Agent-Based Modeling of Blockchain Decentralized Financial Protocols

Igor Struchkov, Alexey Lukashin, Bogdan Kuznetsov
Peter the Great St. Petersburg Polytechnic University
Saint-Petersburg, Russia
struchkov_iv@spbstu.ru, alexey.lukashin@spbstu.ru, kuznetsov.bp@edu.spbstu.ru

Igor Mikhalev
University of Amsterdam
Amsterdam, the Netherlands
i.y.mikhalev@uva.nl

Zoia Mandrusova
Wrike Inc.
Saint-Petersburg, Russia
mandrusova@mail.ru

***Abstract*—Decentralized financial applications running on blockchains using smart contracts have attracted a lot of attention recently. One important class of such applications is decentralized digital asset exchanges. In this paper we present an agent-based modeling approach for decentralized exchanges that allowed us to achieve realistic results both in normal and stress market conditions and also investigate the impact of front runners on the distribution of profits. We also compare the results of the two exchanges - Uniswap and Liquifi - to evaluate the effect of the proposed solution for the price slippage and front running problems.**

I. Introduction

Decentralized financial applications running on blockchains are attracting a lot of attention nowadays [1]. A blockchain is a distributed append-only timestamped data structure. The data in blockchains is persisted on numerous network nodes (or peers) in the form of transactions. Transactions are being signed by submitting participants and then distributed among the network nodes using a certain consensus mechanism, such as Proof-of-work or Proof-of-stake [2]. Another important element of modern blockchain technologies is a smart contract - a special computer program run on every blockchain node in order to perform complex transactions. Decentralized applications that are built using smart contracts, operate in full automatic mode, without any need or even possibility of human intervention. Therefore, their users do not need to trust any party when performing financial transactions.

One important class of decentralized financial applications is decentralized digital asset exchanges (DEXes). DEXes are used to buy digital assets (tokens) of one type selling digital assets of some other type. DEXes usually fall into two types: order book exchanges and liquidity pool exchanges.

- Liquidity pool DEXes attract assets as investments from liquidity providers to form pools where everybody can exchange assets without the need of other participants. Automatic market maker (AMM) algorithms codified in smart contracts define exchange prices that depend on asset amount ratios in the pools.
- Another category, order book DEXes do not have liquidity pools. They process trading orders mostly as traditional exchanges do. A common drawback of order book DEXes is that they usually cannot be implemented within a smart contract because of high performance requirements that are hard to fulfill on a blockchain platform.

We can conclude that the liquidity pool DEX type is the most appropriate solution as it can be implemented purely on a blockchain within smart contracts. However, DEXes of this type also suffer from the price slippage and front running problems. The problems come from the AMM nature itself as the exchange price is being changed by every transaction. Moreover, the larger the transaction size the greater is the price slippage impact. The price slippage also leads to front running attacks possibility - as some network participants may intercept large transactions and win profits acting against them. These problems and possible solutions are also discussed in more details below.

As the impact of the price slippage and front running problems can vary depending on the AMM used and external market conditions, we can use simulation modeling to evaluate DEX performance in different scenarios. There are numerous works asserting that the agent-based modeling approach is very promising in simulation of financial markets behavior. In [3] a methodology is given for applying the agent-based approach to simulate stock markets. While some of the proposed models, e.g. trader models, can be adapted for decentralized markets, there are still many nuances that were not discussed there. One of the first attempts to simulate decentralized markets has been undertaken in [4]. In this work a deep analysis of the arbitrageurs behavior is presented and an optimization problem for the arbitrageur is formulated. It is also proven that for Uniswap [5] AMM the optimization problem is convex and has a closed-form solution. In [6] the optimal arbitrage problem is generalized to fit any constant function AMM. Although these works give a sufficient basis for decentralized markets modeling, some of the problems are still not investigated, specifically the front runners behavior and their impact on profits distribution among market participants. Also the abovementioned works lack an underlying blockchain model and therefore cannot predict dynamic DEX performance in rapidly changing market conditions, when block creation time and transactions order may cause significant changes in the exchange price.

