Using Degree Centrality to Identify Market Manipulation on Bitcoin*

Daiane M. Pereira[1][0000−0001−8681−9516] and Rodrigo S. Couto[1][0000−0002−6921−7756]

Universidade Federal do Rio de Janeiro - PEE/COPPE/GTA - Rio de Janeiro, RJ, Brazil {daiane,rodrigo}@gta.ufrj.br
http://www.gta.ufrj.br

Abstract. In 2014, the Mt.Gox Bitcoin exchange had its internal dataset hacked and leaked. After that, some studies employ this dataset to evaluate if Mt.Gox was doing market manipulation on Bitcoin. Also, they identify patterns of this manipulation. Based on these studies, this paper analyzes the Bitcoin blockchain in the period where Mt.Gox was active. We model the transactions in the blockchain as a graph and evaluate the degree centrality of each node. We thus analyze how the ranking of nodes with the highest centrality values changes over time. Our conclusions indicate that top nodes are stable, but there is a period where it changes. To better understand this behavior, we simulate the insertion of transactions in the network and verify how the ranking changes. As a result, we provide indications that we can use ranking changes to detect malicious activities. We also show a case study using this ranking to predict abnormal behavior in the network.

Keywords: Bitcoin · Mt.Gox · Complex networks.

1 Introduction

A Bitcoin exchange is a place where people can store, buy, or sell Bitcoins using fiat currencies or other cryptocurrencies (i.e., altcoins). Despite their importance, the interaction with these companies can represent a risk factor for Bitcoin holders. It is true since storing fiat or cryptocurrencies in accounts at these companies can be seen as a type of centralization, that is, a single point of failure. Also, the lack of Bitcoin regulation makes it difficult to recover money in fraud cases on an exchange platform. Between 2010 and 2013, 45% of the exchange were closed, and 45% of these platforms did not compensate their customers for their losses [19]. Another problem raised by some companies is the

* This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. It was also supported by CNPq, FAPERJ Grants E-26/203.211/2017 and E-26/211.144/2019, and FAPESP Grant 15/24494-8. The final publication is available at Springer via https://doi.org/10.1007/978-3-030-93944-1_14.

suspicion that they perform market manipulation and their activity artificially increases the USD-BTC price [5].

In 2014, the major Bitcoin exchange, called Mt.Gox, was filed for bankruptcy and had its trade history leaked. This leakage allows researchers to analyze more closely the behavior of this exchange since not all trading in an exchange appears in the blockchain [2]. It happens because some users store their bitcoins in the exchange wallet. As the exchange controls these wallets, their transactions are not added to the blockchain. In this case, the exchange keeps the Bitcoin in its address and uses internal data to controls the users' amount. The users do not have Bitcoins, as they do not have a pair of keys to prove their ownership. The same occurs when two users of the same exchange transfer values between them. In these two cases, these transactions are called off-chain transactions. A different scenario occurs when a user buys Bitcoin in an exchange and transfer the Bitcoins to his wallet (i.e., any wallet that this user has the pair of keys). In this case, a transaction must be added to the blockchain. It is called on-chain transactions.

The leaked data from Mt. Gox is analyzed in papers that try to find evidence of market manipulation. They find transaction patterns in the leaked data that support the hypothesis that Mt.Gox was making market manipulation [2, 5]. Based on the analysis of [2, 5], this work performs a graph analysis of the blockchain, focusing on the same period used in these papers. We hypothesize that the market manipulation made by Mt.Gox changes the expected behavior of blockchain nodes, generating instability in it. To investigate this hypothesis, we are interested in knowing how the top-ranked users change over time and examining if these changes can be related to the Mt.Gox malicious activity and the increase in the Bitcoin price.

Our primary contribution is analyzing how the top-ranked nodes change in the blockchain to identify malicious activity in the network. We show how these changes occurred during the Mt.Gox activity period and how the network's behavior was different from the others. Also, we provide a forecast model to predict this unstable behavior.

This work is organized as follows. Section 2 describes related work. Section 3 presents information about data acquisition and graph construction, including also a preliminary analysis. Next, Section 4 presents the stability analysis and discusses our main findings. Section 5 presents the case study, while Section 6 concludes the paper.

2 Related Work

Different papers analyze aspects of the Bitcoin network. One important aspect is its decentralization, addressed in [3,8]. These works show that few entities control most of the computing power used for Bitcoin mining.

Anonymity is another aspect addressed in Bitcoin literature. The studies about this aspect show that Bitcoin is not entirely anonymous. Fleder *et al.* were able to link Bitcoin addresses to real user identities [4], using data from

Internet forums. Other approaches also break user anonymity, such as linking Bitcoin addresses to IP addresses [11]. The work in [12] provides a survey covering aspects of Bitcoin anonymity [12].

