Rheinische
Friedrich-Wilhelms-Universität Bonn

Master thesis

Let There be Trade: Towards Learning the Evolution of the Bitcoin Transactions Temporal Network

Author:
Ernane Luis Paixão Aguiar
Matr. Number: 2878646

First Examiner:
Prof. Dr. Christian Bauckhage
University of Bonn

Second Examiner:
Prof. Dr. Stefan Wrobel
University of Bonn

Declaration of Authorship

I declare that the work presented here is original and the result of my own investigations. Formulations and ideas taken from other sources are cited as such. It has not been submitted, either in part or whole, for a degree at this or any other university.

Location, Date

Signature

Acknowledgements

First of all I would like to thank my parents for providing me inexhaustible support and continuous encouragement throughout my years of study and the whole process of researching and writing. Specially my mother, this accomplishment would not have been possible without her. I am also grateful to my thesis advisor Cesar Ali Ojeda Marin from the Fraunhofer-Institute for Intelligent Analysis and Informations System (IAIS). His door was always open for me and whenever I had a question he was always there offering me his support.

I would like to also express my appreciation to my examiners: Prof. Dr. Christian Bauckhage and Prof. Dr. Stefan Wrobel. Thank you for giving me the opportunity to work with you on this publication.

Finally, I must express very profound gratitude to however is Satoshi Nakamoto for his brilliant contribution to mankind and for igniting me this passion to dive deeper into the Cryptocurrencies.

Thank you.

Abstract

Since the creation of Bitcoin in 2009, few contributions have been made towards modeling the network dynamics of the Bitcoin Blockchain. To date, no work has been done on trying to understand why Bitcoin agents make a trade with other agents by modeling the set of Bitcoin Transactions as a Temporal Network. In this work, we are interested in discovering why connections are made in the Bitcoin network rather than how it grows. To accomplish this, we derive a temporal network from the Bitcoin blockchain where we capture the quality of the dynamics by examining small temporal subgraph patterns. We then investigate the possibility of modeling such dynamics using a general probabilistic approach. Finally, we attempt to learn the dynamics of the real data and propose promising future work for predicting the before-mentioned dynamics.

Introduction

Bitcoin was proposed in 2008 as an experiment of a digital cash system entirely peer-to-peer, without a need for central authority, by an anonymous author self-entitled Satoshi Nakamoto [Nak08]. Since then, a lot of work has been done related to market analysis and privacy of Bitcoin, but few contributions have been made towards analyzing the network dynamics of Bitcoin. Additionally, at this date, no work has been published related to learning how the Bitcoins agents exchange with each order by modeling the set of Bitcoin Transactions as a Temporal Network. In this work, we address the problem of how the Bitcoin agents make connections rather than how the network grows. Since the algorithms utilized in this work were relatively new, we hold the opportunity to kick off an investigation towards learning the dynamics of the Bitcoin agents interactions.

Thus, we start our work by describing how the Bitcoin Protocol works in Section 1.2 and what kind of data it stores. Second, in Section 2.1 we propose an approach to derive a temporal network from the Bitcoin transaction data. We state why it is challenging to infer the agents based on the available data. Since the Bitcoin Protocol by design does not provide explicit information of which agent send to who.

Later on, on section 2.3 we explain an algorithm to count temporal subgraphs patterns, so-called temporal motifs, proposed by Paranjape et al.[PBL17]. With this algorithm in hands, we can capture the quality of the dynamics and use as a metric of such quality on the following section 3.1. Where we investigate the possibility of modeling such dynamics using a general probabilistic approach as ground zero proposed by Perra et al.[Per+12]. The work of Perra et al. provides us with a generative based model where with only a Distribution is capable of creating complex dynamics and it solely focuses on the creation of edges rather than on the growth of the network. We continue the section by proposing several modifications to the general activity driven model to attempt to achieve better results. Also running our modified models, we count the temporal motifs in order to measure how our models capture the quality of the real Bitcoin temporal network dynamics.

Finally, in Section 4.1 we attempt to learn the dynamics of the real Bitcoin temporal network. We report the work of Yang and Leskovec "Overlapping community detection at scale: a nonnegative matrix factorization approach"[YL13]. Later, we apply the algorithm proposed by Yang and Leskovec, called BIGCLAM, into our Bitcoin temporal network data sample and our simulations as well. Finally, we illustrate that BIGCLAM can be used to learn the probability of edges of the Bitcoin temporal network moreover state that BIGCLAM could allow us to predict the dynamics of the Bitcoin Transactions Temporal Network, proposing then a promising future work for predicting the Bitcoin dynamics.

Contents

List of Figures

1 Bitcoin

1.1 Introduction

The spirit of the Cypherpunk movement is to resist government interference with public privacy. Money is one of the aspects where government interferes a lot. Everything purchased online passes through a third party, e.g., a bank or a credit card company, which takes a cut from the transaction and can reject it for whatever reason. Additionally, people are cornered in an era where the only means of exchange, with privacy, is doing barter, because the whole financial system is monitored, regulated and adulterated for political reasons. This seems to be a frustrating and disappointing era for any Cypherpunk. For those who follow such ideals, the creation of means of exchange with privacy is the the final goal. Luckily, on 31 October 2008, a link to a paper authored by Satoshi Nakamoto titled *Bitcoin: A Peer-to-Peer Electronic Cash System*[Nak08] was posted to a cryptography mailing list[Nak][cry].

In his article, he detailed methods of using a fully peer-to-peer network of computers to generate a new electronic cash system, with no trusted third party. At that day the Cypherpunks observed that a new era, one of freedom and privacy, was born. On January 2009, the bitcoin network came into existence with the release of the first open source Bitcoin client and the issuance of the first Bitcoins [bloa][mai][sou]. On that day, the founder placed an important message indicating the true intent behind what he had just created.

The Times 03/Jan/2009 Chancellor on brink of second bailout for banks.

The London Times ran a cover story titled *"Chancellor on Brink of Second Bailout for Banks"*[tim]. This title was embedded into the very first transaction ever to be included in the new Bitcoin blockchain[blob], by Satoshi Nakamoto. This proved that the first block was mined after 3 January 2009, since that headline was not available before that time. The headline marks a moment where the Government of the United Kingdom planned to give a second financial assistance to the failing Banks in England to save them from collapse. Measure pumped billions of sterling pounds more into the selected bank's treasury. It is more evidence that the market was desperate and seeking a solution. The only problem was: they were looking for the wrong resolution. As *Murray Rothbard* states in his essay *Economic Depressions: Their Cause and Cure* [Rot09]:

> Almost all economists of the present day, treated the economy as a potentially workable, but always troublesome and rebellious patient, with a constant tendency to get into higher inflation or unemployment. Then, the function of the government is to be the wise old manager.

We saw in 2008 that governments were not so good *"wise"* managers, from all the chaotic 2008 depression. Bitcoin was born as an escape for those who want to keep the rewards of their hard work safe and away from greedy self-interested politicians.

Bitcoin was designed to be a digital encrypted money and to remove the need of central agencies. Early inventions of digital money like Bitgold[Sza08] or B-money[Dai98] were never implemented or relied on a central authority like Ecash[Cha95]. The most important challenging to solve without a central authority was the double-spending problem. How do we prove that a user that has paid for something with a specific coin has not used that same coin to buy something else at the same time, without having a central authority to monitor and approve the transactions? By solving these issues, Bitcoin became the very first successful decentralized digital currency. Instead of one central controlling party recording every transaction in one central ledger, all the users of the currency keep and record all of the transactions at the same time. Thus, as a result, any attempt to trick the community would be noticed since the community will verify this transaction and reject it. No one user, government or a bank can control the ledger. Bitcoin is an immutable open, decentralized ledger with censorship resistance maintained by a network of computers over the internet where anyone at any time can access that information. Using Bitcoin, users can transfer their money worldwide in a matter of minutes by paying a small fee.

1.2 How Bitcoin Works

The Bitcoin protocol insures that the total supply of Bitcoins remains fixed, by making it impossible to copy Bitcoins or to create more Bitcoins than is previously coded. The Bitcoin rules are formulated in the source code, and all computers which run the Bitcoin software maintain the validity of the rules and keep a copy of the ledger. Meaning all machines that keep the network running must agree on the same rules. Thus, this mechanism of decentralization of the computers is what guarantees the trust of the system. The Bitcoin open source core repository provides two types of software: the full node, and the mining node, both software are available for free and both software validates the Bitcoin protocol. Bitcoin protocol acts as a decentralized ledger where every machine running the protocol has a copy of the ledger, we call this ledger, the Blockchain.

