaction BATCH^k that swaps τ_i for τ_j. We call this (directed) graph G the economy graph of $\{\text{BATCH}^i\}_{i\in[m]}$.*

The undirected economy graph H is defined as the same graph as the directed economy graph G but all edges are changed to undirected.

Intuitively, if there is an undirected edge between two tokens in H, this means there is a user who is interested in these two tokens (would like to swap one token for the other).

Now we are ready to introduce the lemmas.

At a high level, the first lemma works on the side of inserting attacking transactions, stating that for an optimal attack, it is *never* necessary to make the undirected economy graph more complicated: it is sufficient to only insert transactions along edges of the undirected economy graph of initial users' transactions $\{\text{BATCH}^i\}_{i\in[m]}$. The second lemma concerns the side of selecting users' transactions, showing that for an optimal attack, it suffices to select users' transactions such that the directed economy graph of them is acyclic. These two lemmas bring some insights for the mediator to search for an optimal attack and are crucial for both our efficient algorithm design and NP-hard analysis in subsequent sections.

Lemma 2. *Let $H=(V,E)$ be the undirected economy graph of users' transactions $\{\text{BATCH}^i\}_{i\in[m]}$. There is an optimal attack $(S,\{\text{BATCH}^i\}_{i\in[m+1:m+k]})$ such that the undirected economy graph $H'=(V,E')$ of $\{\text{BATCH}^i\}_{i\in[m+1:m+k]}$ satisfies $E'\subseteq E$.*

Proof. First, we assume that the original undirected economy graph $H = (V, E)$ is connected, otherwise the mediator can attack each connected component separately.

Given any general strategy $(S, \{\textsc{Batch}^i\}_{i\in[m+1:m+k]})$, we will construct a new strategy such that (1) the undirected economy graph of the attacking transactions in the new strategy is a subgraph of H; and at the same time (2) the mediator's profit is the same as using $(S, \{\textsc{Batch}^i\}_{i\in[m+1:m+k]})$. By doing so, we can conclude this lemma. In particular, the selected users' transactions are the same, so we focus on the attacking transactions next.

Consider any transaction $\textsc{Batch}^i = (\tau_j \to \tau_{j'}, w_{ij}, r_i)$ for some $i \in [m+1 : m+k]$. Suppose that $\textsc{Batch}^i$ is successfully executed and $(j, j') \notin E$. Note that if $\textsc{Batch}^i$ is not successfully executed, we can simply remove it from the strategy and everything will remain unchanged.

We will replace the transaction $\textsc{Batch}^i$ with a set of new transactions such that the edges that correspond to new transactions are a subset of E, and the mediator's profit remains unchanged.

Since H is connected, there is at least one simple path from j to j'. Denote such path by $v_0, \cdots, v_t$, where $v_0 = j$ and $v_t = j'$. Let $\mathbf{p}$ be the equilibrium price vector of $(S, \{\textsc{Batch}^i\}_{i\in[m+1:m+k]})$. We construct t new transactions based on these information as follows. For every $\ell \in [t]$, we let the new transaction be $(\tau_{v_{\ell-1}} \to \tau_{v_\ell}, w_{ij} \cdot \frac{p_{v_{\ell-1}}}{p_0}, \frac{p_{v_\ell}}{p_{v_{\ell-1}}})$. Note that by construction all these new transactions are with respect to E, i.e., the edges that correspond to these newly constructed transactions are subset of E.

Now we show that the mediator's profit remains unchanged. Note that mediator brings $w_{ij} \cdot \frac{p_{v_{\ell-1}}}{p_0}$ more $\tau_{v_{\ell-1}}$ tokens. But it is easy to verify that under new market equilibrium, each transaction $(\tau_{v_{\ell-1}} \to \tau_{v_\ell})$ will be successfully executed, and obtain $w_{ij} \cdot \frac{p_{v_\ell}}{p_0}$ many taken τ_ℓ. In total, the payoff of middle transactions will be canceled and the mediator will pay w_{ij} many token τ_j and get $w_{ij} \cdot \frac{p_{v_t}}{p_0} = w_{ij} \cdot \frac{p_{j'}}{p_j}$ many token τ_j, which is exactly the same as before.