As front running activities have a negative effect on the markets, the developers of decentralized financial applications search the ways to mitigate this problem. One possible direction was proposed in [7] based on virtual balances that do not change immediately, but wait for some period, e.g. until the end of a block, making it harder for the front runners to get the profit. Another solution was proposed in [8] and implemented in Liquifi DEX. The idea here is to allow long exchange operations with gradual liquidity flow. Front runners cannot take profits from acting against these long operations because they last much longer than one block so that trying to front run these operations makes no sense as other arbitrageurs will have enough time to correct the price.

Though decentralized financial application are rapidly growing in number, the research results in this area are still not sufficient. The existing works made the first steps to formalization of the agent-based simulation problem for decentralized applications and obtained results for some special cases. But there are still lots of open questions regarding realistic modeling of blockchain networks, digital markets and participants behavior. In this paper we present an extended agent-based modeling approach for decentralized exchanges that allowed us to achieve more realistic results both in normal and stress market conditions and also investigate the impact of front runners on the distribution of profits. We also compare the results of two DEXes - Uniswap and Liquifi - to evaluate the effect of the proposed solution to the price slippage and front running problems.

In section II we describe the basics of automatic market makers (AMMs) and give examples of the two most widely used AMMs - constant product market maker and constant mean market maker. In section III we describe in details our agent-based modeling approach, introduce agent models: a trader, an arbitrageur, a displacement front runner, an insertion front runner, a liquidity provider. We also provide models for a blockchain network and an external market. In section IV there are simulation results that we have got for the two decentralized exchanges (Uniswap and Liquifi) and discussion of these results.

II. Liquidity pool decentralized markets

A. Automatic market makers

An automatic market maker (AMM) is an algorithm, usually codified in a blockchain smart contract, that automatically calculates exchange prices of digital assets. In liquidity pool DEXes AMMs use digital assets amounts in the pools as the main input data to determine the price. Constant function market makers (CFMM) are the most common class of AMMs. These algorithms are based on a constant function that must be kept unchanged after every transaction.

Constant product market maker [9] is the simplest CFMM with the constant function of the form:

$$k = x \cdot y$$

where x and y are amounts of digital assets in the pool, k is some constant.

Uniswap [5] is the most popular DEX based on the constant product market maker.

Balancer [10] is another DEX that use a generalized CFMM - constant mean market maker:

$$V = \prod_t B_t^{W_t}$$

where t ranges over the tokens in the pool, B_t is the balance of the token in the pool, W_t is the normalized weight of the token, such that the sum of all normalized weights is 1, V is some constant.

The main difference of the constant mean market maker is that it supports more than 2 digital assets in one pool.

B. Price slippage

Fig. 1 illustrates how the constant product market maker works.

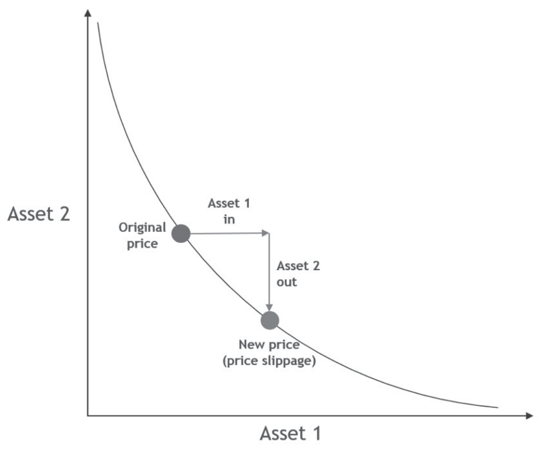

Fig. 1. Price slippage

When a trader adds some amount of asset 1 into the pool, the price of asset 1 decreases to keep the product constant. The problem is mostly significant for large deals or small liquidity pools.

