A Multi Agent Society Model for Simulating Cryptocurrency Markets

Andrei Stelian Potra
Dipartimento di Fisica, Università degli Studi di Torino
(Date: June 14, 2023)

The aim of this project is to create a **society** of traders engaged in the exchange of a specific cryptocurrency, having a certain influence on its price. By utilizing a **Barabasi-Albert** network as the initial structure of this society, we demonstrate the ability to simulate cryptocurrency markets with regards to three **key phenomena**: the *distribution of fat-tail returns*, the *clustering of volatility*, and the *presence of a unit-root* property. Our objective is to illustrate that these outcomes can be achieved *without* employing genetic algorithms, but rather by incorporating agents with diverse beliefs that can yield profits and enhance their wealth. To simulate the dominance of certain entities in **cryptocurrency markets**, we introduce a hierarchical society of traders consisting of **fishes**, **sharks**, and **whales**, who select and trust each other forming a network of agents. By exploring model parameters we show how it is possible to get herding effects, rather than Wickoff distributions.

I. INTRODUCTION

In this study, we utilized a Python framework called Mesa for agent-based modeling, along with the Python package multilevel-mesa, to facilitate the creation and management of a **dynamic network** of interconnected agents. In this network, all agents are represented as nodes, and their levels of trust are simulated through weighted links, ranging from 0 to 1. Additionally, we associated different types of agents with specific node types, which we will now provide a brief overview of:

- **Fish.** This agent is characterized by minimal intelligence, representing a "promoted" version of a random trader that can also be influenced by other traders decisions, appended to a "friends list". The goal is to simulate the influence from various sources, including platforms like Twitter, Telegram, Discord, Reddit, and other social networks, where individuals seek investment advice and form connections with unfamiliar individuals.
- **Shark.** This agent is able to understand if the market is experiencing a bullish or bearish scenario and tries to align with the trends according to most influential agents on the network. By incorporating this feature, we aim to simulate the possibility for anyone to check whale's addresses within a cryptocurrency market's blockchain (which is accessible for everyone).
- **Whale.** This agent possesses a greater initial wealth compared to others and has the ability to exert significant influence on the price with larger fluctuations. They are not constantly active and exhibit both a semi-periodic behavior characterized by accumulating positions and subsequently distributing them, and a fuzzy-logic behaviour.

We started building our model by implementing a network comprising only fishes, and subsequently introducing sharks and whales in a gradual manner. The network is initialized using the Barabasi-Albert algorithm where nodes are progressively added by **preferentially attachment** to other nodes with a higher number of edges.
Alongside the network, we initialize the weights of the trust matrix using a uniform distribution ranging from 0 to 1. The agents' wealth is assigned varying values based on their types, and **random activation** order is employed to introduce different order activations among agents at each timestep.
In first place the study of the model consists of a **qualitative description** of how degrees of freedom (DOFs) influence price dynamics. Subsequently, we conduct a **statistical analysis** to assess the similarities between this artificial model and a real market.

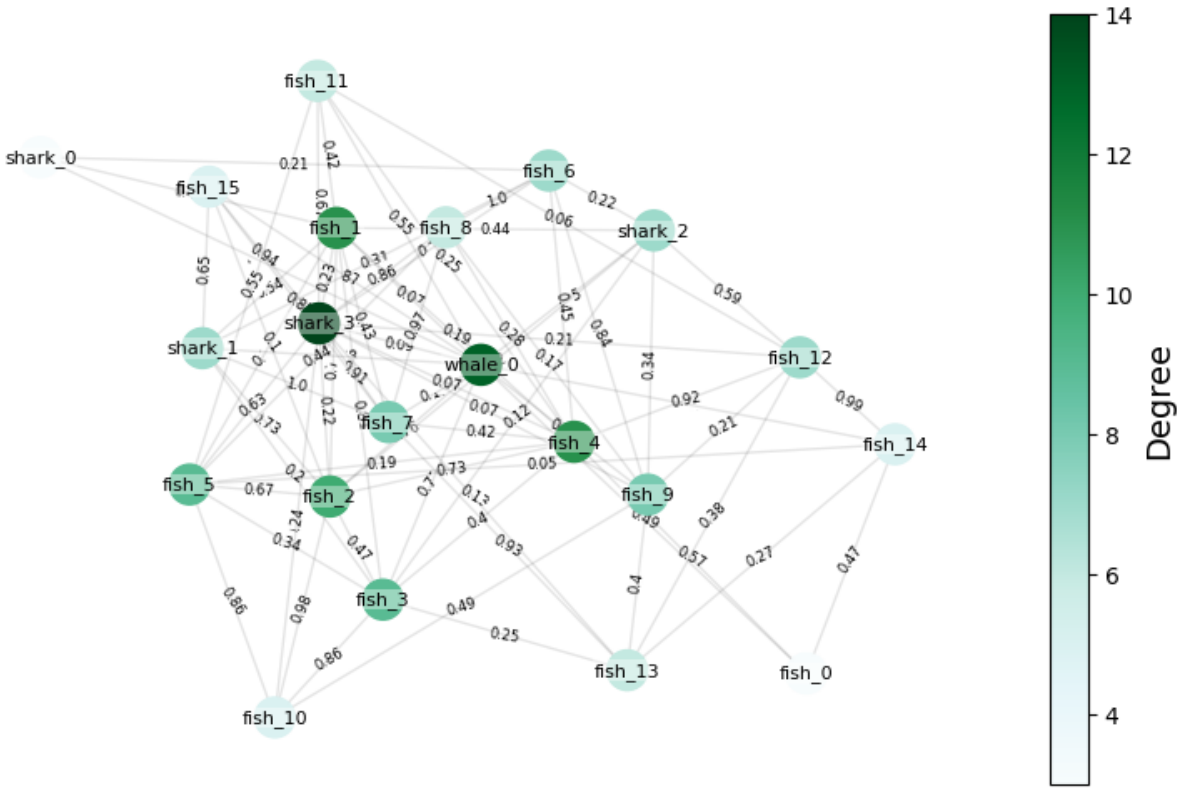

Figure 1. Example of a small network of traders with a whale being one of the most influent agents. Links between agents are weighted by a number from 0 to 1 and will update based on agents' profits.