1.3 Wallets

Besides, the full node and mining node softwares, there are another piece of software, the Wallet. To start using Bitcoin, you must download a Bitcoin Wallet. Wallets are not a type of node, and they are not responsible for maintaining the system, they act as a consumer of the network, and its primary task is to create an address, sign a transaction and broadcast it to the Bitcoin Network. Typically, a wallet transmits a transaction to a full node which is maintained by the wallet provider. They are made to work on regular

commercial computers and do not run any high energy consumption computation.

By generating an address, the user of the wallet now possesses (1) a private key, which acts as a key that unlocks his funds, and (2) a public key which serves as the router number of a bank account, the information that the user needs to provide to someone in order to receive a payment. Bear in mind that a wallet must keep the private key safe on the client computer and must never broadcast that information. Only the public key is transmitted and recorded in the ledger.

1.4 Addresses

A wallet is the software responsible for generating Bitcoin addresses. The process of creating a Bitcoin Address starts by producing a 256 bits random number. This number will generate a private key and a public key. The private key produces the public key using an *Elliptical Curve Function* which guarantees that the same private key will always generate the same public key, and that the matching private key is difficult to discover using only the public key. The private key must be generated using good randomness. Otherwise, it is possible that someone discovers the random number which generated the private key.

The address generated from the public key is what you use to send and receive bitcoins. It is called the Bitcoin Address. The Public keys are hashed using the *SHA 256 Hash Algorithm*, and the output is hashed using the *MD160 Hashing Algorithm* which results in a 160 bits Public Key Hash. Afterwards, the *Base 58 Check Algorithm* encodes it to a 34 characters string that is more human readable. This also excludes the characters: *0OIl*; the number zero, capital letter "o", capital letter "i" and low case "l" to avoid human mistakes. For the sake of simplification, we will name Bitcoin address as public key since this terminology is most popular among Bitcoin users. We can see on Figure 1.1 a diagram of this process simplified.

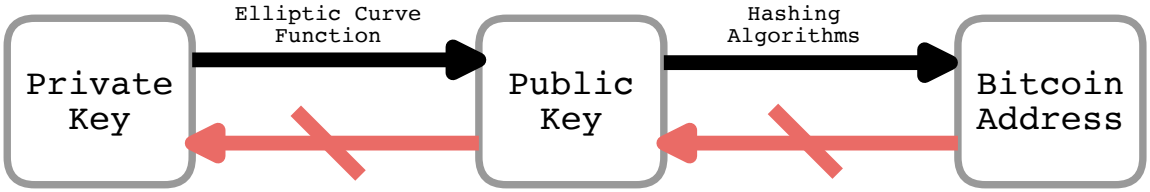


Figure 1.1: The generation process of a Bitcoin Address. Private Key is generated from a 256 bits random number. Sequentially, using the private key as input, the *Elliptical Curve function* outputs the Public Key. Finally, utilizing the public key as input, the *SHA 256 Hash Algorithm* output goes to the *MD160 Hashing Algorithm* which results in a 160 bits Public Key Hash. Then, the *Base 58 Check Algorithm* encodes this a 34 characters string more human readable and presentable form, where it excludes the characters: *0OIl*; the number zero, capital letter "o", capital letter "i" and low case "l" to avoid human mistakes. The result of this process is a Bitcoin Address. The red arrows indicate that the inverse process is not possible.

1.5 Blockchain

Bitcoin broadcasts a transaction to be included in the ledger in the form of a block. A block is a data structure which acts like a bucket by collecting several incoming new transactions. A block also has a reference to its parent block, and that parent block has a reference to its parent block and so on. This collection of references of blocks is called the *Blockchain*. The Blockchain is an ordered, back-linked list of blocks of transactions. The figure 1.2 Represents the blockchain in a visual way, where a Block has: its hash, the previous blocks' hash, and its list of transactions. The previous block hash field creates the actual chain between the blocks. Thus, Block 12 is linked back to Black 11, and Block 11 is connected to Block 10.

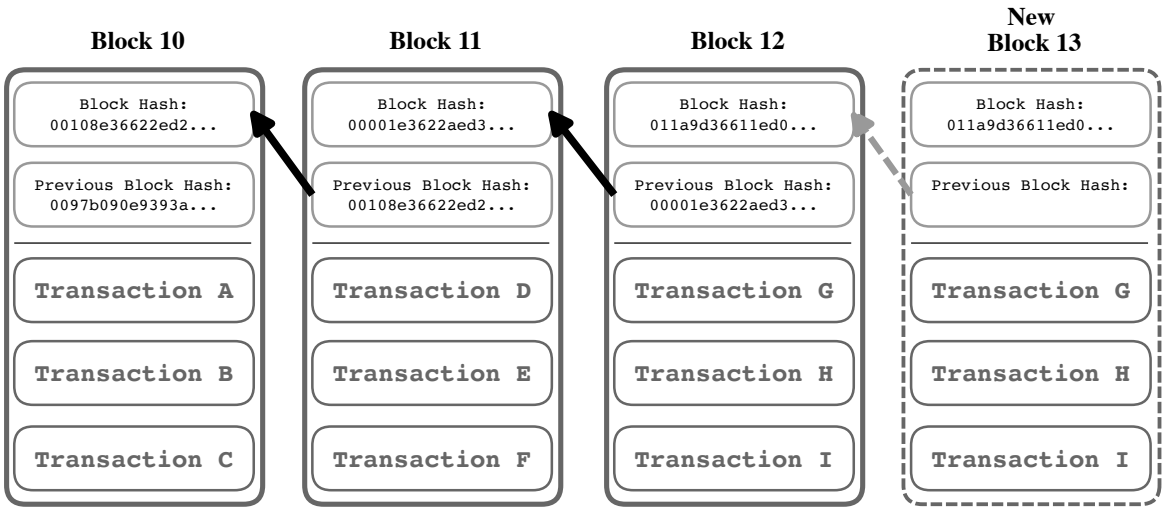


Figure 1.2: Graphical representation of the Bitcoin Blockchain Data Structure. Every Block has its hash, the previous block hash, and its list of transactions. The previous block hash field ensures the chain of blocks. Thus, Block 12 is linked back to Black 11, and Block 11 is connected to Block 10. Block 13 represents a new mined block about to be inserted to the chain.

1.6 Mining Bitcoins

New blocks are added to the blockchain by so-called "mining nodes". Mining nodes are machines running the Bitcoin mining software. This mining software is usually run on specialized computers. Mining nodes are responsible for continuously receiving new transactions from the network and validating all transactions by checking the integrity of the Bitcoin rules. Thus, mining contributes to the security of the network by accepting valid transactions or rejecting invalid transactions. However, *there is no free lunch* and keeping those computers working for the network by validating the transactions has a cost. This validation job requires the miners to solve a hard computation task which uses electricity. The reward for this work is the ability of a miner to create new Bitcoins. For each block mined, the miners are rewarded with a certain amount of Bitcoins. How much is the miner rewarded for mining a block? How much Bitcoins will ever be created? How hard is the problem that must be solved? These rules of the Bitcoin network can be seen in more details in https://en.bitcoin.it/wiki/Main_Page.

We can see in Figure 1.2 the representation of a new mined block (Block 13) about to be inserted into the Blockchain. This block will be included in the final state of the Blockchain only if the majority of full nodes on the network approve the validity of this block. Remember that every participant of the network keeps a copy of the Blockchain. Thus, the addition of the local copy Blockchain of the miner must be broadcasted to the others and validated by them. Figure 1.3 exemplifies the process of adding a new block to the Blockchain. In order for a newly mined block to be accepted in the final

blockchain state, the full nodes must validate this block against the rules of the protocol. If the majority of the full nodes validate this block as correct, then this block is accepted and added to the final Blockchain state and broadcast to the other nodes. If the majority finds an inconsistency, then this block is rejected. However, if the nodes do not reach a consensus, this results in a rupture of the protocol, where part of the participants agree on a set of rules and another part agrees on another arrangement. This break of consensus is called a *Hard fork*, which leads to a bifurcation of Blockchain, and these two chains continue to be maintained by the same network. These two chains share the same past but differ in the future from the moment of the split.