C. Front running

Front running is a special type of attack when some participant, seeing an upcoming trading transaction, puts his own transaction ahead (playing with a transaction fee for example). In public blockchains front running attacks occur rather often because all the transactions are inherently visible to all network participants.

Front running attacks can be of different types:

- **A displacement front running attack** takes place when a front runner causes the transaction under attack to revert. Usually it is because traders set some price slippage tolerance limit. When a front-runner changes the price before the transaction, the limit makes the initial transaction fail.

- **An insertion front running attack** takes place when a front runner does not want the attacked transaction to revert. Instead, the front runner inserts his transaction before the attacked one, making it less profitable due to price slippage, and then makes an opposite transaction to get the profit from the changed price.

III. Modeling approach

The agent-based approach is based on modeling behavior of all major market participants. In our scenario we identify the following agents:

- **Traders** are primary users of a decentralized market. In terms of our model, the only goal of a trader is to sell some amount of a digital asset of one type to buy a corresponding amount of a digital asset of another type. Traders usually seek for the best possible exchange price and have some tolerance for price slippage.
- **Arbitrageurs** are professional market players that seek for price disbalances between several markets that give them opportunities to get profits from buying some asset on one market and selling the same asset on the other market. Though arbitrageurs' profits are entirely speculative, they play an important and useful role on decentralized markets, and especially on liquidity pool markets. While decentralized automatic market makers usually do not use any external sources to determine the exchange prices, arbitrageurs help to adjust the prices to common market levels. We will refer to this type of arbitrageurs as "honest" arbitrageurs.
- **Front runners** are somewhat similar to arbitrageurs, but they get profits not from price disbalances between markets, but from knowledge about the upcoming trades on one market. Front runners then use blockchain specific means to place their transactions in front of the transactions under attack. Therefore, front runners do not play any useful role and just take profits away of other market participants.
- **Liquidity providers** are essential participants of liquidity pool markets. They invest their own assets into liquidity pools to earn exchange fees from every transaction on the market.

A. Trader model

According to [3] the trader behavior can be modeled in different ways:

- **A zero-intelligence trader** makes his transactions randomly, not depending on the current market situation.
- **A fundamentalist trader** makes decisions based on the fundamental value of an asset.
- **A trader based on the historical information** makes decisions based on a comparison between the average of the short-term price and average of the long-term price to detect the trend.

The choice of a trader model depends on the simulation goals. In our scenario the main goal is to investigate the impact of front runners on the distribution of profits. Considering this, the zero-intelligence trader model is quite suitable as we are interested mostly in price slippage measurements and not in evaluation of trading strategies, which may be a part of a future work.

B. Arbitrageur model

Arbitrageurs perform trades between the simulated market and the external market when prices differ. Given two digital assets, A and B, the arbitrageur seeks to maximize the profit made from trading, for example, some amount of loaned asset B, Δ_B to some amount of asset A Δ_A via the simulated market. The arbitrageur then trades back the received Δ_A for Δ'_B and pays back the loan Δ_B to receive profit $\Delta'_B - \Delta_B$.

Therefore, the arbitrageur solves the following optimization problem:

$$\begin{gathered} max_{\Delta_A,\Delta_B} \quad m_p\Delta_A - \Delta_B \\ \Delta_A, \Delta_B \geq 0 \\ CF(\Delta_A, \Delta_B) = k \end{gathered}$$

where m_p is the external market price of asset A, $CF(\Delta_A, \Delta_B)$ is the market maker constant function.

For the constant product market maker the equation $CF(\Delta_A, \Delta_B) = k$ takes the following form:

$$(R_A - \Delta_A)(R_B + \gamma\Delta_B) = k$$

where R_A is the initial reserve of asset A in the pool, R_B is the initial reserve of asset B in the pool, $\gamma = 1 - \alpha$, α is the trading fee of the simulated market.