This finishes the proof. □

Lemma 3. *There is an optimal attack $(S, \{\textsc{Batch}^i\}_{i\in[m+1:m+k]})$ such that the economy graph of selected users' transactions $\{\textsc{Batch}^i\}_{i\in S}$ is acyclic.*

Proof. Without loss of generality, assume that all transactions in this batch, namely, all transactions $\{\textsc{Batch}^i\}_{i\in S} \cup \{\textsc{Batch}^i\}_{i\in[m+1:m+k]}$ are successfully executed. Otherwise the mediator can remove all transactions that are not executed and the profit will be the same.

Suppose there is a directed cycle in the economy graph of selected users' transactions. Without loss of generality, assume the cycle is $(\textsc{Batch}^1, \textsc{Batch}^2, \cdots, \textsc{Batch}^t)$ with a vertex sequence $(\tau_1, \tau_2, \cdots, \tau_t, \tau_1)$, where $t \in [2, m]$. Then, under equilibrium, there are only two cases.

- Case 1: the exchange rates of all transactions in the cycle are exactly the same as the ratio of corresponding tokens' exogenous prices. In this case, replacing one of the user's transactions with an attacking transaction of the same content has no impact on the equilibrium nor the mediator's profit (as the profit of this replacement transaction is 0).

- Case 2: the exchange rates of transactions in the cycle are not all the same as the ratio of corresponding tokens' exogenous prices. Note that $\prod_{j=1}^{t-1} \frac{p_{\tau_j}}{p_{\tau_{j+1}}} \cdot \frac{p_{\tau_t}}{p_{\tau_1}} = 1 = \prod_{j=1}^{t-1} \frac{p^*_{\tau_j}}{p^*_{\tau_{j+1}}} \cdot \frac{p^*_{\tau_t}}{p^*_{\tau_1}}$, where $\frac{p_{\tau_j}}{p_{\tau_{j+1}}}$ is the equilibrium exchange rate of the transaction swapping τ_j for τ_{j+1} and $\frac{p^*_{\tau_j}}{p^*_{\tau_{j+1}}}$ is the exogenous rate of these two tokens. In this case, at least one transaction's equilibrium exchange rate is better than the exogenous one, say $\textsc{Batch}^j$ with the direction $\tau_j \to \tau_{j+1}$.

In other words, $p_j/p_{j+1} > p^*_j/p^*_{j+1}$. Then, replacing *this user's transaction* by an attacking transaction with the same content (namely, remove this user's transaction from the batch and insert a mediator's transaction with the same content) does not influence the equilibrium but increases the mediator's utility.

In this way, for any attacking strategy with a directed cycle in the economy graph of selected user transactions, there is another strategy with no such cycle, such that the attacker receives no less profit than the previous one.

This finishes the proof. □

4.4 Optimal MEV under Fisher Market

Our main result in this section is an efficient (in fact, almost linear-time) algorithm for the mediator to compute an optimal strategy for MEV under *Fisher market* structure, where trades occur between τ_1 and τ_j for all $j \in [2:n]$, with no trades taking place between τ_j and $\tau_{j'}$ for any $j, j' \in [2:n]$. An interpretation of this model is to view τ_1 as USDC and every user's transaction is trying to trade between other cryptocurrencies with USDC.

We first give two lemmas to prepare for our optimal MEV algorithm.

The first lemma says that the mediator can independently attack the user transactions that occur between τ_1 and τ_j for all $j \in [2:n]$.

Lemma 4. *For every $j \in [2:n]$, let $S_{1\leftrightarrow j} \subseteq [m]$ be the indices of all users' transactions that trade between τ_1 and τ_j, and $\mathtt{ATT}_j$ be an optimal strategy to attack $\left\{\textsc{Batch}^i\right\}_{i\in S_{1\leftrightarrow j}}$. Then, $\cup_{j\in[2:n]}\mathtt{ATT}_j$ is an optimal attack to $\left\{\textsc{Batch}^i\right\}_{i\in[m]}$, where a union of two strategy is defined as the union of selected users' transactions and the union of inserted attacking transactions respectively.*

Proof. Let $SS_{1\to j}$ denote the set of successfully executed transactions swapping τ_1 for τ_j in the final batch, while $SS_{j\to 1}$ denote the set of successfully executed transactions that bring token τ_j and get τ_1. Recall that under a market equilibrium, the market clearance condition holds. It implies that for any token τ_j where $j \in [2:n]$, we have

$$\sum_{\textsc{Batch}^i\in SS_{1\to j}} x_{ij} = \sum_{\textsc{Batch}^i\in SS_{j\to 1}} w_{ij}. \tag{3}$$