Extracting complex network characteristics from the Bitcoin environment is also an interesting research topic. For example, the studies conducted in [14] and [16] identifies that the degree distribution of Bitcoin nodes follows a power law. Another fact about the Bitcoin network is that it has characteristics of the small-world phenomenon but with a high diameter [15, 17]. The authors also studied the outliers in the in-degree and out-degree distributions. Maesa *et al.* consider as outliers the nodes with the in-degree or out-degree with at least one order of magnitude higher than the average degree. Both behaviors are associated with the presence of artificial transactions in the network [16, 18]. One example of an artificial transaction is a transference where one node transfers the exact amount 0.00001 BTC to 101 other nodes.

Another topic of interest in the Bitcoin network is the analysis of frauds and market manipulation. In [5], the Mt. Gox leaked data is used to verify if this exchange made market manipulation. The authors find out suspicious trade activity in accounts listed in the leaked data. These accounts were called Willy and Markus bots. The article shows evidence that the increase in the Bitcoin price is related to these bots' activity. In 2017, during a trial in Japan, the Mt. GOX CEO confirmed that the company was responsible for the Willy activity [24].

Chen *et al.* also analyze market manipulation by Mt.Gox [2]. They perform this investigation in the Bitcoin network through graph analysis. The authors use the Mt.Gox leaked data to make a graph analysis to find malicious trade evidence. They classify the accounts of the dataset into three categories (i.e., extremely high, extremely low, and normal) based on the amount that each account paid for the Bitcoins and the daily price reference. They consider that the extremely high account and the extremely low account are abnormal accounts and verify that they have different characteristics when compared with the normal ones. After that, they analyze the dataset looking for patterns in the transactions made by the abnormal accounts. They find out the existence of many self-loops, unidirectional transactions, triangles, and other structures and conclude that these patterns are strong evidence of price manipulation. Most of these abnormal transactions occurred during the last 12 months of Mt.Gox operations, with a more relevant presence in the last six months of 2013.

Although the investigations of [5] and [2] give essential contributions to the studies about market manipulation in the exchanges, they focus the analysis on the Mt. Gox leaked data. As details about internal transactions in other exchanges are not available (i.e., transactions that occur in the exchange and do not appear in the blockchain), their analysis is not extensible to other periods where the Bitcoin price had a high increase. Our work contributes to the literature showing that the abnormal transaction patterns identified in [2] change blockchain's graph behavior. This analysis can be a base to design mechanisms to identify malicious activities, regardless of leaked data.

3 Base Methodology and Preliminary Analysis

In this section, we present the methodology applied in our analysis, together with preliminary results. We start with the details of our dataset. Then we show the graph definitions used in this work. We also conduct the first part of our graph analysis to analyze if there is a relationship between the top user ranking and the Bitcoin price.

3.1 Data Acquisition

To receive an amount of money in the Bitcoin network, a user needs to create a wallet with at least one pair of public/private keys. The public key is used to create the Bitcoin address, used as an input or an output in the transaction. The private key proves the ownership of this address. If a user has Bitcoins in one address and wants to send these Bitcoins to other users, he/she has to prove the ownership of this address. It means that only the user who has the private key can transfer the Bitcoins of the corresponding Bitcoin address.

When a transaction occurs, it is broadcasted to the other nodes in the network. The miners get this transaction, groups with others, and create a block. The transaction is only complete when included in a block in the blockchain. The blockchain contains blocks with all the transactions that have ever happened in the network since its creation. The entire Bitcoin transaction history (i.e., blockchain transactions) is available for download using a Bitcoin client, such as Bitcoin Core [21].

We use the ELTE Bitcoin project dataset, available in[1]. This dataset has pre-processed blockchain data, organized into spreadsheet format. Using this data, instead of downloading and processing more than 270GB of data (i.e., the corresponding size of the Bitcoin blockchain), we have to deal with less than 40GB (the size of the ELTE dataset) of data. Also, we choose the dataset to avoid dealing with the script code format of the blockchain.

An additional pre-processing is necessary to remove inconsistent data from the dataset, such as the mining rewards, represented in the dataset as transactions without input addresses. We also aggregate the Bitcoin addresses that belong to the same entity. However, as we are interested in transactions between the same exchange, we choose to use the well-known aggregate heuristic rule noted in [20]. Using this heuristic, if a transaction has multiple input addresses, we consider all these addresses belonging to the same entity. This assumption is reasonable since the entity that creates the transaction must have all input addresses' private keys to prove its ownership.

The final dataset used in this work has all the transactions up to 02-2018. However, we select the interval between 06-2012 up to 05-2015 to analyze. This period is selected because we are interested in the blockchain's behavior during the reported malicious activities of Mt.Gox, which occurs in the year 2013. We select an interval before and after this year to compare the behavior. For each month in this interval, we create a graph and perform our analysis.

[1] https://senseable2015-6.mit.edu/bitcoin/

3.2 Graph Definition

We model our data as an undirected weighted graph, $G_m(V_m, E_m, w)$ where V_m is the set of nodes representing users that made transactions during the month m. E_m is the set of edges representing the transactions between users that occur in m, and w is a weight associated with each edge. If two users have at least one transaction between them during the month m, there is an edge connecting these nodes. This edge's weight is the total number of transactions made between them in m.