II. AGENTS

Agents share certain common attributes and methods, but their beliefs vary based on their respective types. In this specific scenario, where agents are limited to buying or selling positions, it is common for agents to perform the **same action** but for **different reasons**. As a result, misunderstandings inevitably arise among agents, which will eventually impact on their level of trust.

Every agent in the model possesses a certain amount of available cash (e.g., USD), which they utilize to purchase positions within the market. In contrast to papers we inspired from, our approach differs in that agents have the capability to maintain **multiple** and diverse **positions**, with the freedom to sell one or some of them at any desired moment. These positions are stored within a **personal wallet**, containing information such as the position type ("long" or "short"), the asset price, and the specific model step in which the transaction occurred. To maintain simplicity, agents are limited to opening only one position per step, and there exists an upper limit imposed by the market on short orders to prevent negative prices from arising.

Agents must be able to calculate their profits and update their overall wealth, which comprises the sum of their cash holdings and the combined value of all positions stored within their wallet. Additionally, all agents have the possibility to select a sample of other agents, whom they consider as

neighbors, with the objective of simulating real-world scenarios where investors seek advice from specific groups of individuals.

A. Fish Agent

Fish agent is a great example of a social agent because he is a random trader with a social counterpart, in fact he has a method which allows him to check his social networks and people with his same interests, and consequentely he can add or remove friends with random periods. Each fish has:

- trust threshold and if a bond with a friend goes below it, then that friend is now removed from the friend list;
- probability to be influenced by its friends;
- probability to update the friends list.

We make the fish population agents heterogeneous by varying their initial wealth, by making some of them sleep randomly and by giving them different threshold for when going in survive mode, where fishes have low liquidity and critical wealth and try to sell some positions to regain it.

An example of decisional process. The fish agent starts his step function by checking conditions on sleep mode. If he is in sleep mode then all the following steps will be ignored, saving also computational time. If fish is sleepy he has a random probability to reactivate. After this initial check a fish operates only if he has positive cash and he will perform the three main functions **see**, **next**, and **action**.
The **see** functions allow the fish to observe the environment, in this case informations on asset price and agents who are linked to him.
In the **next** function an agent modifies its beliefs, and in this case fish is able to understand if he should go in survive mode. This will make him believe that he should not open positions in this situation later in the action function. Furthermore fish can modify its friends list according to his internal parameter. In that case he will remove friends with trust less than a threshold, and add friends from a small random sample of the entire network. The last operation on fish beliefs is to *"check socials"* where fish can look at positions opened by his friends in that timestep and gathering all positions in a pool.
In the **action** function the fish will decide to buy positions or sell them if survive mode is not activated, and will only sell positions in the contrary case in order to regain some liquidity. When fish decides to buy positions then he can choose to be influenced and opening the same position ("long" or "short"). If fish doesn't want to be influenced then he performs *random trading*.

B. Shark Agent

Sharks are agents which belief is that following other agents with the biggest influence on the network will bring them more profits. So they never trade randomly but always following the trends of the markets and form beliefs on market condition. Before they act, they **compare** their belief of market condition, i.e. bullish or bearish, and check if it's consistent with the most bought position (either "long" or "short") of most **influential contacts**. Sharks are *vigilant* and never sleep like fishes, because when market is neutral they need to never lose a catch and try to anticipate market trends by looking at most bought positions which can lead to a breakout of that lateral market.
So, briefly, main parameters are:

- popularity threshold, if a contact has less links than this value then the shark will eventually remove him from the contacts list
- points to change belief, it's basically a counter which manages the shark's belief on market condition;
- probability to update contacts list.

We state here that generally it is fishes who choose to follow a shark, and that a shark may choose to follow a very popular fish without knowing his type. So basically sharks follow popular agents and periodically will update their contacts-list based on their popularity.

Example of decision making process of a Shark. The

shark will always perform all three main functions. In the **see** function a shark will check the asset's price and his contacts-list. In the **next** function shark contact will randomly decide whether to update their contacts-list based on a probability. If yes then agent will remove contacts who have a number of links lower than a internal threshold, called *popularity threshold*. Then shark looks for potential new contacts from a small sample of the traders network, then chooses the one with the highest number of links, i.e. influent on the network. The initial trust the shark assign is generally high because of the new contact's popularity.
At the end of the next function the shark finds out what is the dominant opinion on market's next move and will keep mind of it, while checking his belief on present market condition[1].
In the **action** function if the shark realizes that his opinion is concordant with that of other most influential traders, then he opens a position in the relative direction, "long" or "short", and then updates his wealth.
When sharks observe that his opinion is discordant with that of his contacts he will start selling positions relative to the opinion of his contacts. In the end if the shark has no belief on current market condition then he will start selling positions randomly in order to regain liquidity.