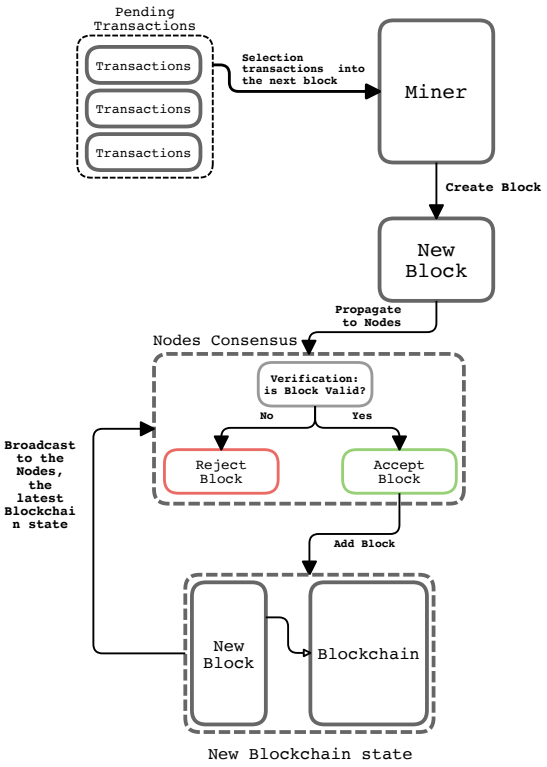


Figure 1.3: Bitcoin Mining Flowchart: process of adding a new block to the Blockchain. In order for a newly mined block to be accepted to the final blockchain state, the full nodes must validate this block against the rules of the protocol. If the majority of the full nodes validate this block as correct, then this block is accepted and added to the final Blockchain state and broadcasted to the other nodes. If the majority finds an inconsistency, then this block is rejected.

1.7 Full Node

A full node is a *watchmen* of the miners. It has the responsibility to check if the block created by the miner follows the Bitcoin rules. If the miner of newly generated block added more Bitcoin than what is allowed, full nodes would check this block, and then

mark this block as invalid. Since every node on the network has the same information, the other full nodes will achieve the same result as the first full node. Then a consensus would be reached about this invalid block. Thus, this block will not be included in the final state of the *Blockchain*. The full node software does not require the same computation power as the mining node, for that reason the network does not reward full nodes. Figure 1.4 illustrates the overview of the Bitcoin network. The network is a combination of machines running the full nodes and miner nodes software. Each node keeps a local copy of the final state of the Blockchain; they are responsible for maintaining the network. However, only miners are allowed to modify the Blockchain state. Finally, the wallets are the end consumers of the network. They are responsible for creating Bitcoin addresses and broadcasting transactions among the addresses.

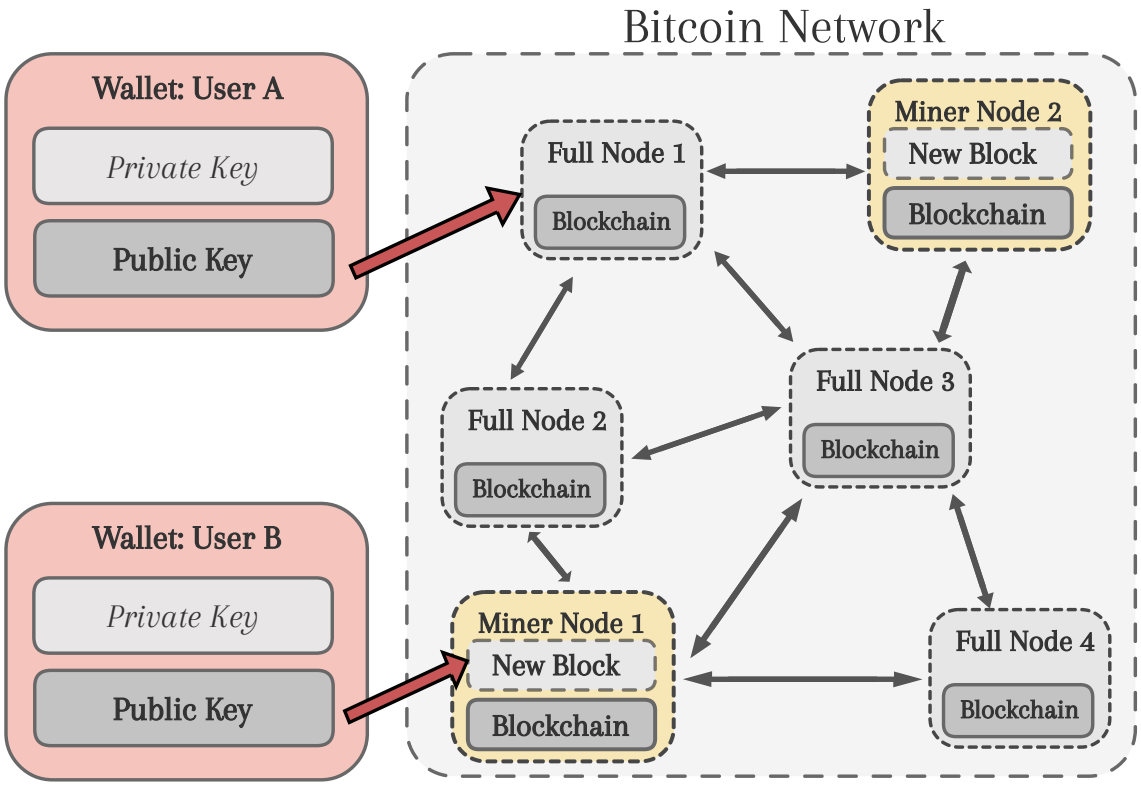


Figure 1.4: Bitcoin Network Overview: a combination of machines running the full nodes and miner nodes software.

1.8 Transactions

As stated on the Bitcoin wiki page,

> A transaction is a transfer of Bitcoin value that is broadcast to the network and collected into blocks.

Since transactions are transmitted and recorded on the Blockchain, they are public information, meaning, they are not encrypted. A Transaction is a transfer of ownership from one public key (i.e. address) to another. It typically contains several inputs and two outputs. An input is a reference to the old owner address from a previous transaction. Thus, inputs reference the outputs of previous transactions. Furthermore, an output is a reference to the new owners' address; this address will now hold a balance made by the transaction. Figure 1.5 helps to understand the relationship between the inputs and outputs of a transaction. Because of this reference, we can calculate the available Bitcoin balance of an address by looking at all past transactions on the Blockchain where the target address appears as output. The balance will be the sum of all the values found.

As we can see in Figure 1.6, the available balance of the wallet is the sum of the output values located on transaction A and transaction B. Note that a wallet can generate several public keys from the same private key, in this case, address C and address E belong to the same private key. The final balance of Address G is 2 Bitcoins.

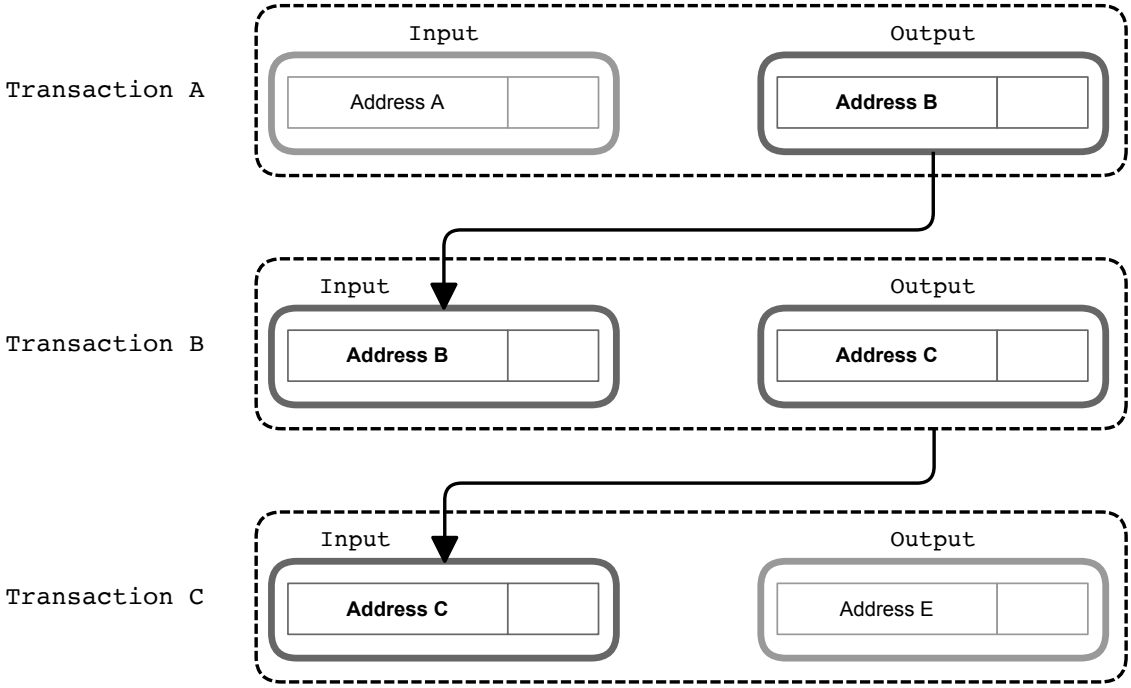


Figure 1.5: The input of transaction C is the output of transaction B meaning address C used the funds received from address B to send to address E. Analogously, transaction B with transaction A.