We make our analysis using degree centrality, which measures a node's importance based on its number of edges. Equation 1 shows how to evaluate the degree centrality. In this equation, $N-1$ is the maximum number of edges that a node can have in month m, and e_{ij} is one if there is a link between node i and node j in month m. If a node has $N-1$ edges, it has one edge with all other nodes.

$$C_D = \frac{\sum_{i=1}^{N} e_{ij}}{(N-1)}. \tag{1}$$

3.3 Rank Analysis

We start our analysis by creating one graph G_m for each month between 06-2012 ($m = 1$) and 05-2015 ($m = 36$). Figure 1 shows the number of nodes and edges of each G_m. As explained in Section 3.1, we use an aggregate rule to group Bitcoin addresses. It means that the total number of nodes for each G_m is less than the total number of Bitcoin addresses used as input and output each month.

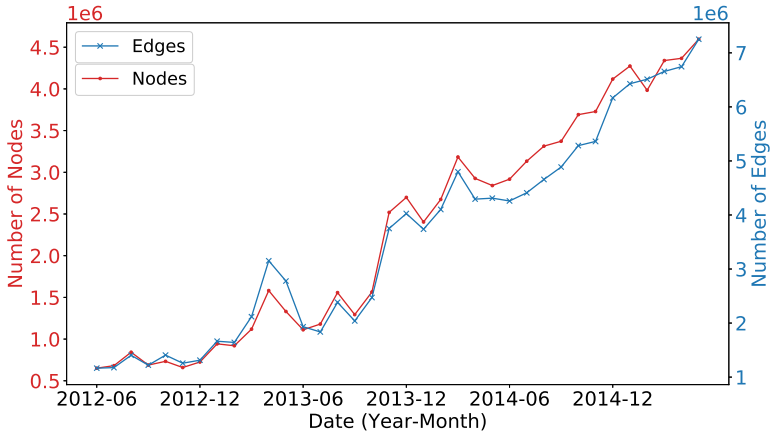

Fig. 1. Total Number of Nodes and Edges in each G_m

After the graph creation, we evaluate the degree centrality of Equation 1 for all nodes V_m, for each G_m. We then list the k nodes with the highest degree centrality in each month m. We use this top-k nodes to build the set $\mathcal{N}_m^k$. This set has the top-k nodes. We evaluate different values for k, and we use the number of transactions these nodes have made to uphold our decision. The top-10 nodes are responsible for almost 20% of transactions. For the top-100, we get precisely 33%. For the top-150, we have 34%. As we increase k to top-1000, we have approximately 40%, and for the top-2000, we have 44%. We choose to use k = 100 to simplify our analysis. Also, because these few nodes are responsible for an expressive number of transactions.

After selecting the top-k nodes, we analyze how this ranking changes over time. To this end, we compare the nodes of each set $\mathcal{N}_m^k$ with the set $\mathcal{N}_{m-1}^k$ of the month before. In this analysis, we are interested in finding out the number of repeated nodes. It means that we are interested in the cardinality of the intersection between $\mathcal{N}_m^k$ and $\mathcal{N}_{m-1}^k$. We call this number User Repetition, and Equation 2 formalizes it.

$$UserRepetition = |\mathcal{N}_m^k \cap \mathcal{N}_{m-1}^k|. \tag{2}$$

Figure 2 shows how User Repetition changes over time[2]. This figure also includes the Bitcoin price difference during this period by comparing the bitcoin price in the month m with the month $m-1$. We use the bitcoin historical market price, available in [1]. We can note that, in the period between 2013-02 and 2013-12, User Repetition has its lowest values. It means that we have fewer nodes repeated in the top-100 ranking list. This behavior is different from the previous and the next months, where we have a higher User Repetition. These higher values in the other months show that the 100 top-ranked nodes change little, indicating stability in this ranking's behavior. An exception occurs between 2014-12 and 2015-05. Although, in this case, User Repetition remains low for a short period, it could indicate that external factors (e.g., political scandals, terrorist attacks, government decisions, etc.) can influence the metric. This analysis is not covered in this paper and will be studied in future work.

We also calculate the average User Repetition in each six-month interval of the considered period, shown in Table 1. We use the interval names of this table in the rest of this paper. The six intervals were called A0, A1, A2, B1, B2, and C1. The intervals A1 and A2 represent the period where the evidence points out that the exchange Mt.Gox made most of its suspicious transactions [2]. A1 and A2 are also intervals with the lowest User Repetition. In the interval A0, we already had Mt.GOX activity, but based on the leaked data analysis, the number of suspicious transactions was inferior to the number presented in A1 and A2. We can also see that the interval A2 (i.e., between 06-01-2013 and 11-30-2013) has the lowest average, as expected based on Figure 2. This paper focuses on understanding network behavior during interval A2 because of its low average

[2] To evaluate User Repetition in the first month (i.e.,$m = 1$), we use the set $\mathcal{N}_1^{100}$ and the set $\mathcal{N}_0^{100}$, where $m = 0$ indicates the month 2012-05.