Let p_1 and p_j be the equilibrium prices of τ_1 and τ_j, respectively. The requirement that everyone spends their entire profit under the equilibrium implies that

$$\sum_{\textsc{Batch}^i\in SS_{1\to j}} x_{ij}p_j = \sum_{\textsc{Batch}^i\in SS_{1\to j}} w_{i1}p_1, \qquad \sum_{\textsc{Batch}^i\in SS_{j\to 1}} x_{i1}p_1 = \sum_{\textsc{Batch}^i\in SS_{j\to 1}} w_{ij}p_j. \tag{4}$$

Combining Equation (3) and Equation (4), we have

$$\sum_{\textsc{Batch}^i\in SS_{1\to j}} w_{i1} = \sum_{\textsc{Batch}^i\in SS_{j\to 1}} x_{i1}. \tag{5}$$

It means that under equilibrium, the consumed token τ_1 in the direction $\tau_j \to \tau_1$ are all from the transactions in the opposite direction; and vice versa for token τ_j according to Equation (3). In other words, each pair of tokens (τ_1, τ_j) in the Fisher market can be viewed as a sub-market where transactions between them are self-sufficient under equilibrium. Thus, the mediator is able to independently attack the user transactions in each sub-market, namely, to consider selecting which user transactions and inserting which attacking transactions. □

Now we are able to focus on each individual pair of tokens (τ_1, τ_j) for every $j \in [2:n]$. The next lemma further simplifies the strategy space for optimal attacks therein.

Lemma 5. *For each pair (τ_1, τ_j) where $j \in [2:n]$, there is an optimal attack which inserts transactions in at most one direction, specifically. Furthermore, all transactions in this direction are from the mediator.*

Proof. Without loss of generality, suppose the mediator inserts a transaction in each direction between τ_1 and τ_j where $j \in [2:n]$, denoted by $\textsc{Batch}^{m+1} = (\tau_1 \to \tau_j, \delta_{\tau_1} = w_{m+1,1}, r_{m+1} = p_1/p_j)$ and $\textsc{Batch}^{m+2} = (\tau_j \to \tau_1, \delta_{\tau_j} = w_{m+2,j}, r_{m+2} = p_j/p_1)$. If $w_{m+1,1} \cdot p_1/p_j = w_{m+2,j}$ where p_1 and p_j are the tokens' equilibrium prices, $\textsc{Batch}^{m+1}$ and $\textsc{Batch}^{m+2}$ supply each other, bringing a utility of 0. Then, removing both transactions has no impact on the mediator's utility. Otherwise, let

$$\textsc{Batch}^{m+3} = \begin{cases} (\tau_1 \to \tau_j, \delta_{\tau_1} = w_{m+1,1} - w_{m+2,j} \cdot p_j/p_1, r_{m+3} = p_1/p_j), & w_{m+1,1} \cdot p_1/p_j > w_{m+2,j}; \\ (\tau_j \to \tau_1, \delta_{\tau_j} = w_{m+2,j} - w_{m+1,1} \cdot p_1/p_j, r_{m+3} = p_j/p_1), & w_{m+1,1} \cdot p_1/p_j < w_{m+2,j}. \end{cases} \tag{6}$$

Following the observation that $\textsc{Batch}^{m+1}$ and $\textsc{Batch}^{m+2}$ will be partially exchanged with each other, replacing them with $\textsc{Batch}^{m+3}$ has no impact on the batch execution and mediator's utility. As a result, only one direction has the attack transaction. In fact, if $1 \to j$ is the direction, then the prices must satisfy $p_1/p_j \geq p_1^*/p_j^*$, and if $j \to 1$ is the direction, then the prices must satisfy $p_1/p_j \leq p_1^*/p_j^*$. Otherwise removing this attack transaction (and all user transactions between τ_1 and τ_j) from the batch increases the mediator's utility.

Then we show that all transaction in this direction are from the mediator. Under equilibrium, if there are user transactions successfully executed in a this direction, replacing them with attacking transactions of the same content is still an equilibrium but brings more profit for the mediator.

This concludes the proof. □

Now we are ready to describe our main algorithm and state our main theorem.