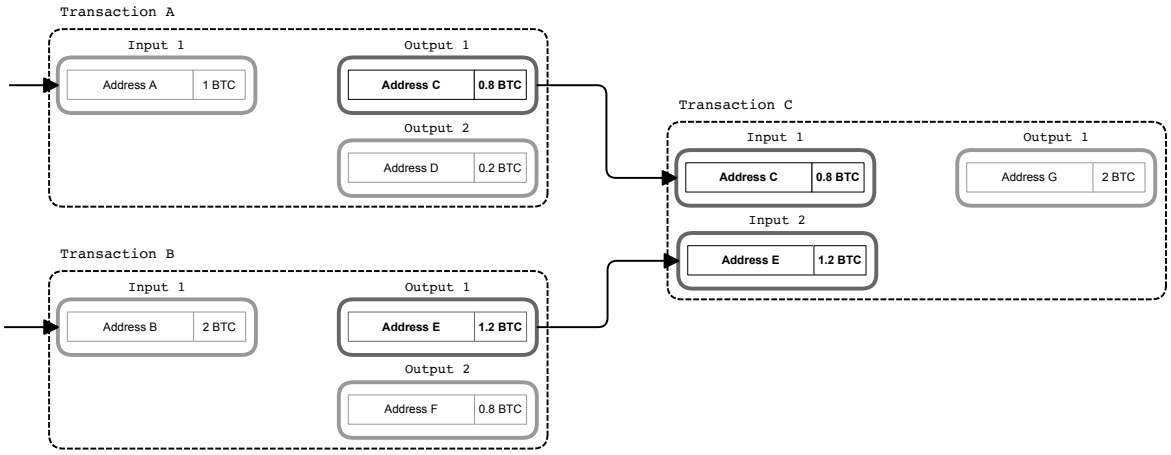


Figure 1.6: A chain of transactions, the available balance of address C is the sum of the output value located on transaction A and for address E is the sum of the output located on transaction B. Address C and E are owned by the same private key. The Final balance of Address G is 2 Bitcoins.

2 Unveiling the Bitcoin Transaction Temporal Network

Our motivation in this analysis is that behind the transaction data stored on the Bitcoin Blockchain there is a hidden network which can provide valuable information on how agents made transactions through the usage of Bitcoins. As a consequence, we are interested in performing an analysis on the relationships between Bitcoin addresses. In this section, we introduce a way to translate the Bitcoin transactions into a Graph data structure which takes into consideration the time that the edge were created, and characterize this graph.

2.1 Bitcoin as a Graph

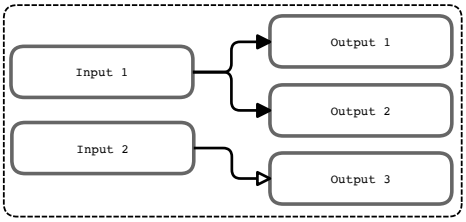
Our analysis begins by realizing that the Bitcoin transactions can be seen as a Graph. We saw in Section 1.8, that a Bitcoin transaction is a record of input and output addresses. The addresses can be translated into the vertices of our Graph. The transferring of funds from *Address A* to *Address B* is converted to a directed edge from *Address A* to *Address B*. By design, the Bitcoin protocol allows a transaction to have more than one output address. Meaning, a user could transfer to more than one different user on the same transaction. Meaning, the relationship between inputs and outputs of a transaction is *N to M*. For that reason, we can not discover from which input the funds are going exactly to which output. Figure 2.1, visualizes this problem. Thus, we applied the same technique proposed on [Kon+14a; Kon+14b], where they state:

> A transaction with two inputs and three outputs results in six edges; an edge may appear multiple times, with the corresponding transaction.

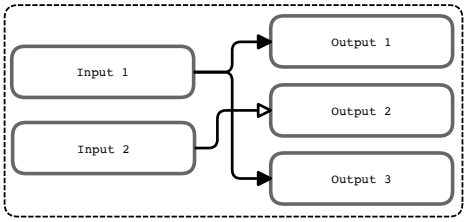
In other words, we create an edge for all the possible combinations of inputs to outputs. Figure 2.1 visualizes all the possible inputs and outputs pairs problem.

However, Bitcoin users do not generate a static graph; the same address is allowed to make a transfer again in the future (if there are funds available). Consequently, the Bitcoin graph we are trying to create is not a simple graph, it is a graph that changes over time; one that evolves. Hence we must take into consideration the time. In our case, the time of the transaction that was broadcast on the network.

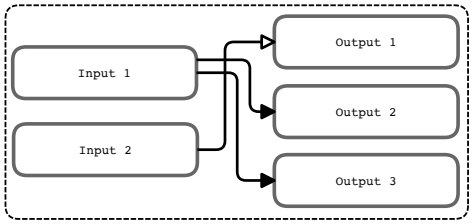
Hence, our final graph is a list of edges with timestamps. Note that we are not taking into consideration the amount of money transferred from one address to another, our sole interest is on the relationships between the vertices.



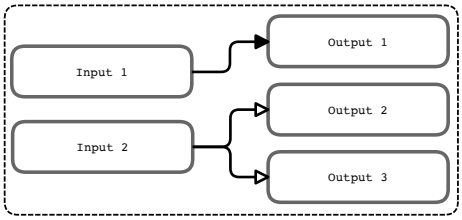
(a) First possible combination.



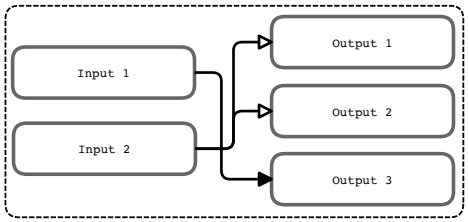
(b) Second possible combination.



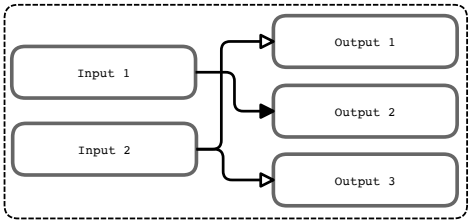
(c) Third possible combination.



(d) Fourth possible combination.



(e) Fifth possible combination.



(f) Sixth possible combination.

Figure 2.1: Example of a transaction with two inputs and three outputs with all the possible arrangements of edges. By design, Bitcoin allows an edge to appear multiple times, within the corresponding transaction. Thus, we can not discover from which input the funds are going exactly to which output.

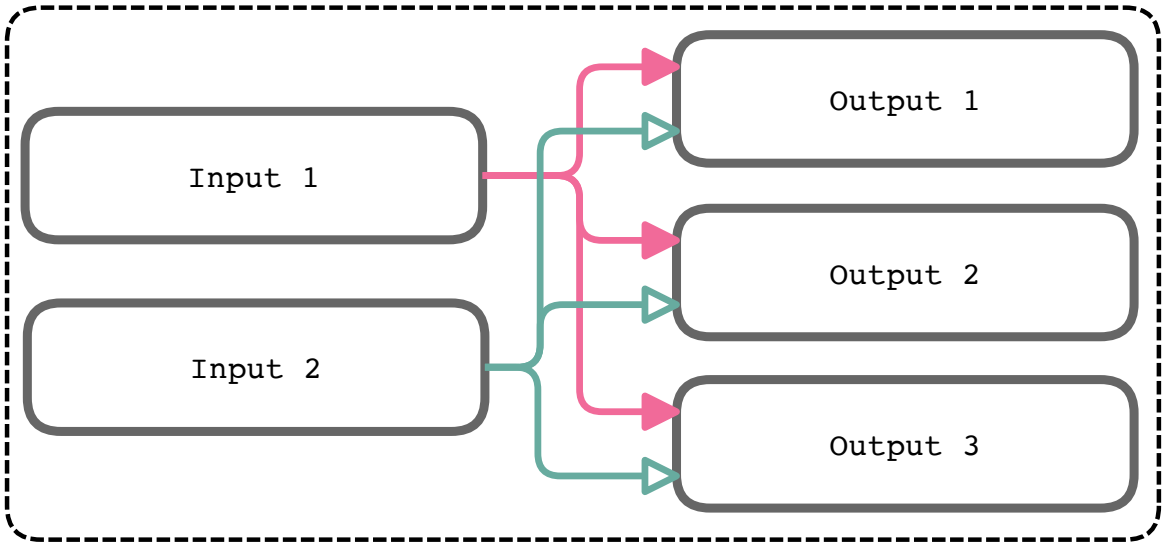


Figure 2.2: The Solution of fig.2.1, a transaction with two inputs and three outputs results in six edges. We create an edge from each of the inputs to all of the outputs.