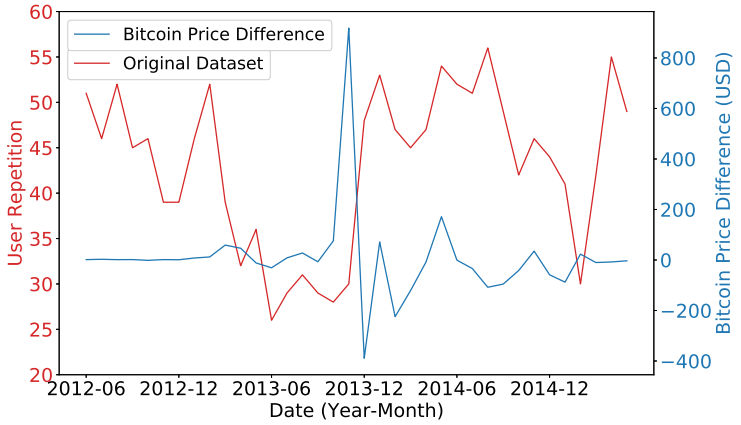

Fig. 2. Bitcoin Market price and User Repetition per Month

User Repetition, its high exchange market price, and the reported malicious activity [2, 5].

Table 1. Average Rank Difference

Date Range (From ∼To)	Interval Name	Average Rank Difference
06-01-2012 ∼11-30-2012	A0	46.5
12-01-2012 ∼05-31-2013	A1	40.6
06-01-2013 ∼11-30-2013	A2	28.8
12-01-2013 ∼05-31-2014	B1	49.0
06-01-2014 ∼11-30-2014	B2	49.3
12-01-2014 ∼05-31-2015	C1	43.5

The results of Figure 2 and Table 1 lead to a hypothesis that there is a relationship between the User Repetition metric and the market manipulation made by Mt.Gox, noted by the relationship between our metric and the Bitcoin price. In Section 4.2, we investigate if this hypothesis holds, analyzing the behavior of the User Repetition metric in G_m. In a nutshell, we analyze how a node enters the top-k ranking. It gives us information about how many transactions a node needs to perform and how it needs to behave to enter in this ranking.

4 Stability Analysis

As explained in Section 2, there is evidence of fraudulent behavior of Mt.Gox during the months before its filing for bankruptcy. The leaked data analysis shows evidence of artificial trades, mostly during its last months of operations, which correspond to our interval A2 and part of interval A1 of Table 1. Based on

our analysis of Section 3.3, where we observe low values in User Repetition, we want to verify how the insertion of new transactions affects our User Repetition metric. To this end, we perform a stability analysis. In this section, we first describe the employed methodology and then present its results.

4.1 Methodology

To perform the analysis, we select a six-month interval to insert the new transactions. For each month in this interval, we create the graph G_m and we use a uniform distribution to choose a subset S_m with $k = 100$ nodes from V_m. The nodes in S_m are responsible for the input of the new transactions. We choose each transaction's output from V_m or S_m, depending on the methodology that we are applying.

As Mt.Gox suspended all trading in 02-2014, we insert the new transaction in the interval B2 (06-01-2014 ∼11-30-2014) to make sure we are making our analysis without the influence of this exchange. Hence, in the rest of this section, the variable m refers to the months included in the interval B2, $m \in B2$. In other words, it means that our m values range from 25 to 30. The amount of new transactions inserted in each month m is based on a percentage of the total number of transactions that occurred in it.

The percentage of the total number of transactions that we choose goes from 2% up to 16% of the total number of transactions that occurred in the month, increasing 2% at each step. It means that, at each step, we generate a new graph $G_m^{'}(V_m^{'}, E_m^{'}, w^{'})$, where $V_m^{'}$, $E_m^{'}$, and $w^{'}$ are the set of nodes, edges and weights updated with the new transactions included. We calculate the degree centrality, select the top-100 nodes, and calculate User Repetition for each step, repeating the same procedure in all months $m \in B2$.

We employ three different methods to add new transactions. Our objective is to analyze how organized the transactions must be to change the ranking. In Method 1, we want to investigate our metric's behavior when we increase the network transactions, but with a high number of nodes responsible for making them. In Method 2, we also analyze our metric's behavior when we increase the network transactions, but this with fewer nodes being responsible for making them. In Method 3, we stably insert new transactions keeping the same nodes during all months.

Our goal with Method 1 is to analyze the impact of random transactions in User Repetition. It means that we want to verify User Repetition's behavior when random users made many new transactions. We assume that random transactions are not organized and controlled by an entity. It represents nodes that increase their number of transactions without any control of the network's major players.

In Method 1, for each percentage, we make the following procedures. First, we randomly choose the subset S_m (i.e., a subset with 100 nodes) used in the transaction generation. We remove these nodes from V_m. After that, we generate new transactions using uniform distribution to select an input node from S_m. We randomly choose the output node from V_m (i.e., a subset with all nodes

that make transactions during the month m) or S_m. At each transaction, we use one of the sets in the output. The selection from S_m increases the number of self-loops and triangles, as this group has fewer nodes than V_m. When we select from V_m, we increase the chance that this transaction will create a new edge. It happens because, as this set has a higher number of nodes, we have less chance of selecting a node that already has an edge with the input. Next, we create $G_m^{'}$ and evaluate C_D, using Equation 1. Then, we select the top-k nodes and evaluate User Repetition, using Equation 2.