In [4] it is proven that this optimization problem is convex and can be rewritten as a closed form expression for the optimal solution:

$$\Delta_A^* = \left(R_A - \sqrt{\frac{k}{\gamma m_p}}\right)_+$$

where $(x)_+ = max\{x, 0\}$ for $x \in \mathbf{R}$.

For the constant mean market maker the equation $CF(\Delta_A, \Delta_B) = k$ takes the following form:

$$(R_A - \Delta_A)^{W_A}(R_B + \gamma\Delta_B)^{W_B} = k$$

where W_A and W_B are the weights of assets A and B in the pool respectively.

C. Front runner models

1) Displacement front runner: A displacement front-runner in our scenario acts against "honest" arbitrageurs. The front runner monitors blockchain transactions and, when an arbitrage transaction is detected, he places a similar transaction but with higher transaction price, increasing its priority when forming a new block. When this tactics succeeds, the attacked ”honest” arbitrage transaction is reverted due to the slippage tolerance limit and the front runner gets the profit.

Network delays are modeled by a front runner activation timeout meaning that the front runner detects a new transaction not immediately but after a random waiting time. That is why some ”honest” arbitrage transactions may still succeed when, for example, they occur very close to the end of a block, so that the front runner does not have enough time to place his transaction before the block is committed.

2) Insertion front runner: An insertion front-runner acts against traders. This front runner monitors trading transactions and estimates the price slippage. The front runner also has access to the information about the slippage tolerance of the attacked transaction. Using this information, the front runner can calculate the maximum transaction size that will not exceed the trader’s slippage tolerance. Then the front runner places his transaction ahead of the trader’s. The trader still accomplishes his transaction but at the worst possible price. After that the front runner places a transaction in the opposite direction and sells back the assets at better price.

D. Liquidity provider model

Liquidity providers invest their assets into liquidity pools to earn fees taken from each transaction. Usually it is required to hold the assets in a pool for some time to gather enough fees. On the other hand, in this work we focus on the short-term effects. That is why in our scenario we do not model any activity from the side of liquidity providers. The liquidity providers in our case invest some predefined amount of assets in the beginning of the simulation and hold them without changes until the simulation ends.

In some situations, especially, in rapidly changing market conditions, actions of the liquidity providers may still have short-term effects. This may be further investigated in our future works.

E. Blockchain model

The most important aspect of blockchain operation in our scenario is forming of blocks and transaction ordering in the blocks. Therefore, we use a simplified blockchain model where all the new transactions are included in a list of pending transactions that is sorted by the transaction price proposed in each transaction. As soon as enough transactions to form a new block arrive, the block is created and an appropriate event is fired.

Some aspects that we have left aside by now are network delays and possibility of alternative chains at different nodes. The first aspect is approximately modeled by random delays of each agent. These delays correspond to the message propagation delays in the real blockchain network. The second aspect (alternative chains), though very important for the consistency of the whole blockchain network, does not have a significant impact on the simulation results as eventually only one chain will be chosen by all the participants.

F. External market model

We update the market price every time step (after all agents have completed their actions) in the following way:

$$m_p \to m_p \cdot e^{\sigma X + \mu}$$

where $X \in N(0,1)$ is drawn from a normal distribution and $\mu, \sigma \in \mathbf{R}$ represent the mean returns and volatility of the market when no trades are performed.

When exchange operations are performed, the external market follows a simple power law model where the price m_p of some token A, is updated in the following way:

$$m_p \to m_p + \kappa \Delta_A^{1+\xi}$$

where $\kappa \geq 0$ and $\xi \geq 0$ are given in the problem data.

In reality external markets are often large enough compared to the size of the modeled transactions. This also allows us in most cases to take an assumption of an infinitely large external market where the simulated transactions do not affect the market price.