C. Whale Agent

Whales are the **richest** agents of the network and have **private information** on whether they are buying or selling which they can choose to make it public or not. The objective is to simulate a real world fact where very influential and rich people on the world can declare on social medias that they "bought 1000 bitcoins because of some reason", or that "China banned cryptocurrencies for security reasons". We believe that in some of these cases whales are already profiting when announcing these kind of news because of past and not announced transitions they made. So whales are meant to be silent during their **accumulation** period, when market is reigned by fear and they're opening positions in the same direction in a semi-periodic way. On the other side whales are meant to be loud during their **distribution** period, that is when whales are already profiting and trying to sell their cryptocurrencies to other agents, while catching as many fishes as possible from social media.
Internal DOFs of the whale agent are:

- WPS, whale profit streak.
- N_orders_thr, how many orders in the same direction a whale must have in order to change its social mode from "silent" to "loud".

Example of decisional process. Whales are initialized with an active sleep mode which can be deactivated with a probability. The whale will always perform the see and next function, while the action function only when not in sleep mode.
In the **see** function the whale checks the actual price, his links and his profits when in social mode [2]. If whale is profiting than a waiting counter is updated until reaching the threshold N_orders_thr.
The **next** function is about when whales change their behaviours. Accumulation period starts only if wallet is empty. At that point also social mode is set to False, and the waiting counter to 0. A whale can also fall asleep in this moment. Whenever his wallet is not empty, whale checks if the waiting threshold WPS is reached while in social mode. In this scenario the whale (in the **action** function) will start selling in the so called *distribution mode*. Here whale will also set his social mode to False and will start removing links from neighbors.
Therefore a whale is characterized by three phases. Silent accumulation mode, whale invest a bigger percentage of his wealth and have very few links. Loud accumulation mode is the second, where the whale starts increasing the number of links with other

[1] belief is updated with a system of points

[2] Profits are checked by looking at mean portfolio value. If price at the current timestep is higher than the mean of all positions in the wallet, then it means whale is profiting. That is because for simplicity whales only open "long" positions

agents and invest less than when he was silent. Distribution mode is the final stage, where the whale will slowly empty his wallet.

III. MODEL

The core components of our model consist of the **network manager**, responsible for updating, creating and removing connections between agents, and the **price manager**, responsible for adjusting the price in a proportional manner based on the current agents' positions.

A. Price Update

Taking in consideration the sum of long and short orders (*slo* and *sso*) and knowing that $\alpha = \mu P$, then the price updates as follows:

$$P \leftarrow P + \Delta, \quad \Delta = \alpha\,(slo - sso) \tag{1}$$

Also there is a possibility for the price to become negative. In order to avoid this situation the market imposes a limit on short orders. Considering $\delta = slo - sso$ we have:

$$P' = P\,[1 + \mu\delta] \tag{2}$$

which leads to the condition of $\delta > -\frac{1}{\mu}$ in order to avoid *negative prices*. Since in the worst case $slo = 0$ is possible, then we need $sso < \frac{1}{\mu}$. So, knowing the number of agents, we can limit each agent to have an upperbound limit on how many orders to short. This limit is simply:

$$limit = \frac{sso\_max}{N\_agents} \tag{3}$$

B. Trust Matrix

Regarding the process of trust matrix updating, we encountered the challenge of determining what makes trust among agents grow or decline. We reached the conclusion that when **profits** of linked agents are **synchronized**, then trust should grow, which reflects having taken the same action.

Therefore, if two agents are both experiencing profits during the same timestep, it indicates a higher level of **mutual trust**. The same principle applies when both agents are incurring losses because trust is also a measure of belief's alignment. If they *believe* that their beliefs align, it should be reasonable for agents to accept financial losses. When profits are desynchronized, it indicates that agents have somehow taken different actions, leading them to infer that the other agent has differing beliefs.

The model calculates the wealth of agents at the current timestep and compares it with the previous step. If there is a link between two agents, the model verifies whether the profit's **sign** is positive or negative and **updates** the **weights** accordingly.

IV. EXPLORING DOFS

In the current section, we examine the behavior of our model across different degrees of freedom in the system. The grid parameters used for the model's batch run are presented in Table I. Due to time constraints, our research did not cover a wide range of values. However, we will provide explanations for our choice of certain values. The following decisions were made:

- N_agents: We opted to keep the number of agents at 200. Testing with 500 and 1000 agents revealed that the time complexity exploded. For instance, when simulating 2000 steps, using 1000 agents took approximately 12 hours, while using 200 agents only required 1 hour.
- μ: we tested 0.001, 0.01, 0.05 and we observed that oscillations in price get larger.
- N_init_edges: this parameter was modified during its initial implementation, but no significant changes in behavior were observed when varying this initialization parameter.
- Agent type ratios: In our model, fishes constitute the majority of agents in the society, while

Table I. Parameters batch-run

Type	**Parameters**	
model	N_agents	200
model	μ	0.01
model	N_init_edges	5
model	p_sharks	0.20
model	p_whales	0.01
whale	WPS	10, 25
whale	N_orders_thr	10, 20
shark	mean_popularity_thr	10
shark	points_change_belief	5, 15
fish	mean_trust_thr	0.50
fish	prob_tobe_influenced	0.80

sharks represent a fraction (around 20%) of the overall population, and whales make up a small percentage (generally 1%).

- Fish parameters: we decided to keep values for the trust threshold for removing friends and the probability to be influenced constant. We already know that if fishes wouldn't be influenced, then their behaviour would be that of a random trader, thus we keep this probability high in order to not fall on gaussian distributions of returns.

So, we have 8 different combination of parameters to explore and each model is signed with a label based on which parameter is taken in consideration. For example, WPS = 10, N_orders_thr = 20, points_change_belief = 5 would be model_121.