Method 2 and Method 3 have many similarities with Method 1. Because of that, we explain next only the differences between these methods and Method 1. The difference is in how we choose S_m.

In Method 2, we select only one subset S_m, for each m, and this subset is responsible for creating all-new transactions in G_m. This process is different from Method 1, where we select one subset for each percentage. The behavior simulated in Method 2 is the insertion of a high number of transactions made by the same group of nodes. The consequence is that few users control the transaction. Thus, these users make a considerable amount of interactions (from 2% up to 16% of all transactions in the network).

In Method 3, instead of selecting S_m from each G_m, we create a graph $G(V, E, w)$, using the entire B2 dataset and select the S_m from this graph. It means that all months use the same subset S_m. In other words, an edge exists between two nodes if they have a transaction in any month of the B2 dataset. In the same way as Method 2, in Method 3, the subset S_m is responsible for the input of all new transactions for each m. If in a month m the node selected in S_m is not in G_m, we add this node to the graph. Our goal in this method is to analyze the stability of the original top-k changes, as we force nodes to be inserted and repeated each month. It gives us information about the behavior in the periods with high User Repetition (e.g., A0, A1, B1, and B2).

4.2 Results

We apply the three methods described in Section 4.1 to the employed dataset. The figures below show the average values of User Repetition and the confidence intervals, evaluated with a confidence level of 95%. It is difficult to visualize the confidence intervals in our curves because our results have a narrow confidence interval.

Figure 3 shows User Repetition after applying Method 1. The mark by a shaded area in the graph shows the B2 interval. As described before, this method changes S_m for each percentage. There is a difference in User Repetition between the original dataset curve and the curve for 2%. However, as we keep adding transactions, this difference does not have a considerable change, even when inserting 16% of new transactions. This happens because the subset S_m changes constantly, and we select $|S_m| = 100$ new nodes in the entire G_m. Even without any rules forbidding a node to be selected again, the probability that it happens is small. It implies that a node's chance to make more than 2% of new transactions is small.

Method 1 gives us information about the concentration of transactions in the network. With 2% of transactions created by random users, we decrease User Repetition by 15 positions. As User Repetition represents the number of repeated nodes in the top-100 rank, this decrease means that we insert 15 new nodes in the top-100 by just creating 2% of new transactions. Figure 3 shows this decrease as the maximum difference between the original dataset curve and the curve for 2%.

As we insert further transactions, the other curves (i.e., 4%,8%, and 16%) do not change significantly. It happens because of the randomness and frequency in selecting the subset S_m. We change the nodes in this subset frequently (i.e., at each percentage of new transactions). Hence, at each change, a different set of new nodes enter the top-k rank and keeps User Repetition with this decrease by almost 15 nodes.

Before the insertion of new transactions, our top-100 nodes were responsible for almost 30% of the total number of transactions each month. Based on that, the fact that we can change 15 positions in this rank with 2% of new transactions indicates a higher concentration of transactions in the first ranking positions, showing that the addition of random transactions can change the subset of top-100 nodes, but not the major part of these nodes.

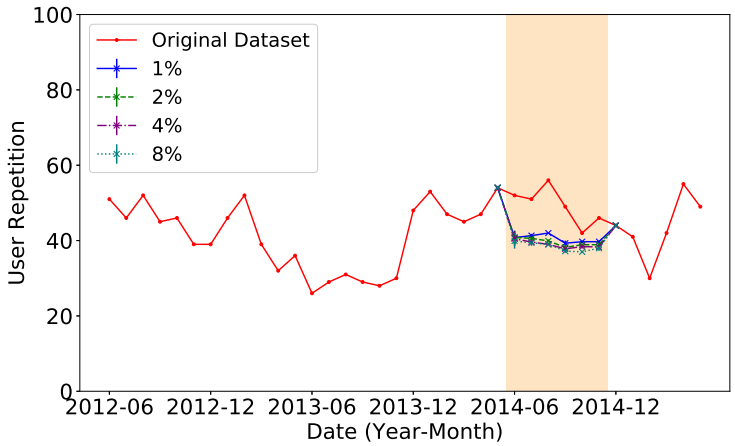

Fig. 3. User Repetition using Method 1

Figure 4 shows User Repetition after applying Method 2. In this method, we do not change the subset S_m for each percentage (i.e., as in Method 1). Instead, we change it only once each month. For example, in a given month, to evaluate User Repetition for 16% of new transactions, we use the same subset S_m employed for 8%, and so on. Figure 4 shows that, for 4% of new transactions, we include almost 35 new nodes in the top-k ranking (i.e., the maximum decrease in User Repetition). Furthermore, with 16% of new transactions, User Repetition

is low. It means that at each month in B2, we insert most of S_m nodes in the top-k list. As this list changes once per month, User Repetition keeps low values because fewer nodes are repeated. The behavior of User Repetition in Method 2 looks more similar to the one of the A2 interval (i.e., the period where Bitcoin prices have increased due to market manipulation). It reinforces our initial hypothesis that the decrease in User Repetition is related to the activity of Mt. Gox.