Theorem 2. *Given a set of users' transactions $\{\textsc{Batch}^i\}_{i\in[m]}$ such that they form a Fisher market (i.e., trades occur exclusively between τ_1 and τ_j for all $j \in [2:n]$, with no trades taking place between τ_k and τ_j for any $k, j \in [2:n]$), Algorithm 1 finds a strategy that can obtain optimal MEV in time $\tilde{O}(m)$, where the notation $\tilde{O}(\cdot)$ hides polylogrithemic factors.*

Proof. Algorithm 1 takes a batch of user transactions that forms a Fisher market and the exogenous prices as input and outputs a strategy to make the mediator obtain the optimal profits. It independently processes each pair (τ_1, τ_j) for all $j \in [2:n]$. For each pair, it decides which user transactions to select and how to insert its own attacking transaction (including the direction, the amount of endowment, and the exchange rate threshold).

As the first step, by Lemma 4, it suffices for Algorithm 1 to work separately on pairs τ_1 and τ_j for every $j \in [2:n]$. Fix a $j \in [2:n]$ below. By Lemma 5, it suffices to consider strategies that only insert attacking transactions in one direction. So Algorithm 1 tries both directions between τ_1 and τ_j and pick the best one. Take the direction $\tau_1 \to \tau_j$ as an example. Again, by Lemma 5, we can throw away all users' transactions that are for $\tau_1 \to \tau_j$ direction. Now the question reduces to the following: Given a set of users' transactions $\{\textsc{Batch}^i\}_{i\in S_{1\to j}}$ such that every $\textsc{Batch}^i$ is in the direction $\tau_1 \to \tau_j$, what is the optimal way for the mediator to select a subset of users' transactions and insert its own transactions in the opposite direction, *i.e.*, the $\tau_j \to \tau_1$ direction?

Algorithm 1: Optimal MEV Strategy for Batch Transactions that Form a Fisher Market

```
Input: A set of users' transactions $\{\textsc{Batch}^i\}_{i\in[m]}$ that forms a Fisher market and a set of
       exogenous prices $\{p_i^*\}_{i\in[n]}$.
Output: A strategy for the mediator that obtains optimal profits.
1  Without loss of generality, assume that every transaction trades between $\tau_1$ and $\tau_j$ for
    $j \in [2:n]$, i.e., $\tau_1$ is the special token.
2  for each $j$ from 2 to $n$ do
       // Work on the direction $\tau_1 \to \tau_j$.
3      Let $S_{1\to j} \subseteq [m]$ be the set of indices $i$ such that $\textsc{Batch}^i$ is in the direction $\tau_1 \to \tau_j$.
4      Sort transactions in $S_{1\to j}$ in an ascending order w.r.t. their exchange rate thresholds
        (break tie arbitrarily). Let $\pi_1$ be such an order and denote the $k$-th transaction in the
        order as $\textsc{Batch}^{\pi_1(k)} = (\tau_1 \to \tau_j, \delta_{\tau_1}^{\pi_1(k)}, r^{\pi_1(k)})$.
5      Let $k_1 \in \arg\max_{k_1\in[|S_{1\to j}|]} \left\{ \left(\sum_{k\in[k_1]} \delta_{\tau_1}^{\pi_1(k)}\right) \cdot \left(p_1^* - r^{\pi_1(k_1)} \cdot p_j^*\right) \right\}$.
6      Let $\textsc{profit}_1$ be the value corresponding to $k_1$.

       // Work on the direction $\tau_j \to \tau_1$.
7      Let $S_{j\to 1} \subseteq [m]$ be the set of indices $i$ such that $\textsc{Batch}^i$ is in the direction $\tau_j \to \tau_1$.
8      Sort transactions in $S_{j\to 1}$ in an ascending order w.r.t. their exchange rate thresholds
        (break tie arbitrarily). Let $\pi_2$ be such an order and denote the $k$-th transaction in the
        order as $\textsc{Batch}^{\pi_2(k)} = (\tau_j \to \tau_1, \delta_{\tau_j}^{\pi_2(k)}, r^{\pi_2(k)})$.
9      Let $k_2 \in \arg\max_{k_2\in[|S_{j\to 1}|]} \left\{ \left(\sum_{k\in[k_2]} \delta_{\tau_j}^{\pi_2(k)}\right) \cdot \left(p_j^* - r^{\pi_2(k_2)} \cdot p_1^*\right) \right\}$.
10     Let $\textsc{Profit}_2$ be the value corresponding to $k_2$.

11     if $\textsc{Profit}_1 \le 0$ & $\textsc{Profit}_2 \le 0$ then
12         Do nothing.
13     else if $\textsc{Profit}_1 \ge \textsc{Profit}_2$ then
14         Include all $\left\{\textsc{Batch}^{\pi_1(k)}\right\}_{k\in[k_1]}$ transactions;
15         Insert one mediator's Batch $\left(\tau_j \to \tau_1, \left(\sum_{k\in[k_1]} \delta_{\tau_1}^{\pi_1(k)}\right) \cdot r^{\pi_1(k_1)}, \frac{1}{r^{\pi_1(k_1)}}\right)$.
16     else
17         Include all $\left\{\textsc{Batch}^{\pi_2(k)}\right\}_{k\in[k_2]}$ transactions;
18         Insert one mediator's Batch $\left(\tau_1 \to \tau_j, \left(\sum_{k\in[k_2]} \delta_{\tau_j}^{\pi_2(k)}\right) \cdot r^{\pi_2(k_2)}, \frac{1}{r^{\pi_2(k_2)}}\right)$.
```