In our simulations we can already see the **herding effect** emerging not only in linear scale for the price, but also on the **logarithmic scale**, which was a limitation of our previous work.

Furthermore, an interesting phenomenon of this network model is the emergence of **Wickoff**-like **patterns**, characterized by *accumulation* and *distribution* regions of price as showed in Figure 5. These patterns are the evidence of a simulated market made of distinct phases where buying and selling pressure can prevail, leading to *shifts* in **market sentiment**.

A. Shark's DOFs

In the shark's batchrun we take a look at how the **flexibility** of shark's **beliefs** influence the price dynamics. The findings, displayed in Figure 2, show how points_change_belief is a relevant parameter since it can undermine the whales ability to dump the market. Since whales are still pretty influent this leads to the formation of **bubble-like trends**. Finally, when sharks lack confidence, whales assert their dominance by triggering **frequent** price **crises** in the market.

B. Whale's DOFs

During the batchrun involving whales, we observed that by varying the parameter N_orders_thr had no significant impact on price dynamics, so we won't make further considerations. Let's focus instead on WPS. In Figure 3, we observe that **larger** values of **WPS** contribute to market stability, resulting in **fewer crashes**. In particular, we discovered that in models with higher WPS, like model_212, whales were able to generate profits at the expense of sharks, whereas in the model_112 (lower WPS) sharks where able to make profits together with whales. An example illustrating this is showed in Figure 4 of model_221, where whales with higher WPS manage to achieve consistent profits while sharks do not.

V. STATISTICAL ANALYSIS

We investigated some stylized facts of **high volatility markets**, including the *fat-tail phenomenon*, *unit-root property*, and *volatility clustering*, taking inspiration from [1].

We made a preliminary examination on log returns of our **network model**, comparing it with a previous one which lacks a network structure, then we **compare** it with a real cryptocurrency market, with data collected on **17** different **cryptocur-**

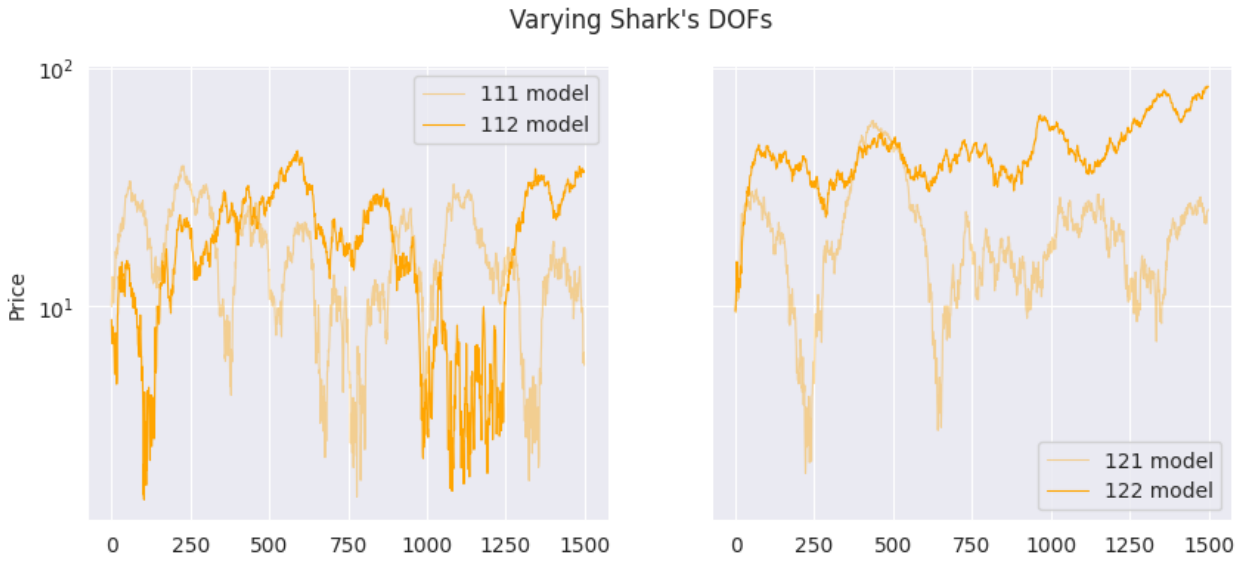

Figure 2. The models **1 refer to shark's having a higher flexibility on changing their belief on market condition. This means that a lower amount of points (5 points) is required for changing beliefs on market condition. The models **2 refer to sharks being more rigid on their belief (15 points).