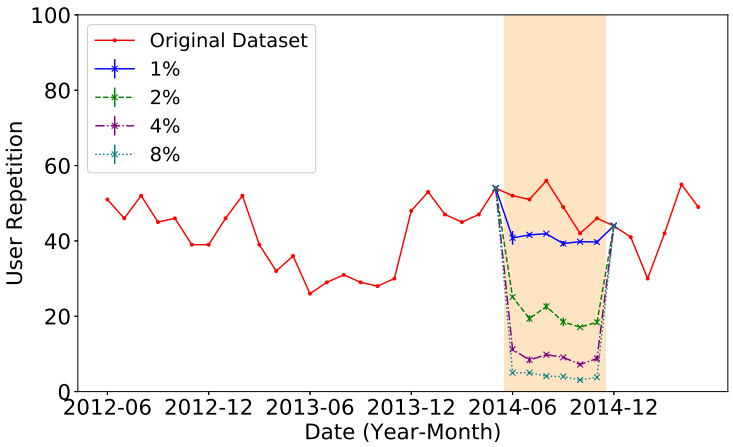

Fig. 4. User Repetition using Method 2

Figure 5 shows User Repetition after applying Method 3. In this method, we keep the same subset S_m of nodes during all months. We thus verify the behavior of User Repetition when these nodes insert new transactions. As expected, when the same subset of nodes is responsible each month for a considerable number of transactions (from 2% up to 16% of new transactions), we have a high User Repetition. This result reproduces the stability present outside the A2 interval. It also indicates that a different behavior occurs in User Repetition during the A2 interval, as the top-k nodes change more during the months in this interval.

This section showed User Repetition's behavior after applying the three methods described in Section 4.1. Method 1 showed us that an arbitrary transaction insertion is not able to decrease User Repetition considerably. This result reinforces that the decrease during the A2 interval is not a natural behavior in the network. Method 2 showed that to reproduce the behavior present during the A2 interval, we need to insert the random transactions in a more organized way. Method 3 reproduced the natural behavior during the intervals outside A2, where we have stability in the top-k rank. These results reinforce our initial intuition that the low values during the interval A2 are related to the fraudulent behavior of Mt.Gox.

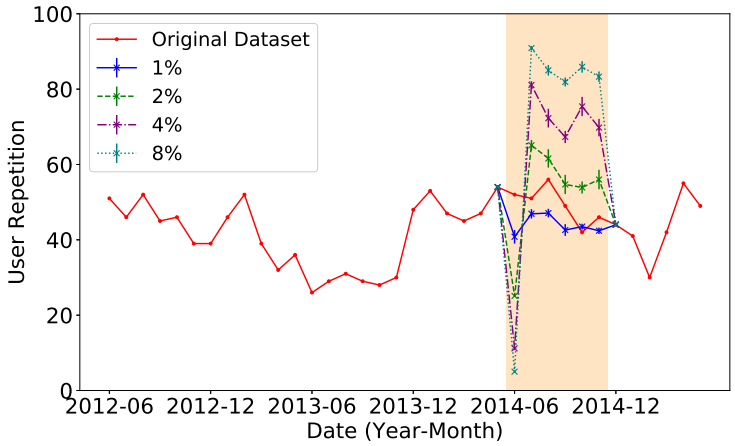

Fig. 5. User Repetition using Method 3

5 Case Study

This section provides a case study using a time-series model to forecast the User Repetition metric. This model allows an online behavior estimation of the best-ranked nodes and, together with the previous analysis, can indicate malicious activity in the network. Next, we first detail the data pre-processing steps. After that, we explain the forecast model and show an example of its execution.

5.1 Data pre-processing

As mentioned before, the previous sections perform a per-month analysis. The forecast model uses the User Repetition metrics per hour since this case study can be a base to an online mechanism. Hence, it is more helpful for this mechanism to have malicious activity clues in a shorter period.

In the first part of our data-preprocessing step, we repeat the graph analysis of Section 4, evaluating the User Repetition metric. Different from Section 4, we have one graph per hour instead of one graph per month. Hence, our dataset has the information of User Repetition in a one-hour interval. We employ a fixed-size sliding-window approach. It means that we break our dataset into sequences with the same length. We choose to use the last 24h to predict the User Repetition metric in the next hour. Consequently, we use the windows with a length of 25 (i.e., 24 for the input and 1 for the target). We also use the same approach to forecast for two hours in the future, based on the last 48h. Consequently, we use the windows with a length of 50 in this case.

Next, we split our dataset into three parts, with the data grouped into windows. One for training, with 80% of the data. Another for validation, with 10% of the data. And the last one for the test, with the remaining 10% of the data. We use the training dataset and the validation dataset to build the model (i.e.,

training and tuning the hyperparameters). We use the test dataset to evaluate the performance. We standardize all three parts using the mean and standard deviation calculated using the training dataset only.