2.2 Temporal Networks

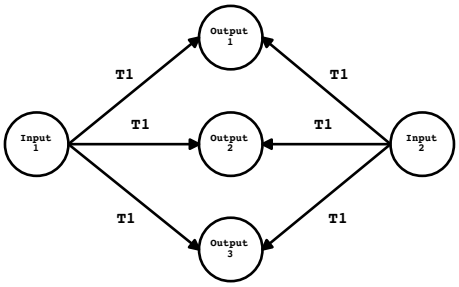
Continuing our investigation, we modeled the Bitcoin transactions inputs and outputs addresses into a list of edges with timestamps. Then, this edge list can be seen as a graph that changes its edges over time. Hence, we shaped the Bitcoin transactions data into an instance of a Temporal Network [HS12].

Temporal Network is a type of network where edges are not continuously active. The edges have an extra attribute which tells *when* this edge happened in time. On the contrary of traditional network theory, temporal network approaches target the analysis of information, about *when* things happen. The motivation behind is the edge activations affect the dynamics of the network. Therefore, the central interest is the flow of information.

In static networks, network motifs, small induced subgraphs patterns occurring in large network structures [BGL16; Mil+02; Wu+10], are crucial to understanding the structure and behavior of these complex systems [PBL17]. Despite the abundant literature on static graphs, existing methods either see the networks as strictly growing where nodes connect once and stay connected forever [BA99a; Jac+15; LKF07], or see temporal information as a sequence of graph snapshots [Ara+14; DKA11; TBK07]. Those methods are not enough to assess our approach, as we want to capture information that takes into consideration that edges between nodes dynamically change over time. Luckily, the proposed work from Paranjape et al.[PBL17] enables us to count temporal motifs in our Bitcoin temporal graph considering this. Our interest in this work relies on their results, where it shows that networks from the same domain tend to have similar motif counts, and networks from different domains have significantly different motif counts. With this result in hands, it allows us to have a metric when modeling the Bit-

Edge	From	To	Timestamp
#1	Input 1	Output 1	T1
#2	Input 1	Output 2	T1
#3	Input 1	Output 3	T1
#4	Input 2	Output 1	T1
#5	Input 2	Output 2	T1
#6	Input 2	Output 3	T1

(a) The result of Figure 2.2 as an edge list containing six edges with timestamps.



(b) The result edge list (A) as a timestamped subgraph.

Figure 2.3: The example of converting the same transaction with two inputs and three outputs from Figure 2.2 into (1) the form of edge list timestamped and (2) the form of a subgraph. Where inputs and outputs are Bitcoin addresses and the timestamp is the time where this transaction occurred.

coin temporal network i.e, we will assess the quality of the dynamics by reproducing the temporal motifs. We investigate the application of the proposed methodology [PBL17] for analyzing motifs in our Bitcoin temporal network.

2.3 Motifs in Temporal Networks

Temporal network motifs are induced subgraphs on sequences of temporal edges, as is definedion the work of Paranjape et al. It is based on generalizing the concept of static network motifs towards temporal networks[PBL17]. Before we continue our study, we must first define the vocabulary used by Paranjape et al.

2.3.1 Definitions

Definition 2.3.1. Temporal Edges: A timestamped directed edge between an ordered pair of nodes (a line from the Table 2.3a). Formally, given a node set V, a tuple (u_i, v_i, t_i), $i = 1,\ldots,m$, is a temporal edge if u_i and v_i are elements of V and t_i is a timestamp in $\mathbb{R}$. There can be many temporal edges directed from u to v, and we refer to them as edges between u and v. We assume that the timestamps t_i are unique.

Definition 2.3.2. Temporal Graph: A collection of temporal edges (Fig. 2.3b, Fig. 2.5a). Formally, a temporal graph T on a node set V is a collection of tuples (u_i, v_i, t_i), $i = 1,\ldots,m$, where each u_i and v_i are elements of V and each t_i is a timestamp in $\mathbb{R}$.

Definition 2.3.3. δ-Temporal Motifs: Classes of isomorphic subgraphs (Fig. 2.5b) where it takes into account the ordering of edges.

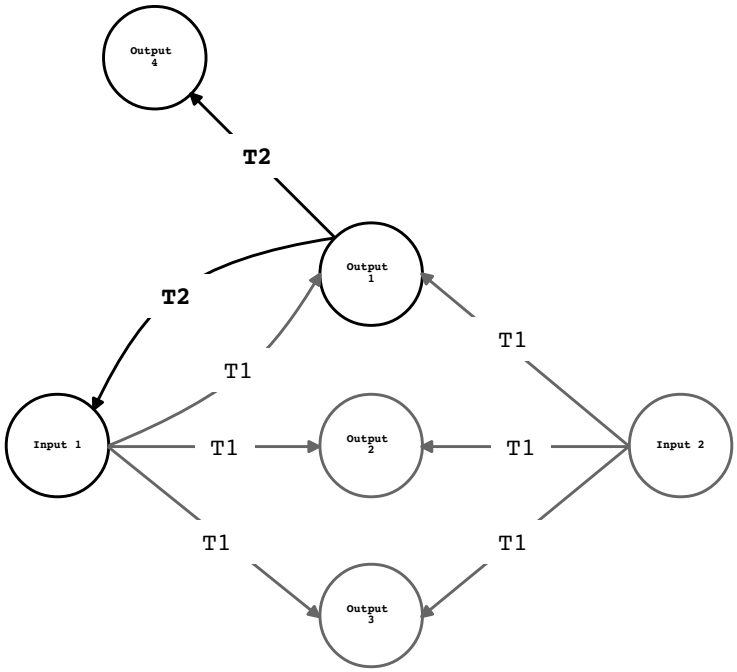


Figure 2.4: Example of the resulting temporal network of two transactions: *T1*(two inputs and three outputs) and *T2*(one input and two outputs). Where the Output 1 of the *T1*, is the Input 1 of *T2*.

Definition 2.3.4. Instance of δ-Temporal Motif: A collection of edges in a given temporal graph is an instance of a δ-temporal motif M that matches the same edge pattern and all of the edges occur in the right order within the time window (Fig. 2.5c). Properly, any time-ordered sequence $S = (w_1, x_1, t'_1), \ldots, (w_l, x_l, t'_l)$ of l unique edges is an instance of the motif $M = (u_1, v_1, t_1), \ldots, (u_l, v_l, t_l)$ if:

1. There exists a bijection f on the vertices such that $f(w_i) = u_i$ and $f(x_i) = v_i$, i = 1,...,l;

2. All the edges occur within δ time, i.e., $t'_l - t'_1 \leq \delta$;

Definition 2.3.5. Induced Static Graph: Finally, by ignoring timestamps and duplicate edges, the temporal graph induces a standard directed graph, which we call the static graph G of T with static edges, i.e., (u, v) is an edge in G if and only if there is some temporal edge (u, v, t) in T.