Obviously, we want to control the final prices to satisfy $p_1/p_j \le p_1^*/p_j^*$ (otherwise the mediator will loss profit), thus we should delete (and ignore) users' transactions Batch^i such that $r^i > p_1^*/p_j^*$. Suppose that the final exchange rate is $p_1/p_j = r \le p_1^*/p_j^*$, then for each user's transaction Batch^i such that $r^i \le r$, we can obtain profit $\delta_{\tau_1}^i \cdot (p_1^* - r \cdot p_j^*) \ge 0$. Thus we should include all users' transactions Batch^i such that $r^i \le r$.

Now the correct way to attack them seems ready to come out: sort all user transactions in an ascending order π with respect to their exchange rate thresholds. Then the algorithm enumerates each involved threshold and calculates the corresponding profit:

$$\textsc{Profit}(k) = \left(\sum_{k' \in [k]} \delta_{\tau_1}^{\pi(k')}\right) \cdot (p_1^* - r^{\pi(k)} \cdot p_j^*), \tag{7}$$

where $r^{\pi(k)}$ is exchange rate threshold of the k-th user transaction and $\delta_{\tau_1}^{\pi(k')}$ is the endowment of the k'-th transaction in the order π. This $\textsc{Profit}(k)$ is obtained by setting the threshold of the k-th transaction as the exchange rate in this direction, selecting all user transactions with thresholds no larger than that, and inserting an attacking transaction in the opposite direction to provide the exact amount of token τ_j they need, which is $r^{\pi(k)} \cdot \left(\sum_{k' \in [k]} \delta_{\tau_1}^{\pi(k')}\right)$. Let $\textsc{Profit}_1$ and k_1 be the maximal profit and corresponding index for the direction $\tau_1 \to \tau_j$. Symmetrically, the algorithm works on the direction $\tau_j \to \tau_1$ and obtains the $\textsc{Profit}_2$ and k_2. If both $\textsc{Profit}_1$ and $\textsc{Profit}_2$ are no larger than 0, we just ignore all user transactions between π_1 and π_j and insert no transaction between them, as neither direction is profitable. Otherwise, choosing the strategy with the highest profit.

It is easy to verify that all transactions in the batch will be successfully executed. This finishes the proof.

□

4.5 Optimal MEV under General Arrow-Debreu Market

In this section, we show the computational hardness of finding an optimal MEV strategy when the user transactions form a general Arrow-Debreu market.

Theorem 3. *Given a set of users' transactions $\left\{\textsc{Batch}^i\right\}_{i\in[m]}$ such that they form a general Arrow-Debreu market, it is NP-hard to compute an optimal strategy.*

Proof. We reduce the NP-hard Feedback Arc Set Problem to our MEV optimization problem. Recall that an instance of the feedback arc set problem contains a directed graph and asks to find an acyclic subgraph with a maximum number of edges.

Suppose we are given an arbitrary graph $G = (V, E)$ where V is a set of $|V| = n$ vertices and E is a set of $|E| = m$ directed edges (there are no multiple edges with the same source and target nodes). Note that we can decide if a graph is acyclic in polynomial time, and if the graph is acyclic, we can simply output $|E|$ as the answer of the given instance of the Feedback Arc Set Problem. Thus, without loss of generality, we assume that the graph is not acyclic, which implies the maximum acyclic subgraph of G has $q < m$ edges.