We choose to use in the forecasting a baseline model and a Recurrent Neural Network (RNN) model, which is a widely used model for time-series analysis [9]. A time-series analysis can apply other approaches, but our focus is to present a case study of our User Repetition metric and not compare different methods.

5.2 User Repetition Forecasting

A Recurrent Neural Network (RNN) is a type of neural network that can recognize patterns in a sequence of data, where the notion of temporal order in the inputs affects the output. It happens because an RNN model takes as input the current input sequence but also information from the past. It means that the decision made at a time step t is affected by the decision made at a time step $t-1$.

Figure 6 shows an example of a model using a single RNN neuron. It is an unrolled through time representation [6]. In this figure, $x(t)$ is the input, $\hat{y}(t)$ the prediction, $h(t)$ is the transferred state, and the target is the correct value that we are trying to predict. This example shows that this model predicts one value of $x(t)$ in the future (i.e., $x(t+1)$). We can also note that we use the transferred state $h(t)$ as an input in the next time step. As $h(t)$ is a function of the current input $x(t)$ and the previous state $h(t-1)$, it illustrates the idea of memory in an RNN, as the state transferred to the next time step depends on the current value of the input and the previous state.

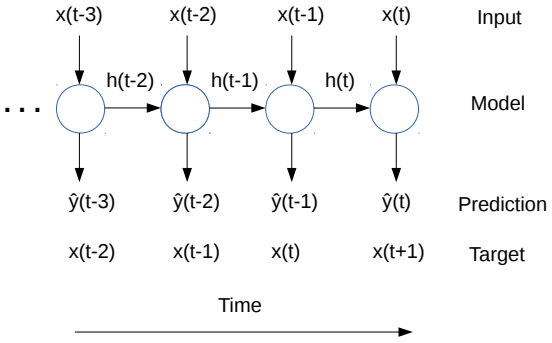

Fig. 6. Simple Recurrent Neural Network Model

RNNs are common in speech recognition, language modeling, stock price prediction, and other tasks based on time-series data [7]. The RNNs models have problems modeling long-term dependency, which means that they are not able to remember information for long periods [7]. One alternative is to use LSTM

(Long Short-Term Memory) networks [22]. LSTM is a special type of RNN that can learn long-term dependency. To be able to do that, an LSTM cell has its internal states split into two, $h_{(t)}$ and $c_{(t)}$. We can consider the first one as a short-term state and the second one as a long-term state. The long-term state allows the LSTM to remember information for long periods. At each timestep, the LSTM updates the $c_{(t)}$ by forgetting unnecessary information and adding new ones.

Given the above discussion, we use an LSTM model in our case study. We implement this model using Keras and TensorFlow libraries[3]. Our architecture has two hidden layers and 128 neurons each. We use the TANH (Hyperbolic Tangent) activation function [23] in both layers and a dense layer in the output. We choose to use the ADAM optimization algorithm [13]. The model is trained during 300 epochs, using the infrastructure of Google Colaboratory [4].

Figure 7 shows a one-window example of our prediction during the test. This window is used to predict one hour in the future. At each time step in this window, we plot the input (i.e., the 24 consecutive User Repetitions), the correct value of our target (i.e., 24 target values), and the predictions made by our model (i.e., 24 predictions). We can see that our model gets close to the Target value in most of the points.

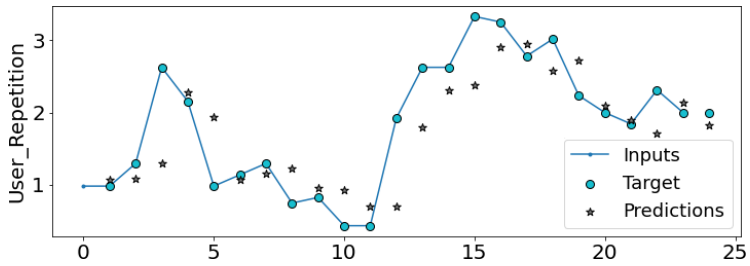

Fig. 7. User Repetition forecasting (1h)

This case study also uses a trivial baseline solution adopted in the literature [10]. As mentioned before, our focus is to show a case study of the User Repetition metric and not investigate the best forecasting method. As we are dealing with a time-series, we choose to use as a baseline an approach where we consider the forecast at time $t + 1$ to be equal to the value at time t. It means that $y(t + 1) = y(t)$. For the two hour prediction, we use $y(t + 2) = y(t + 1) = y(t)$. We evaluate the RNN model's performance and the baseline using the MAE (Mean Absolute Error) and the RMSE (Root Mean-Square Error). MAE computes the absolute average distance between the real value and the predicted value. RMSE evaluates the root squared difference between the real value and the predicted value.

[3] https://www.tensorflow.org/
[4] http://colab.research.google.com

Table 2 shows the performance of both models, considering our test set. In this table, we also provide the error for the forecast 2 hours in the future. We can see that the RNN model performs slightly better than the baseline model in both cases.