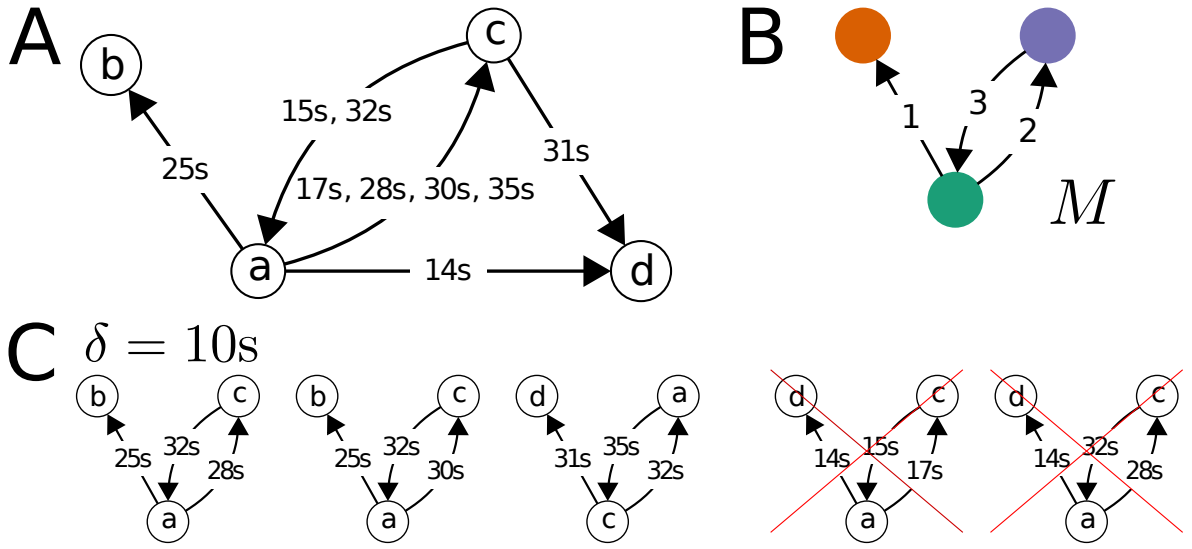


Figure 2.5: A) A temporal graph with nine temporal edges. Each edge has a timestamp (listed here in seconds). B) Example of a 3-node, 3-edge δ-temporal motif M. The edge labels correspond to the ordering of the edges. C) Instances of the δ-temporal motif M in the graph for $\delta = 10$ seconds. The crossed-out patterns are not instances of M because either the edge sequence is out of order or the edges do not all occur within the time window δ.

2.3.2 Counting Temporal Motifs

The work of Paranjape et al. describes a general algorithm for the problem of counting how many times each δ-temporal motif occurs in a given temporal network. The central goal is to count the number of ordered length-l of edges that are instances of a given k-node, l-edge, δ-temporal motif, from a given temporal graph T. Even though the proposed algorithm is a general approach, we are only interested in small patterns.

The mentioned algorithm for counting instances of temporal motifs in a temporal graph T has as *Input*: Sequence S' of edges $(u_1, v_1, t_1),...,(u_L, v_L, t_L)$ with $t_1 < ... < t_L$, time window δ and as *Output*: A $6X6$ matrix storing the number of instances of each 2-node and 3-node, 3-edge δ-temporal motifs contained in the sequence. Figure 2.6 provides a visualization of all motifs from the 6X6 matrix output.

The algorithm can assess the count respecting different types of motifs: 2-node, stars, and triangles. Furthermore, the algorithm is divided into three procedures which evaluates each kind of motif. The algorithm starts with a *general approach* that follows the above steps:

1. Identify all instances H' of the static motif H induced by M within the static graph G induced by the temporal graph T.

2. For each static motif instance H', gather all temporal edges between pairs of nodes forming an edge in H' into an ordered sequence $S' = (u_1, v_1, t_1),...,(u_L, v_L, t_L)$

3. Count the number of subsequences of edges in S' occurring within delta time units that correspond to instances of M.

Any induced graph H of a 2-node δ-temporal motif is either a single or a bidirectional edge. Then, if the motif is of type 2-node motifs, the algorithm runs those steps:

1. For each pair of nodes u and v for which there is at least one edge, gather and sort the edges in either direction between u and v;

2. Call the above general approach with these edges;

3. Then, sum all the returns of (2) to obtain the total motif count.

However, if the type of the k-node, l-edge motif is a star type, the induced static graph H consists of a center node and $k-1$ neighbors, where edges may occur in either direction between the center node and a neighbor node. Then, the algorithm use this information to computes the following steps:

1. For each node u in the static graph and for each unique set of $k-1$ neighbors, gather and sort the edges in either direction between u and the neighbors;

2. Call the general approach with these edges;

3. Each return counts from (2) are summed across all center nodes.

Finally, in triangle motifs types, the induced graph H consists of 3 nodes and at least one directed edge between any pair of nodes. The induced static graph H of M contains at least three and at most six static edges. Thus, if the type of the motif is a triangle, the algorithm evaluates those steps:

1. Use the algorithm proposed by [Lat08] to find all triangles in the static graph G induced by T;

2. For each triangle (u, v, w), merge all temporal edges from each pair of nodes to get a time-sorted list of edges;

3. Use general approach to count the number of instances of M.

These methods are able to count the number of instances of any k-node, l-edge δ-temporal motif. However, this method is only optimal for 2-node and 3-node motifs, and the computational cost may be expensive for other motif types. Consequently, the output of the algorithm only counts instances of 2-node and 3-node, 3-edge delta-temporal motifs. For more details of the algorithm, see [PBL17].

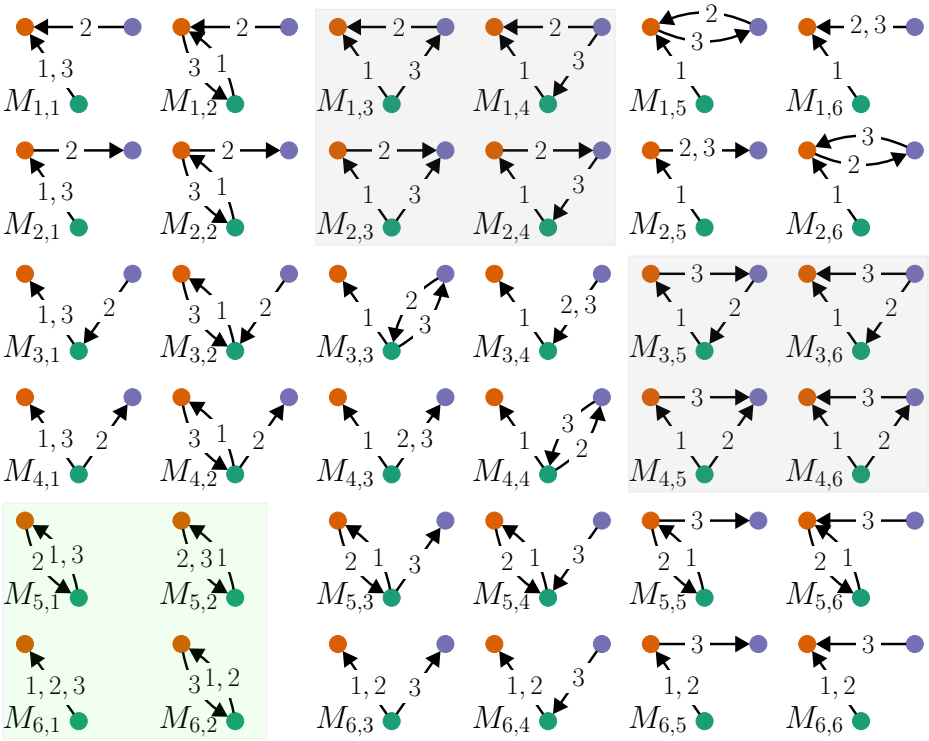


Figure 2.6: The output of the algorithm proposed above: A matrix with all 2-node and 3-node, 3-edge δ-temporal motifs counts. The 24 white background motifs are stars. The green background highlights the four 2-node motifs (bottom left) and the grey background highlights the eight triangles. All the 36 motifs are indexed $M_{i,j}$ by 6 rows and 6 columns. The first edge in each motif is from the green to the orange node. The second edge is the same along each row, and the third edge is the same along each column.

What is exciting for us from the results achieved by [PBL17] was, applying the motif counting algorithm in a variety of temporal network datasets they were able to find that the number of instances of different δ-temporal motifs reveals fundamental mechanisms of the networks. The algorithm was applied on a variety of datasets: a collection of emails between members of a European research institution, telephone call records for a major

European service provider, a group of posts from StackOverflow, all the payments made up to October 19, 2014 from the Bitcoin Blockchain, and four more different datasets.

They calculate the median time for a node to become part of three edges in the dataset to discover the most suitable delta time window to run the algorithm. Their results show $\delta = 1$ hour for the time window. An empirical observation of the motif counts was made to examine the distribution of 2-node and 3-node, 3-edge motif instance counts from the target datasets. Two collections of datasets from similar domains were formed. The counts of the 36 different motifs were normalized to build the count distributions. As a result of those distributions, they conclude that temporal networks from the same domain have similar counts. This result confirms their motivation, where it resembles the results from [Mil+04; Vaz+04; Yav+14], static graphs from related domains tend to have similar motif count distributions.