Then, we construct the MEV optimization problem as follows. For each vertex $v_i \in V$, we construct a token τ_i and set its exogenous price p_i^* as 1. For each edge $(i,j) \in E$, we construct a user transaction $\textsc{Batch} = \left(\tau_i \to \tau_j, m^m, \frac{1}{m^m}\right)$. We refer the constructed instance to $\left\{\textsc{Batch}^{(i,j)}\right\}_{(i,j)\in E}$.

The correctness of this reduction follows from the following two lemmas.

Lemma 6. *If the maximum acyclic subgraph of G has q edges, then the mediator's optimal profit under the instance $\left\{\textsc{Batch}^{(i,j)}\right\}_{(i,j)\in E}$ lies in $\left[\left(m^m - m^{m-1}\right) \cdot q, m^m \cdot q\right]$.*

Proof. Given Lemma 3, the upper bound proof is easy. Consider any mediator's strategy $\left(S, \left\{\textsc{Batch}^i\right\}_{i\in[m+1:m+k]}\right)$ *such that the economy graph of* $\left\{\textsc{Batch}^i\right\}_{i\in S}$ *is acyclic*, so we have

$|S| \leq q$. The requirement of acyclic graph is without loss of generality due to Lemma 3. For each token in the instance, its demand under a market equilibrium equals the supply. The mediator's profit is the value of received tokens in the final allocation minus the value of its initial endowments, which is upper bounded by the value of users' initial endowments, *i.e.*, $1 \cdot m^m \cdot q$.

The lower bound proof is more involved and we focus on it next.

Assume that $G' = (V', E')$ is a directed acyclic subgraph of G that has q edges. We construct a strategy for the mediator that can obtain at least $(m^m - m^{m-1}) \cdot q$ profit. The construction is based on *levels* of nodes in G', which we define next. We start with all nodes that have no incoming edges and label them as level 1; then we (virtually) remove the level-1 node(s) and all edges taking them as the source node; next, we label all nodes that have no incoming edges as level 2 and repeatedly process the remaining subgraph with an increasing level (*i.e.*, level 3, 4, $\cdots$) until all nodes are labeled. In this way, all selected user transactions start from a token with a lower level and end at a token with a higher level.

Now we are able to describe the mediator's strategy: First, the mediator selects all users' transactions that correspond to edges in E'. Then for each selected user transaction $\textsc{Batch} = \left(\tau_i \to \tau_j, m^m, \frac{1}{m^m}\right)$ in G', the mediator inserts an attacking transaction $\textsc{Batch}'$ in its opposite direction, specifically, $\textsc{Batch}' = \left(\tau_j \to \tau_i, m^{m+l_i-l_j}, m^{l_j-l_i}\right)$ where l_i is the level of node i (*i.e.*, token τ_i). The two steps derive an attacking strategy. Next, we prove that these selected user transactions and newly inserted attacking transactions are executed under equilibrium, and bring a profit no less than $\left(m^m - m^{m-1}\right) \cdot q$.

Let $\mathbf{p}$ be a price vector of tokens in G', where token τ_i's price $p_i = m^{l_i}$. It is easy to verify that $\mathbf{p}$ is the price equilibrium where the unique allocation is as follows: Each selected user transaction $\textsc{Batch} = \left(\tau_i \to \tau_j, m^m, \frac{1}{m^m}\right)$ is executed at the exchange rate $m^{l_i-l_j}$ which is larger than its requirement $\frac{1}{m^m}$, and receives $m^{m+l_i-l_j}$ amount of token τ_j; Each attacking transaction in the opposite direction $\textsc{Batch}' = \left(\tau_j \to \tau_i, m^{m+l_i-l_j}, m^{l_j-l_i}\right)$ receives m^m amount of token τ_i, bringing a profit of $m^m - m^{m+l_i-l_j} \geq m^m - m^{m-1}$. Combining that we insert q attacking transactions in total, the mediator's profit is at least by $\left(m^m - m^{m-1}\right) \cdot q$. □

Lemma 7. *If the mediator's optimal profit is in* $\left[\left(m^m - m^{m-1}\right) \cdot q, m^m \cdot q\right]$*, then the number of edges in the maximum acyclic subgraph of* G *is* q*.*

Proof. The proof in fact follows from the disjointedness of $\left[\left(m^m - m^{m-1}\right) \cdot q, (m^m - 1) \cdot q\right]$ for different q. Specifically, we show the upper bound for $q-1$ interval is strictly smaller than the lower bound of q interval, which follows from the following simple calculation.

$$q < m \quad \Rightarrow \quad mq - q > mq - m \quad \Rightarrow \quad (m-1)qm^{m-1} > m(q-1)m^{m-1}$$
$$\Rightarrow \quad \left(m^m - m^{m-1}\right) \cdot q > m^m \cdot (q-1).$$

□

Overall, deciding which interval the mediator's optimal profit lies in is equivalent to finding the size of maximum acyclic subgraph of G. So finding an optimal strategy is as hard as solving the Feedback Arc Set Problem, which is NP-hard.