G. Simulation tool

To run the simulation experiments we use SimPy simulation tool [11]. It is a lightweight discrete event simulation library written in Python programming language.

IV. Simulation results

We simulate transactions for ETH/USDT token pair on decentralized exchanges using the method described above, based on two AMMs: Uniswap and Liquifi. All rules for changing states of these AMMs are taken from the whitepapers, and the change of the external market price is generated randomly. Although both exchanges use the constant product market maker AMM, Liquifi introduces a special type of long time-locked operations that are supposed to be a cure to the price slippage and insertion front running problems. Using the simulation, we can confirm our hypotheses about the different distribution of revenues at each exchange due to their peculiarities. In Figures 2 and 3 we can observe the key difference between the selected AMMs. Specifically, the ETH/USDT token pair price change in the Liquifi model is a sloping line with corrections done by market agents, this is a result of the time-locked operations described in details in the Liquifi whitepaper. In the Uniswap model, the price changes stepwise as a result of swap operations by market agents. We can see that a large exchange operation that occured at time 100 on the Uniswap market (fig. 2) caused a significant price leap below 850, while on the Liquifi market the opertion of the same volume, but stretched in time, did not cause the price to go lower than 990 (fig. 3). This confirms that the time-locked operations effectively reduce the price slippage.

Fig. 2 plots the ETH/USDT price change during the simulation time for the Uniswap exchange. The abscissa axis shows the simulation time from 0 to 2000 seconds. The ordinate axis expresses the price of the token pair in USDT. The solid line shows the change in the price of the token pair according to the

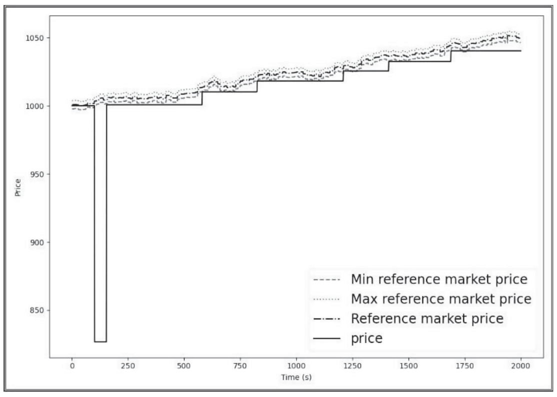

Fig. 2. Uniswap AMM price chart

Uniswap AMM. The dotted lines show the mean, minimum and maximum prices of the reference market.

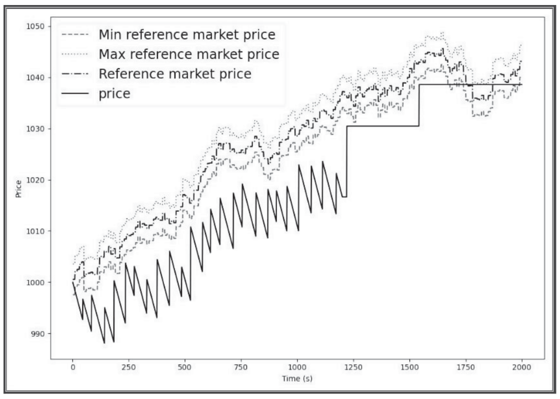

Fig. 3. Liquifi AMM price chart

Fig. 3 plots the ETH/USDT price change during the simulation time for the Liquifi exchange. The abscissa axis shows the simulation time from 0 to 2000 seconds. The ordinate axis expresses the price of the token pair in USDT. The solid line shows the change in the price of the token pair according to the Liquifi AMM. The dotted lines show the mean, minimum and maximum prices of the reference market.