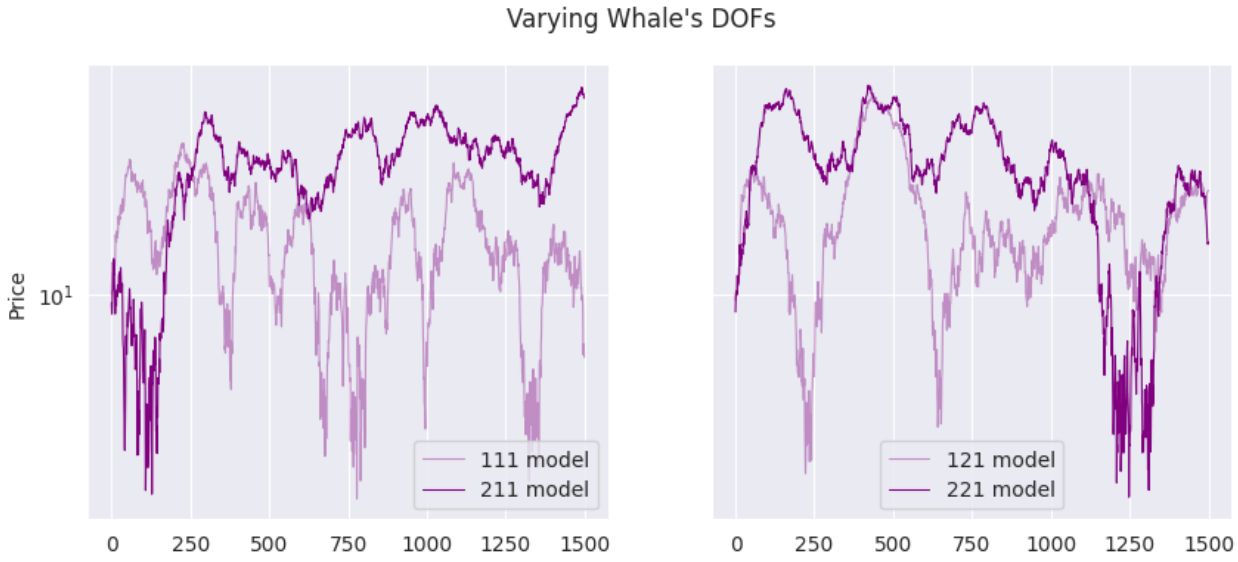

Figure 3. In this plot we examine the impact of changing WPS's DOF from 1*1 to 2*1, keeping the last DOF fixed to 1 because we showed that it's dominant. In this situation one can observe how fewer crashes occur.

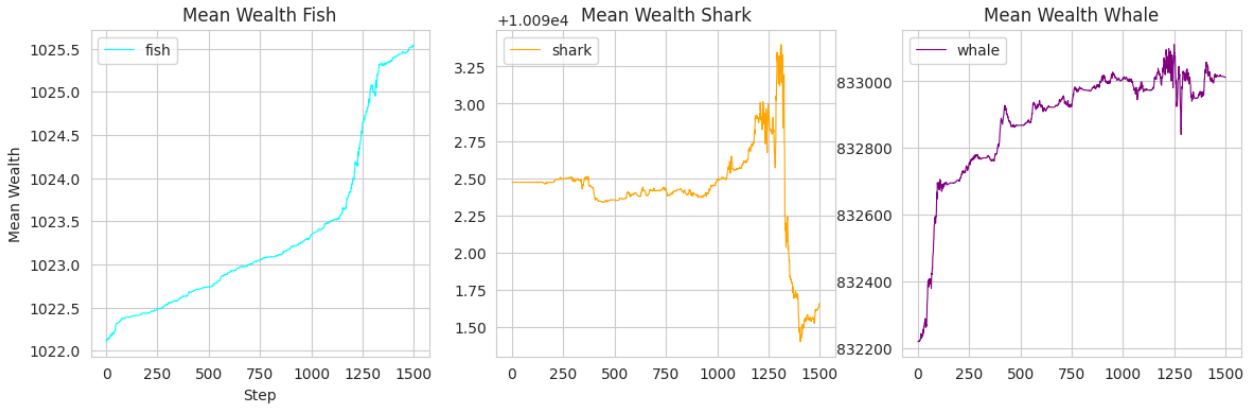

Figure 4. Mean wealths for model_221, whales are able to make profits at the expenses of sharks. This happens in models with any combination 2**, as shown also in the appendix in Figure 9.

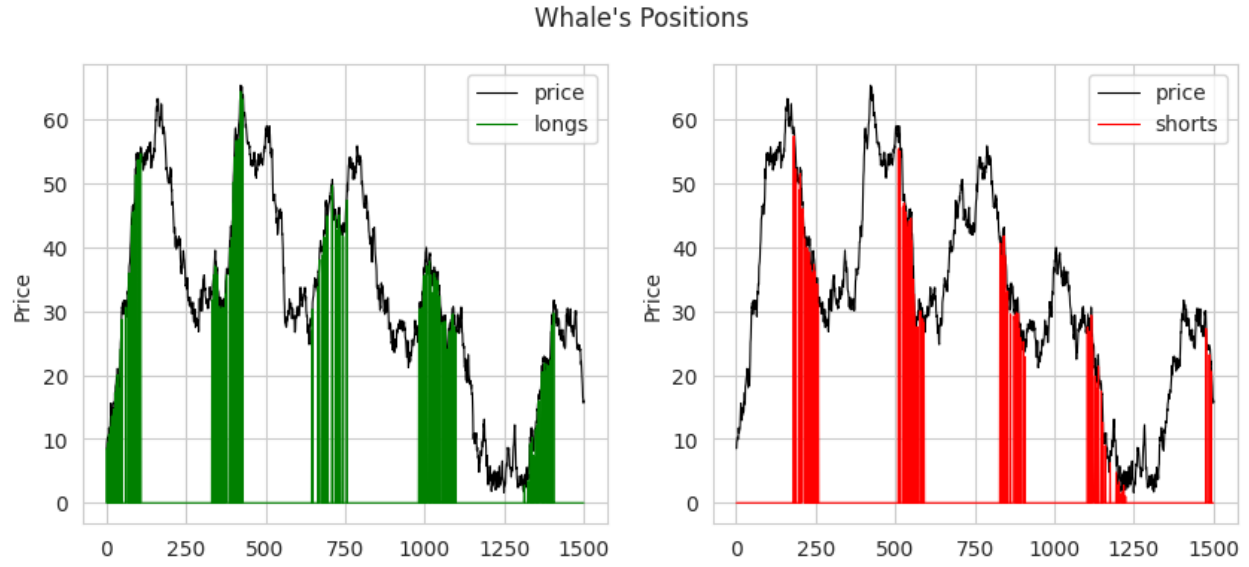

Figure 5. Flags for whale positions in model_221. Green flags refer to long positions taken by a whale, which cause buying pressure, and red flags refer to short position taken, which cause selling pressure. Here one can observe the emergence of **Wyckoff dynamic** as whales are anticipating rises and falls of the market price.

rency markets from *Yahoo! Finance*[3]. From this comparison shown in Figure 6 and Figure 7 it is observed how the model without network structure presents a normal distribution with no kurtosis and how the network model is a better candidate for simulating a real market.