This finishes the proof. □

5 Discussion and Open Problems

Approximation Algorithm for AMMs. The MEV optimization problem in fact subsumes the famous sandwich attack, which has attracted much attention in both academia and industry. However, as our main theorem in Section 3 showed, it is NP-hard to compute an optimal attack. On the one hand, this rules out the possibility of having *exact* efficient optimal algorithm. On the other hand, it naturally raises the question of studying the *approximation algorithms.* Note that the Partition problem that we reduce from is weakly NP-hard, which admits fully polynomial time approximation scheme (FPTAS). Although the MEV optimization problem has more structure than knapsack problem thus is seemingly harder than that (maybe closer to scheduling problems), we still think an efficient approximation algorithm with a good approximation guarantee exists. We leave this as the first concrete open problem.

Hardness of Approximation for Batch Auctions. As we discussed before, the proof steps used in our approach suggest that it may remain NP-hard even if we want a polynomial-time algorithm with good approximation guarantee. In particular, Lemma 6 and Lemma 7 together show that even if approximating the solution with a roughly additive $\frac{1}{m}$ fraction would imply P = NP. However, we conjecture that the stronger hardness of approximation should also hold. While both MEV optimization problems are NP-hard in the worst-case analysis, showing distinct results in terms of approximation would also give us new insights to understand their different levels of vulnerability.

Manipulating Market. We provided a polynomial-time algorithm to attack a set of transactions that form a Fisher market. It is also possible to study the same market equilibrium manipulation problem for the general linear markets (where the utility functions may have more than two non-zero coefficients). In particular, we are curious if our ideas could be extended to general linear Fisher markets to obtain a polynomial-time optimal-attack algorithm. If so, this could be a new point of view about the fundamental differences between Fisher markets and Arrow-Debreu markets.

Empirical MEV on BATCH *Auctions.* The theoretical results of computational complexity are fundamental, but our interest of course extends to practical environments MEV, as decentralized exchanges are happening in blockchain everyday with incredible volumes. Even though optimal attack is generally an NP-hard problem, it is still promising to design heuristic algorithms to extract MEV; the possibilities of such novel MEV behaviors were not discussed by the community before. Technically speaking, our combinatorial structure still provides clear insights about how to narrow the strategy space for an attack to design strategies. However, to make the whole algorithm practical, there is still a large space for the design of robust and truly practical algorithms (with very fast running time). This is beyond the scope of the current work, but we are expecting emerging works related to MEV in batch auctions in the near future.

References

[1] DeFiprime, "Decentralized exchanges trading volume," Feb 11th 2024. [Online]. Available: https://defiprime.com/dex-volume

[2] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, and D. Robinson, "Uniswap v3 core," *Tech. rep., Uniswap, Tech. Rep.*, 2021. [Online]. Available: https://uniswap.org/whitepaper-v3.pdf

[3] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and

consensus instability," in *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*. IEEE, 2020, pp. 910–927. [Online]. Available: https://doi.org/10.1109/SP40000.2020.00040

[4] Flashbots, "Mev-explorev1," Feb 11th 2024. [Online]. Available: https://explore.flashbots.net/

[5] ——, "Flashbots transparency dashboard," Feb 11th 2024. [Online]. Available: https://transparency.flashbots.net/

[6] L. Zhou, K. Qin, C. F. Torres, D. V. Le, and A. Gervais, "High-frequency trading on decentralized on-chain exchanges," in *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 2021, pp. 428–445. [Online]. Available: https://doi.org/10.1109/SP40001.2021.00027

[7] L. Zhou, K. Qin, A. Cully, B. Livshits, and A. Gervais, "On the just-in-time discovery of profit-generating transactions in defi protocols," in *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 2021, pp. 919–936. [Online]. Available: https://doi.org/10.1109/SP40001.2021.00113