In our experiments, we analyze the dependency of the market agents (traders, arbitrageurs and front runners) profit amounts on the size of the liquidity pools for each AMM in normal and high volatility market conditions. To obtain representative data, we selected 6 simulation configurations for each AMM and ended up with 12 variations, then we performed 50 repetitions of each of them and took the average values. Below we use abbreviations to denote the size of the liquidity pools: 100 ETH / 100000 USDT is denoted by the letter S (small), 1000 ETH / 1000000 USDT is denoted by the letter M (medium), 10000 ETH / 10000000 USDT is denoted by the letter L (large). The experimental data obtained for the Uniswap market are shown in Table I and for the Liquifi market - in Table II.

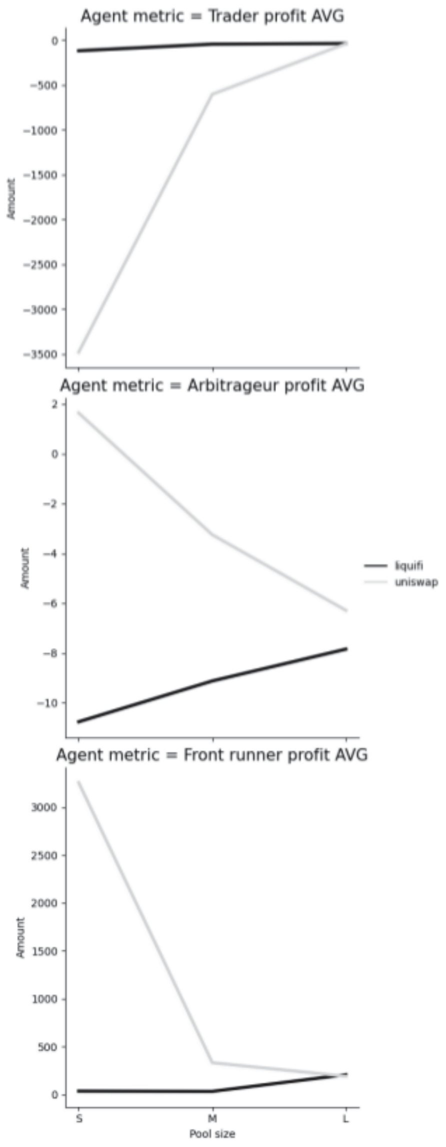

Fig. 4. Comparison of profits in normal market conditions

Table I shows the amounts of the Uniswap agents profits. The table shows the 3 types of AMM market agents: traders, arbitrageurs, front runners (both displacement and insertion). The results are converted to USDT and rounded for clarity.

Table II shows the amounts of the Liquifi agents profits. The table shows the 3 types of AMM market agents: traders, arbitrageurs, front runners (only displacement as insertion front running is not possible on the Liquifi market). The results are converted to USDT and rounded for clarity.

Fig. 4 shows the comparison of the market agents profit amounts at different liquidity pool sizes for Uniswap and Liquifi, under normal market conditions. By normal conditions we mean here the low value of the external market volatility ($\sigma = 0.001$).The abscissa axis shows the size of the liquidity pools. The ordinate axis expresses the profit in USDT. The