Table 2. Model evaluation

Metric	1h		2h	
	RNN	Baseline	RNN	Baseline
MAE	0.3650	0.3722	0.3990	0.4239
RMSE	0.4781	0.4998	0.5228	0.5646

In this section, we provided a case study using two approaches to forecast User Repetition. This analysis, together with the one made in Section 4, makes it possible for us to identify periods of suspicious activities by forecasting the User Repetition metric. For example, if the difference between our predicted and current User Repetition is high, we can consider that the network is in a suspicious state. If so, we can trigger a more detailed analysis to detect anomalous behavior.

6 Conclusion and Future Work

In this work, we analyzed the Bitcoin transaction graph, focusing on how the most important nodes (i.e., the top-ranked nodes) change each month. This analysis shows us a stable behavior in repeating these top nodes in almost all the months analyzed. However, this pattern changes during the period when Mt. Gox made its major number of fraudulent trades. We analyze the period where this pattern was changed by simulating new transactions in the network. This analysis gives us evidence that this change can be the consequence of market manipulation.

We also use an RNN model, together with a simples baseline one, to forecast User Repetition. As the results show, the RNN model was able to predict the behavior of our metric. With further analysis, the idea is that this prediction model can be extended to build an online model to identify periods of abnormal behavior.

As future work, we want to investigate the behavior of our metric in the entire blockchain to measure its ability to detect malicious activity. This analysis will also allow us to understand how facts not related to Bitcoin can influence our metrics. We also plan to perform the same analysis for other centrality metrics, such as Eigenvector Centrality and Closeness Centrality.

References

1. Blockchain.info: Bitcoin market price, https://blockchain.info/charts/market-price?timespan=all&format=json, accessed March 2021

2. Chen, W., Wu, J., Zheng, Z., Chen, C., Zhou, Y.: Market manipulation of Bitcoin: Evidence from mining the Mt. Gox transaction network. In: IEEE Conference on Computer Communications (INFOCOM). pp. 964–972 (2019)
3. Cong, L.W., He, Z., Li, J.: Decentralized mining in centralized pools. Tech. rep., National Bureau of Economic Research (2019)
4. Fleder, M., Kester, M.S., Pillai, S.: Bitcoin transaction graph analysis. arXiv preprint arXiv:1502.01657 (2015)
5. Gandal, N., Hamrick, J., Moore, T., Oberman, T.: Price manipulation in the Bitcoin ecosystem. Journal of Monetary Economics **95**, 86–96 (2018)
6. Géron, A.: Neural networks and deep learning. O'Reilly (2018)
7. Géron, A.: Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media (2019)
8. Gervais, A., Karame, G.O., Capkun, V., Capkun, S.: Is Bitcoin a decentralized currency? IEEE security & privacy **12**(3), 54–60 (2014)
9. Hewamalage, H., Bergmeir, C., Bandara, K.: Recurrent neural networks for time series forecasting: Current status and future directions. International Journal of Forecasting (2020)
10. Hyndman, R.J., Athanasopoulos, G.: Forecasting: principles and practice. OTexts (2018)
11. Juhász, P.L., Stéger, J., Kondor, D., Vattay, G.: A bayesian approach to identify Bitcoin users. PloS one **13**(12) (2018)
12. Khalilov, M.C.K., Levi, A.: A survey on anonymity and privacy in bitcoin-like digital cash systems. IEEE Communications Surveys & Tutorials **20**(3), 2543–2585 (2018)
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
14. Kondor, D., Pósfai, M., Csabai, I., Vattay, G.: Do the rich get richer? an empirical analysis of the bitcoin transaction network. PloS one **9**(2) (2014)
15. Lischke, M., Fabian, B.: Analyzing the Bitcoin network: The first four years. Future Internet **8**(1), 7 (2016)
16. Maesa, D.D.F., Marino, A., Ricci, L.: An analysis of the Bitcoin users graph: inferring unusual behaviours. In: International Workshop on Complex Networks and their Applications. pp. 749–760 (2016)
17. Maesa, D.D.F., Marino, A., Ricci, L.: Uncovering the Bitcoin blockchain: an analysis of the full users graph. In: 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA). pp. 537–546 (2016)
18. Maesa, D.D.F., Marino, A., Ricci, L.: Detecting artificial behaviours in the Bitcoin users graph. Online Social Networks and Media **3**, 63–74 (2017)
19. Moore, T., Christin, N.: Beware the middleman: Empirical analysis of bitcoin-exchange risk. In: International Conference on Financial Cryptography and Data Security. pp. 25–33 (2013)
20. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Tech. rep., Manubot (2019)
21. Project, B.: Bitcoin core, https://bitcoin.org/en/bitcoin-core, accessed March 2021
22. Quinn, J., McEachen, J., Fullan, M., Gardner, M., Drummy, M.: Dive into deep learning: Tools for engagement. Corwin Press (2019)
23. Sharma, S.: Activation functions in neural networks. Towards Data Science **6** (2017)
24. Suberg, W.: Mt. Gox trial update: Karpeles admits 'willy bot' existence, https://cointelegraph.com/news/mt-gox-trial-update-karpeles-admits-willy-bot-existence, accessed March 2021