[8] G. Ramseyer, A. Goel, and D. Mazières, "SPEEDEX: A scalable, parallelizable, and economically efficient decentralized exchange," in *20th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2023, Boston, MA, April 17-19, 2023*, M. Balakrishnan and M. Ghobadi, Eds. USENIX Association, 2023, pp. 849–875. [Online]. Available: https://www.usenix.org/conference/nsdi23/presentation/ramseyer

[9] K. Qin, L. Zhou, and A. Gervais, "Quantifying blockchain extractable value: How dark is the forest?" in *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*. IEEE, 2022, pp. 198–214. [Online]. Available: https://doi.org/10.1109/SP46214.2022.9833734

[10] M. Bartoletti, J. H. Chiang, and A. 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*, ser. Lecture Notes in Computer Science, I. Eyal and J. A. Garay, Eds., vol. 13411. Springer, 2022, pp. 3–19. [Online]. Available: https://doi.org/10.1007/978-3-031-18283-9_1

[11] Y. Wang, Y. Chen, H. Wu, L. Zhou, S. Deng, and R. Wattenhofer, "Cyclic arbitrage in decentralized exchanges," in *Companion of The Web Conference 2022, Virtual Event / Lyon, France, April 25 - 29, 2022*, F. Laforest, R. Troncy, E. Simperl, D. Agarwal, A. Gionis, I. Herman, and L. Médini, Eds. ACM, 2022, pp. 12–19. [Online]. Available: https://doi.org/10.1145/3487553.3524201

[12] C. F. Torres, R. Camino, and R. State, "Frontrunner jones and the raiders of the dark forest: An empirical study of frontrunning on the ethereum blockchain," in *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, M. D. Bailey and R. Greenstadt, Eds. USENIX Association, 2021, pp. 1343–1359. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/torres

[13] Z. Li, J. Li, Z. He, X. Luo, T. Wang, X. Ni, W. Yang, X. Chen, and T. Chen, "Demystifying defi MEV activities in flashbots bundle," in *Proceedings of the 2023 ACM SIGSAC Conference*

on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023, W. Meng, C. D. Jensen, C. Cremers, and E. Kirda, Eds. ACM, 2023, pp. 165–179. [Online]. Available: https://doi.org/10.1145/3576915.3616590

[14] A. Obadia, A. Salles, L. Sankar, T. Chitra, V. Chellani, and P. Daian, "Unity is strength: A formalization of cross-domain maximal extractable value," *CoRR*, vol. abs/2112.01472, 2021. [Online]. Available: https://arxiv.org/abs/2112.01472

[15] L. Heimbach and R. Wattenhofer, "Eliminating sandwich attacks with the help of game theory," in *ASIA CCS '22: ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022*, Y. Suga, K. Sakurai, X. Ding, and K. Sako, Eds. ACM, 2022, pp. 153–167. [Online]. Available: https://doi.org/10.1145/3488932.3517390

[16] Y. Li, M. Zhang, J. Li, E. Chen, X. Chen, and X. Deng, "MEV makes everyone happy under greedy sequencing rule," in *Proceedings of the 2023 Workshop on Decentralized Finance and Security, DeFi 2023, Copenhagen, Denmark, 30 November 2023*, K. Qin and F. Zhang, Eds. ACM, 2023, pp. 9–15. [Online]. Available: https://doi.org/10.1145/3605768.3623543

[17] C. Team, "Cow protocol overview," September 20th 2023. [Online]. Available: https://docs.cow.fi/

[18] Flashbots, "Cowswap solver metrics," September 20th 2023. [Online]. Available: https://dune.com/flashbots/cowswap-solver-metrics

[19] K. J. Arrow and G. Debreu, "Existence of an equilibrium for a competitive economy," *Econometrica: Journal of the Econometric Society*, pp. 265–290, 1954.

[20] D. Gale, "The linear exchange model," *Journal of Mathematical economics*, vol. 3, no. 2, pp. 205–209, 1976.

[21] B. Codenotti, A. Saberi, K. Varadarajan, and Y. Ye, "Leontief economies encode nonzero sum two-player games," in *SODA*, vol. 6, 2006, pp. 659–667.

[22] X. Chen, D. Paparas, and M. Yannakakis, "The complexity of non-monotone markets," *J. ACM*, vol. 64, no. 3, pp. 20:1–20:56, 2017. [Online]. Available: https://doi.org/10.1145/3064810

[23] R. R. Maxfield, "General equilibrium and the theory of directed graphs," *Journal of Mathematical Economics*, vol. 27, no. 1, pp. 23–51, 1997.