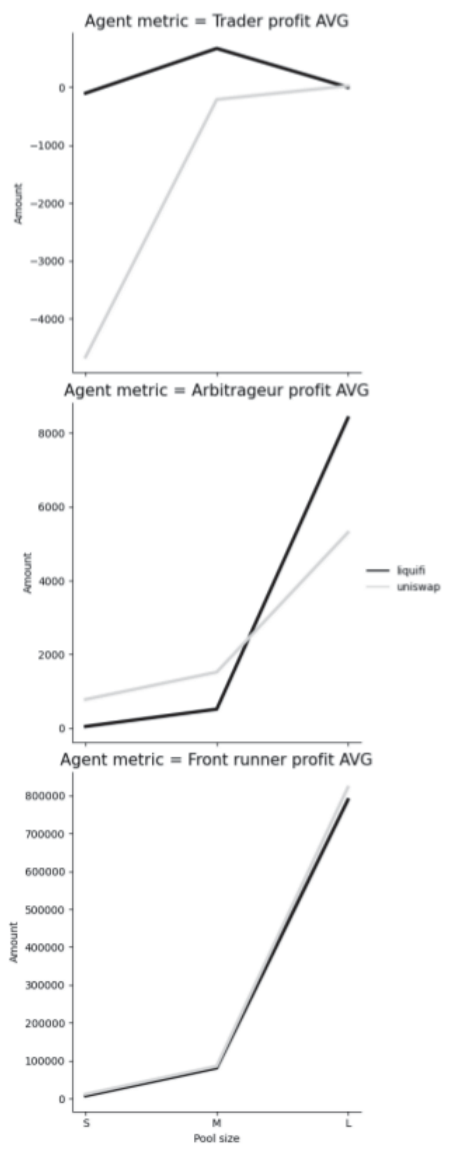

Fig. 5. Comparison of profits in critical market conditions

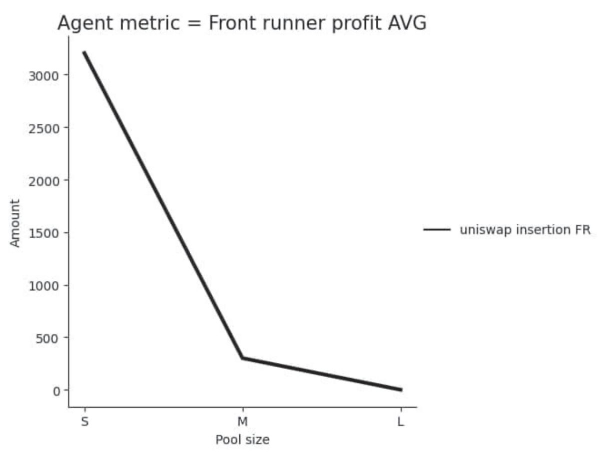

Fig. 6. Insertion front runner profits on Uniswap

TABLE I. MARKET AGENT PROFIT AMOUNTS FOR THE UNISWAP AMM

Market conditions	**Normal**		
Pool size	S	M	L
Trader	-3476	-602	-38
Arbitrageur	2	-3	-6
Front runner	3256	332	190
Market conditions	**Critical**		
Pool size	S	M	L
Trader	-4660	-208	27
Arbitrageur	782	1523	5304
Front runner	11970	86013	821816

black line corresponds to Liquifi, the gray line - to Uniswap.

Fig. 5 shows the comparison of the market agents profit amounts at different liquidity pool sizes for Uniswap and Liquifi, under critical market conditions. By critical conditions we mean here the high value of the external market volatility ($\sigma = 0.03$). The abscissa axis shows the size of the liquidity pools. The ordinate axis expresses the profit in USDT. The black line corresponds to Liquifi, the gray line - to Uniswap.

A. Discussion

When comparing the results of AMM simulation, we were able to find interesting trends depending on the size of liquidity pools. Liquifi shows a significant advantage in activity returns for the trader and the arbitrageur at the size of the liquidity pools M and S, the corresponding charts are shown in Fig. 4 and 5. When the size of the liquidity pool is large enough (L), the difference from Uniswap is reduced to zero. As a result of the simulation, we confirmed the hypothesis of increasing arbitrage and front running profits in critical market conditions. The results also show that the insertion front running technique cannot be applied to the Liquifi AMM because of the special time-locked operations. The Liquifi whitepaper describes the time-locked operation as a special mechanism that allows a user to split a large volume swap and execute it in parts over the time. The time-locked operation should make the swap more profitable for the trader, and counteract the actions of the insertion front runner. At the same time, the Uniswap market is vulnerable to the insertion front running attacks, the simulated insertion front runner profits depending on the pool size are shown on fig. 6.

Fig. 6 shows the dependency of the insertion front runner profits on the size of the pool for Uniswap. The abscissa axis shows the size of the liquidity pools. The ordinate axis expresses the profit in USDT.