[3] `https://it.finance.yahoo.com` for the period spanning from 2018-01-01 to 2022-01-01

Table II. Fat Tail Phenomenon

	models	**markets**
Mean Kurtosis	12.41 ± 8.57	10.01 ± 4.00
Significance level	Critical value	Z Test result
0.05	1.96	0.25

For the **statistical analysis** we considered **24 runs** of our network models, some taken by the batchrun, and others with same parameters but changing random seed, and calculated the returns, kurtosis, and autocorrelation functions in order to perform the three tests of our interest for the validation of key aspects of our model.

A. Fat Tail Phenomenon

In this section, we analyze the return distribution of our network model across 24 runs and calculate its **kurtosis**. Our findings consistently show a positive kurtosis, and we present the mean value with its error in Table II. Then we calculated the kurtosis for 17 real markets and determine their mean value with its error. We observe that real markets exhibit a range of kurtosis, varying from 5 to 15, while in our model we observe *larger variations*, ranging from 2 to 25. Consequently, the mean value incurs a higher error. Finally, we perform a **Z Test** to compare the two means and find that we cannot reject the null hypothesis, indicating that our network model generates returns with a **kurtosis similar** to that of a **real market**.

B. Augmented Dickey-Fuller test

Augmented Dickey-Fuller test is performed to check whether a **time series** is **stationary**, i.e. if it has no trend, exhibits constant variance over time and has a constant autocorrelation structure over time. The null hypothesis of this test is that the time series is non-stationary, which means it has some **time-dependent structures** and doesn't have constant variance over time. If the p-value of the test is greater than the significance level (in our case $\alpha = 0.05$), then we can't reject the null hypothesis.

It is known that real markets are non-stationary, so we verified it for cryptocurrency markets with success. Then we proceed to test the simulated markets, obtaining success in 23 out of 24 runs, indicating that our network model **well simulates** the **non-stationarity** of real market's price time series.

C. Volatility Clustering

Volatility clustering is a phenomenon observed in financial markets where large returns in prices are likely to be followed by similarly large returns, regardless of their direction, while small returns tend to be followed by small ones. This **pattern** can be quantitatively **measured** through the **autocorrelation function** (ACF), which measures the correlation between a price timeseries and its **lagged values**. We focus on the ACF of absolute returns to measure volatility clustering, in particular because it only considers the magnitude of returns, not their signs.

The objectives of this section are two:

- show that the network model has volatiliy clustering phenomenon, unlike the previous abm model with no network structure;
- see if the volatility clustering effect is *similar* to that of a real market.

For this purpose we calculated the ACFs (for our 24 runs) and collected their mean values to obtain a new mean (with error calculated as standard error of the mean). Then we did the same for our 18 cryptocurrency markets and the results are shown in Table III. To explain these results we bring the attention to the Figure 8 were we show how our network model is **successful** in having a **volatility clustering** phenonmenon, unlike the previous abm model, because the ACF of absolute returns show a slowly decaying movement. **But**, this clustering effect is not comparable with that of a real market, in fact it's **more intense**, as one can also get an intuition from Figure 6.

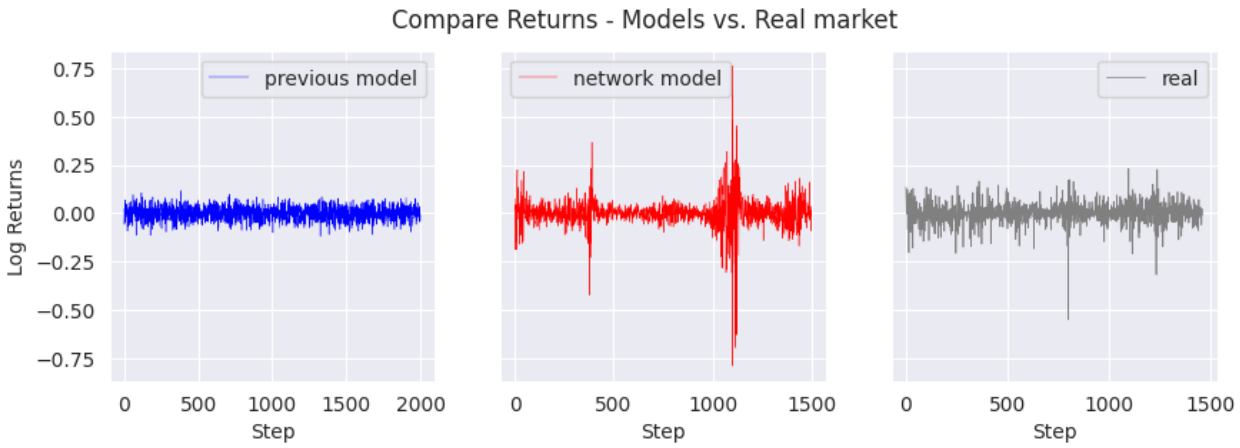

Figure 6. A comparison example of log returns of the previous ABM model without network structure, the new model (model_212) with a network structure and a BTC-USD from 2018-01-01 to 2022-01-01. One can observe how the network model is a better candidate for simulating a real market.