2.4 Motifs in the Bitcoin Transaction Temporal Network

In the previous section, we showed a method for analyzing temporal networks. In Section 2.1 we showed a procedure to model the bitcoin transaction data as a temporal graph. To combine these two results, we start by downloading the Bitcoin transaction data from `https://blockchain.info/api/blockchain_api`. Using this API, we were able to download the raw JSON data of all the transactions between *5 January 2017* and *12 August 2017*, totalling 220 days of transactions. For each day of raw data, we follow the method proposed in section 2.1 to obtain the temporal network of the Bitcoin transactions. The outcome is a collection of 220 files, where each file is a list of temporal edges, which represent the temporal network (see Definition 2.3.1). Afterwards, we run the Algorithm 2.3.2 for each Bitcoin temporal network day file, with parameter $\delta = 3600$ seconds, as Subsection 2.3.2 describes.

In order to describe the network, we calculated the number of nodes, number of edges, and number of triangles over time. Resulting in three time-series. Figure 2.7 shows the number of nodes in our dataset, from 220 days, the maximum number of nodes is 940,446 on 23 May, the minimum 482,455 on 1 August, and average 697,657.

Sequentially, Figure 2.8 shows the number of edges with as a maximum 3,092,194 on 11 August, a minimum of 960,204 on 15 January and 1 August as third lowest, and an average of 1,647,875. Lastly, Figure 2.9 shows the number of triangle motifs where the maximum is 3,887,064 on 4 August, the minimum 114,873 on 1 August, and with average 735,590. From our dataset, we can see 1 August is the day with the lowest usage of the Bitcoin protocol.

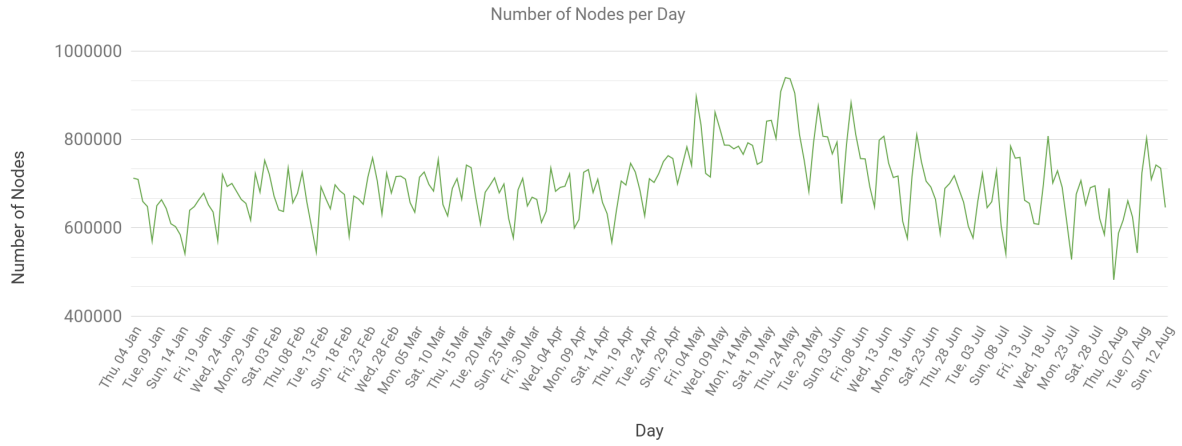


Figure 2.7: Time-series of the number of nodes computed from our bitcoin dataset. The maximum number of nodes is 940,446 on 23 May, the minimum 482,455 on 1 August, and average 697,657 nodes.

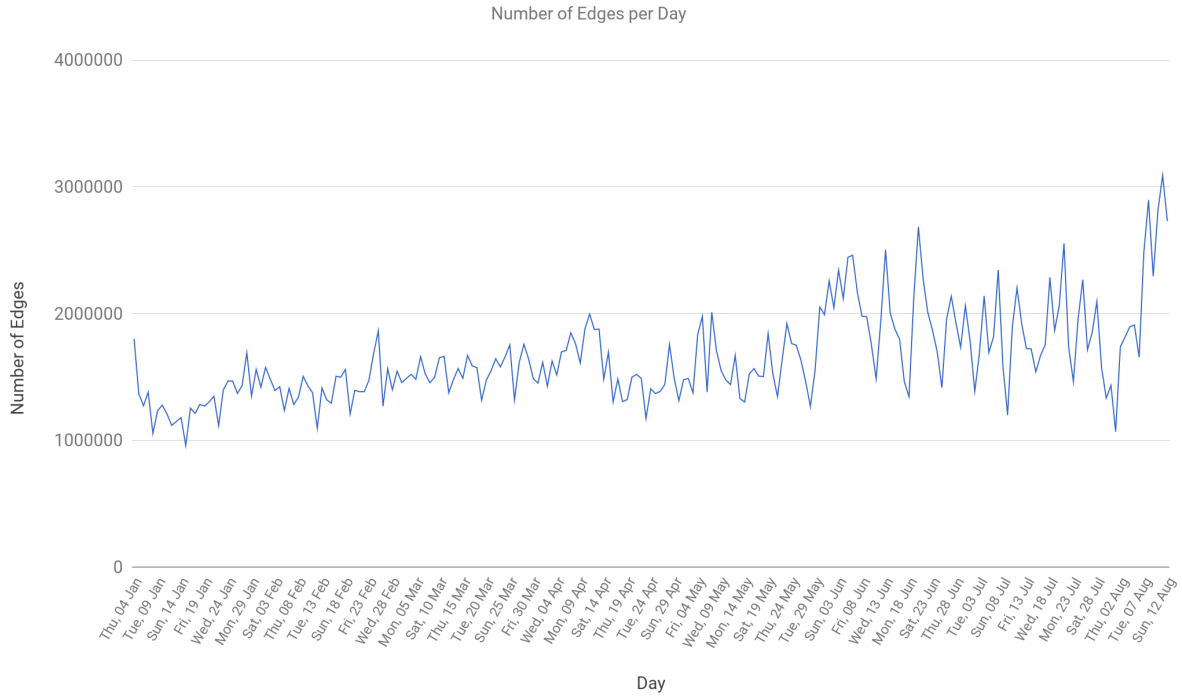


Figure 2.8: Time-series of the number of edges computed from our bitcoin dataset. The number of edges with maximum 3,092,194 on 11 August, the minimum 960,204 on 15 January and 1 August as third lowest, and average 1,647,875 edges.

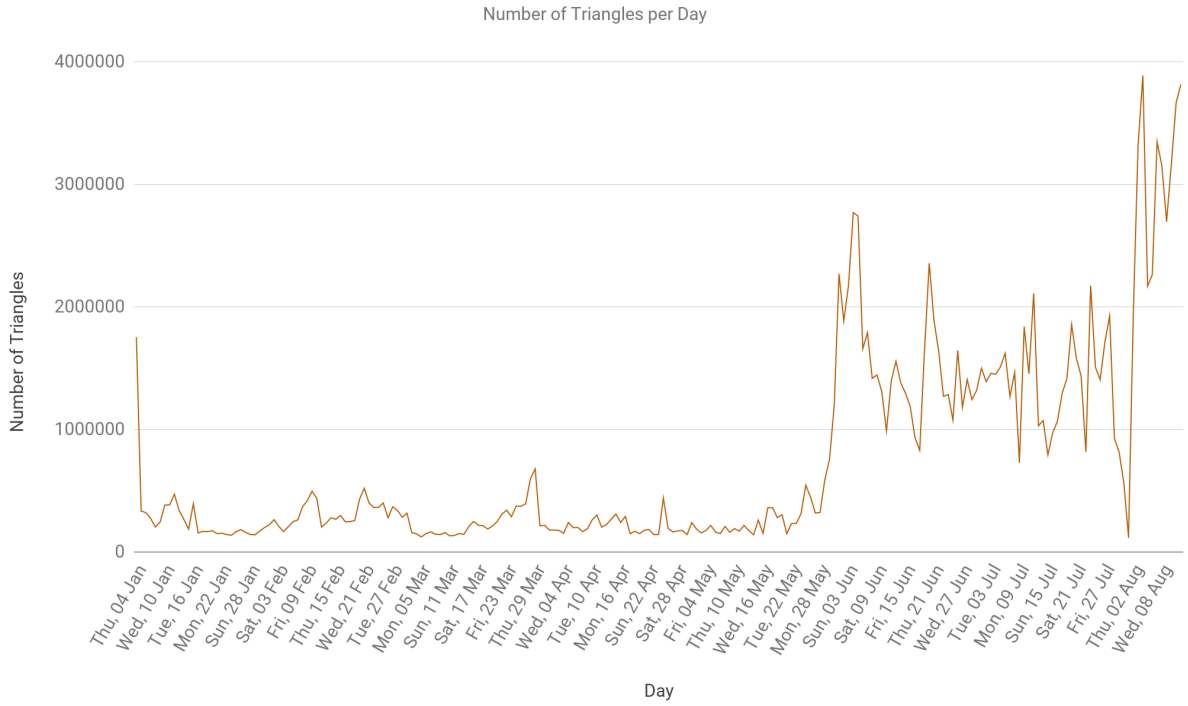


Figure 2.9: Time-series of the number of triangles computed from our bitcoin dataset. The maximum is 3,887,064 on 4 August, the minimum 114,873 on 1 August, and with average 735,590 triangles.