V. CONCLUSION

In this paper we have presented our agent-based modeling approach and a simulation model implementation that we have applied to investigate the properties of blockchain decentralized financial applications - digital asset exchanges. Starting from the results found in the existing works, we

TABLE II. MARKET AGENT PROFIT AMOUNTS FOR THE LIQUIFI AMM

Market conditions	**Normal**		
Pool size	S	M	L
Trader	-116	-42	-35
Arbitrageur	-11	-9	-8
Front runner	36	32	207
Market conditions	**Critical**		
Pool size	S	M	L
Trader	-100	673	1
Arbitrageur	52	520	8406
Front runner	7605	81358	789090

have improved the modeling approach with a more realistic blockchain network model. We have also introduced two agent models that describe the behavior of front runners: a displacement front runner and an insertion front runner.

As a practical output we have presented simulation results for the two existing decentralized digital asset exchange protocols - Uniswap and Liquifi. We have shown how the simulation can be used to investigate the impact of arbitrageurs and front runners on profits distribution in different market conditions for different protocols.

As a general conclusion, we can state that the presented models can be used in practice to evaluate dynamic properties of various automatic market maker (AMM) algorithms, identify their advantages and drawbacks in different market scenarios.

Future work on this theme will allow us to extend the list of modeled decentralized exchanges, improve the external market model to be able to reproduce real world scenarios and validate our models on these realistic examples. Such realistic market models can be built based on statistical transactions data from public blockchain networks. We will also continue working on the blockchain network model to account for network and block confirmation delays, distribution of transactions between blocks, reproduce specific traits of different blockchain technologies.

REFERENCES

[1] F. Schaer, “Decentralized finance: On blockchain- and smart contract-based financial markets,” *Federal Reserve Bank of St. Louis*, 2021. [Online]. Available: https://research.stlouisfed.org/publications/review/2021/02/05/decentralized-finance-on-blockchain-and-smart-contract/-based-financial-markets

[2] F. Casino, T. K. Dasaklis, and C. Patsakis, “A systematic literature review of blockchain-based applications: Current status, classification and open issues,” *Telematics and Informatics*, vol. 36, pp. 55–81, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736585318306324

[3] M. A. Souissi, K. Bensaid, and R. Ellaia, “Multi-agent modeling and simulation of a stock market,” *Investment Management and Financial Innovations*, vol. 15(4), pp. 123–134, 2018. [Online]. Available: https://www.researchgate.net/publication/328840698_Multi-agent_modeling_and_simulation_of_a_stock_market

[4] G. Angeris, H.-T. Kao, R. Chiang, C. Noyes, and T. Chitra, “An analysis of uniswap markets,” *Cryptoeconomic Systems Journal*, 2019. [Online]. Available: https://ssrn.com/abstract=3602203

[5] *Uniswap v2 Core*. [Online]. Available: https://uniswap.org/whitepaper.pdf

[6] G. Angeris and T. Chitra, “Improved price oracles: Constant function market makers.” [Online]. Available: https://arxiv.org/abs/2003.10001

[7] *Improving front running resistance of x*y=k market makers*. [Online]. Available: https://ethresear.ch/t/improving-front-running-resistance-of-x-y-k-market-makers/1281

[8] *Liquifi Protocol*. [Online]. Available: https://liquifi.org/pdf/whitepaper-liquifi.pdf

[9] Y. Zhang, X. Chen, and D. Park, “Formal specification of constant product (x times y = k) market maker model and implementation,” 2018. [Online]. Available: https://github.com/runtimeverification/verified-smart-contracts/blob/uniswap/uniswap/x-y-k.pdf

[10] *A non-custodial portfolio manager, liquidity provider, and price sensor*. [Online]. Available: https://balancer.finance/whitepaper/

[11] *Documentation for SimPy*. [Online]. Available: https://simpy.readthedocs.io/en/latest/contents.html