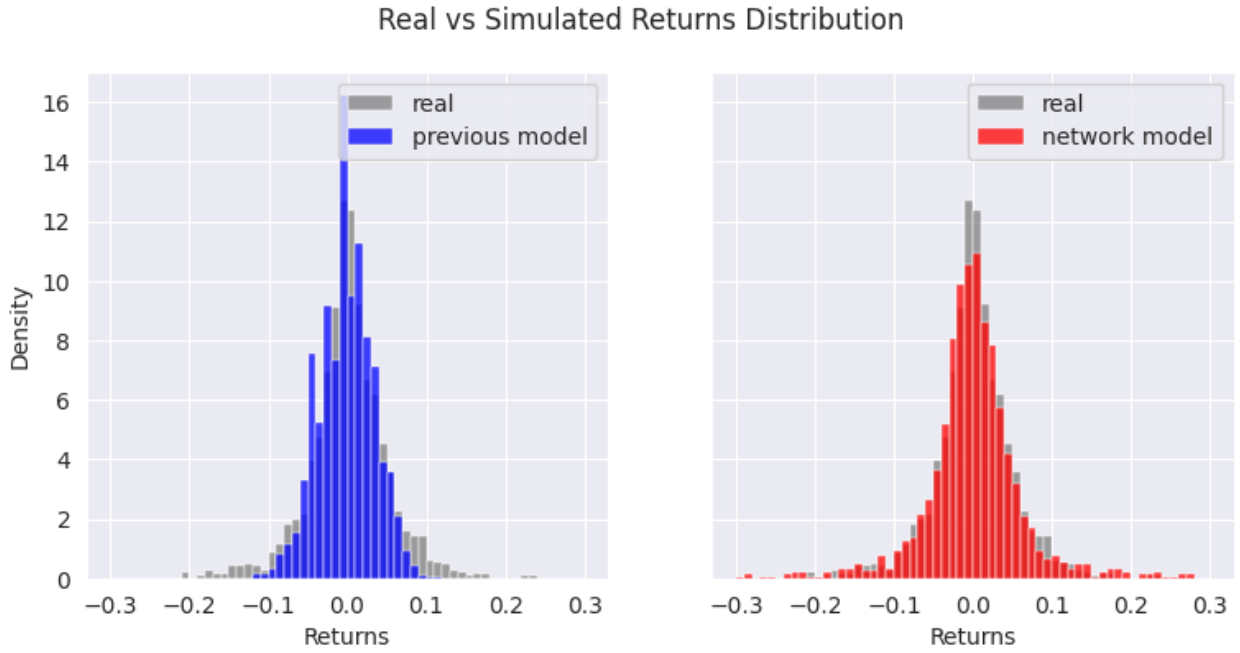

Figure 7. The same comparison of Figure 6 where it is shown how the distribution of the new model is more outlier prone with less noise, which in the previous model was given by random traders.

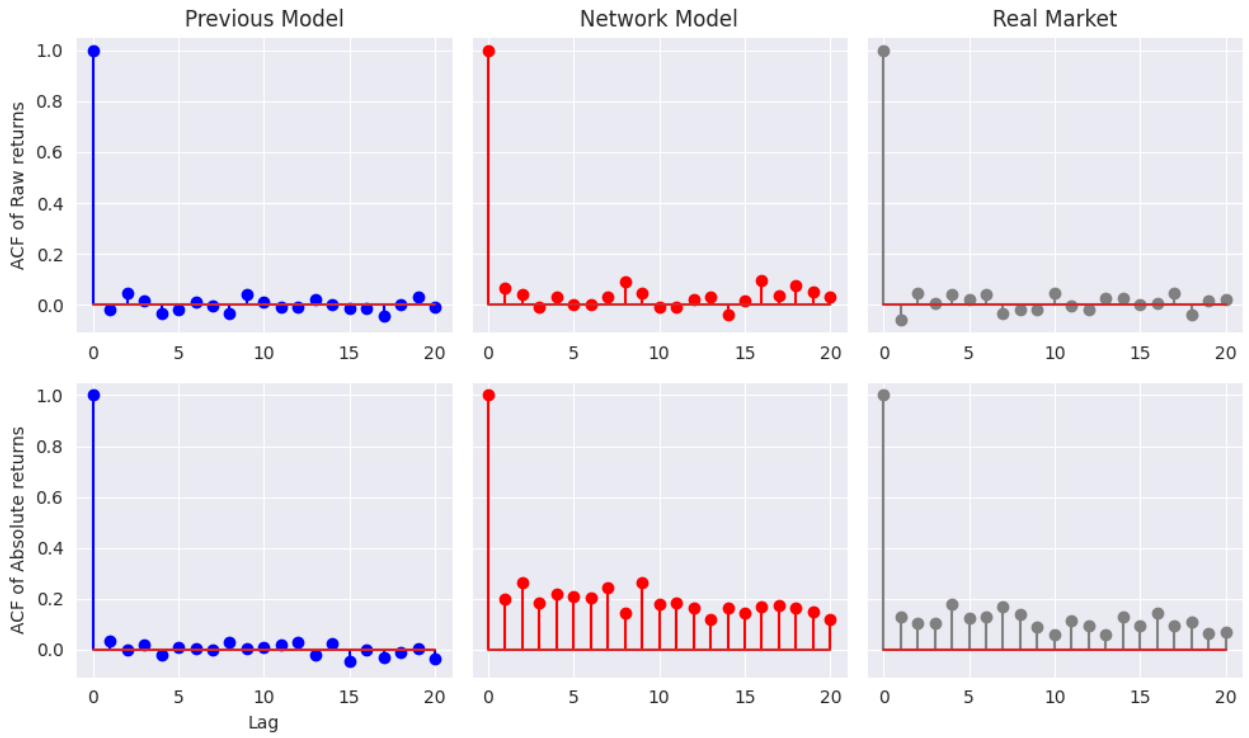

Figure 8. Comparison of the autocorrelation functions of absolute returns among the old model, the network model and an example of a real market. As one can see the network model has a similar decaying behaviour, unlike the previous model (without network structure).

Table III. Volatility Clustering with ACF

	models	**markets**
Mean Raw	-0.002 ± 0.003	0.005 ± 0.001
Mean Abs	0.293 ± 0.020	0.102 ± 0.006

VI. CONCLUSIONS

The objective of our study was to **replicate** the behavior of **highly volatile markets**, and we now summarize the results. We successfully developed a model capable of generating returns with a **similar kurtosis** to real markets. These simulated markets exhibit **non-stationary** price time series, characterized by decaying autocorrelation functions that indicate the presence of **volatility clustering**, and present also the **emergence** of *herding-effect* and *Wyckoff dynamics*.

However, our model presents some problems:

- **Kurtosis wide variability**: in certain simulations, the kurtosis values were either too low or too high. Further research should focus on finding methods to balance this aspect and achieve more controllable kurtosis values.
- **Intense volatility clustering**: the observed volatility clustering effect in our model was more pronounced compared to real markets, resulting in a significantly higher values for the autocorrelation function of absolute returns. This is a clear evidence that our model has room for improvement.

In terms of agent and model design, we identified additional limitations:

- **Time complexity** and **agent parallelization**: the network model's time complexity is substantial, and the agents are not parallelized. Consequently, our study was limited to only 200 agents. To address this, we propose a structural modification that involves transforming agents into independent process threads. This enhancement would enable network models with thousands of agents, providing more insightful outcomes.
- **Indirect communication**: currently, communication within the model relies on a trust matrix. We have contemplated an additional approach where influential agents (sharks) disseminate broadcast messages throughout the network to express their beliefs on market condition. Implementing this type of communication in the model would present valuable opportunities for further exploration.

Finally, we suggest conducting additional technical investigations into the analysis of simulated Wyckoff markets. This exploration would aim to identify the **psychological factors** that can contribute to a predictive model for outliers and **market crashes**, thereby enhancing our understanding of real-world applications in cryptocurrency markets.

[1] L. Cocco, R. Tonelli, and M. Marchesi, An agent-based artificial market model for studying the bitcoin trading, IEEE Access **7**, 42908 (2019).
[2] R. Palmer, W. Brian Arthur, J. H. Holland, B. LeBaron, and P. Tayler, Artificial economic life: a simple model of a stockmarket, Physica D: Nonlinear Phenomena **75**, 264 (1994).
[3] U. Forte, Econophysics: The emergence of complexity patterns in a stock market using agent based models, (2017).
[4] J. Kazil, D. Masad, and A. Crooks, Utilizing python for agent-based modeling: The mesa framework, in *Social, Cultural, and Behavioral Modeling*, edited by R. Thomson, H. Bisgin, C. Dancy, A. Hyder, and M. Hussain (Springer International Publishing, Cham, 2020) pp. 308–317.
[5] P. Tom and E. Brad, Multi-level mesa.

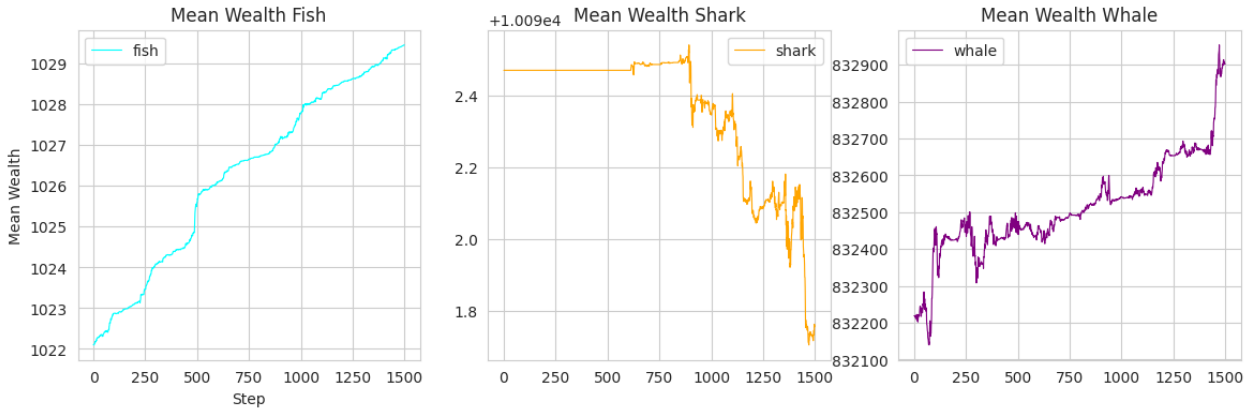

Figure 9. Mean wealths for model_212. As said for model_221, we can see how WPS (models 2** or 1**) is a dominant DOF in determining the whale's and shark's mean wealths.

Appendix A: Additional Plots