Lastly, we compute the temporal motifs by selecting 3 data samples. We found for day A) May 4 2017, the motifs $M_{4,3}$(star), $M_{3,3}$(cycle), $M_{4,4}$(cycle), $M_{3,4}$(star), $M_{4,1}$(star), $M_{2,1}$(star) were the most counted and $M_{6,2}$(cycle 2-nodes) the less counted. For day B) June 16 2017, we resulted in $M_{6,6}$, $M_{5,4}$(cycle), $M_{5,6}$(cycle), $M_{6,4}$(star), $M_{1,1}$(star), $M_{1,2}$(cycle) as most counted and $M_{3,5}$(triangle) as less counted. Finally, for day C) August 1 2017, the most counted were motifs $M_{6,6}$(star), $M_{1,6}$(star), $M_{1,1}$(star), $M_{4,1}$(star), $M_{4,3}$(star), $M_{6,3}$(star) and less counted was $M_{3,5}$(triangle), as it was in day B) as well. Note here that triangles motifs were never high counted.

Our attention in this work is how the temporal motifs in the Bitcoin transactions temporal network *change over time*. For that reason, our analysis is solely over subgraphs by days. Lastly, we normalize our motifs counts data. For better visualization of the behavior of the motifs, we compute the variance of the motifs and selected 6 highest variance motifs: $M_{1,6}$,$M_{6,6}$, $M_{1,1}$, $M_{2,5}$, $M_{1,5}$ and $M_{2,6}$. Surprisingly, the motifs with highest variance had a significant increase at the end of May 2017, as we can see the first plateau, and a vital increase, the second plateau, around August 2017 in its counts as Figure 2.10 shows. We can infer that: those days precisely match with meaningful changes to the Bitcoin protocol. Accurately, on 22 May 2017, the Bitcoin Implementation Proposal Number 91 was created. This proposal, allows the Bitcoin miners to agree or not to the new changes on the Bitcoin protocol, having until 15 November 2017 to decide. Furthermore, on 1 August 2017, the motifs accurately capture a modification of

the network. On that day, the first Bitcoin break of consensus happened [1, 2, 3, 4, 5], which ended in the split of the Bitcoin Blockchain into two parallels Blockchains, this is the reflex of the mining nodes not finding a consensus among them, where some of them agreed and some other disagreed, on the rules proposed on 22 May 2017. Thus, resulting in rupture of the Blockchain. This action produced a second cryptocurrency sibling of the Bitcoin, called Bitcoin Cash. Additionally, this change on the protocol pushed a halt on several Cryptocurrencies Exchanges, and Wallets. These reasons can explain why 1 August was the day with the lowest usage of the Bitcoin protocol.

Therefore, we conclude by showing empirical evidence that the temporal motifs proposed by [PBL17] were able to reveal fundamental structural changes of the Bitcoin transactions temporal network. Thus, confirming our intuition.

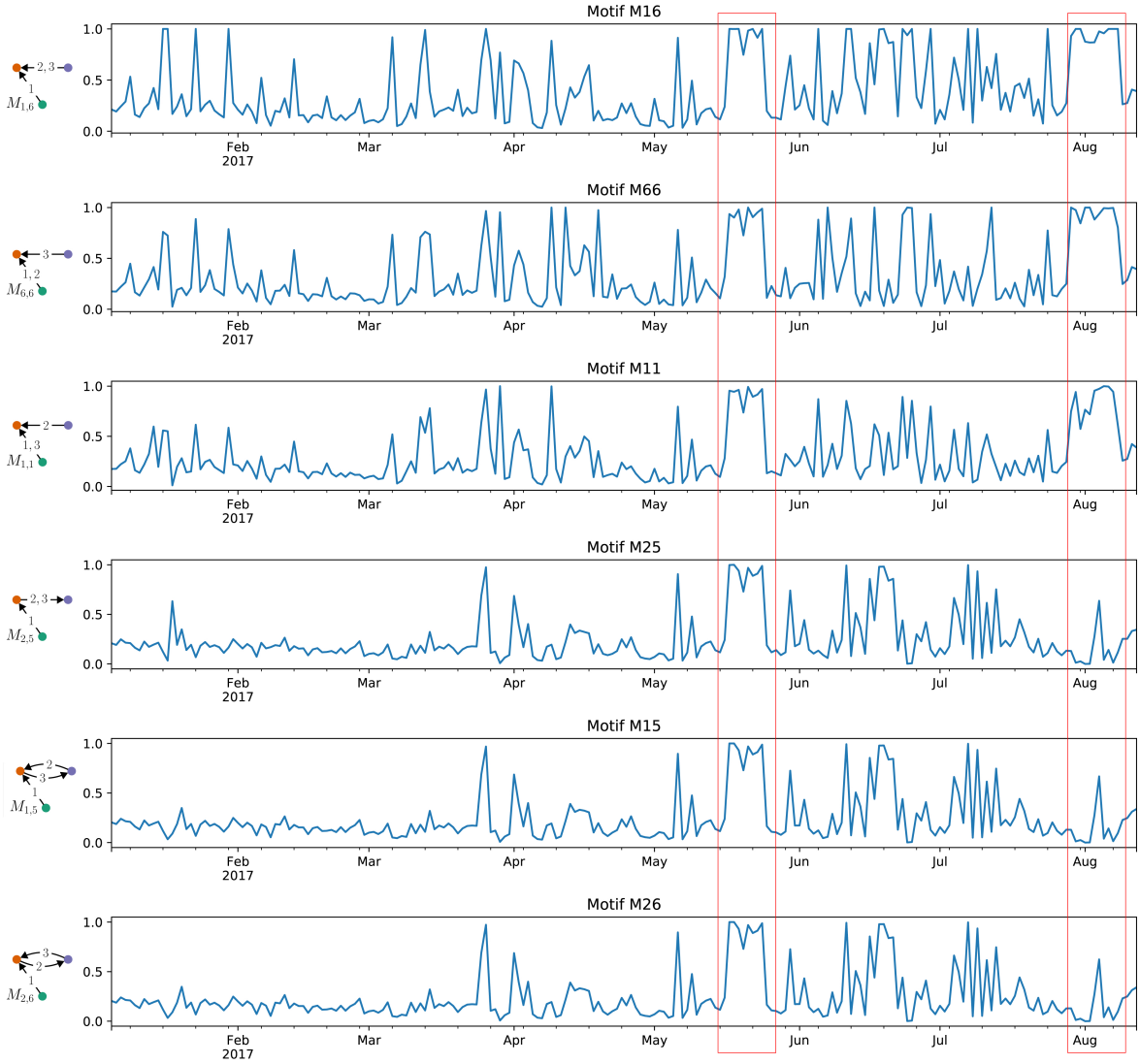


Figure 2.10: Time series from all dates chart of the 6 highest variance motifs: $M_{1,6}$,$M_{6,6}$, $M_{1,1}$, $M_{2,5}$, $M_{1,5}$ and $M_{2,6}$ counts over the Bitcoin transaction temporal network. The red lines mark the exactly dates with meaningful changes on the Bitcoin Protocol: 22 May 2017 the Bitcoin Implementation Proposal Number 91 activation allowing the Bitcoin miners to agree or not to the new changes, and 1 August 2017 Bitcoin Cash creation [1, 2, 3, 4, 5], where the miners did not agree on the new changes proposed on 22 May 2017.

3 Modelling the Bitcoin Transaction Temporal Network

In the previous section, we identify the hidden patterns in our temporal network sample. We continue our work by investigating the possibility of modeling the Bitcoin transactions temporal network. In this section, we propose an analytical model that aims to identify statistical regularities of the distribution of the motifs counts (see sec[motif]) from our Bitcoin transactions temporal network sample (see Section 2.1). We achieve this by modifying the activity driven model proposed by [Per+12].

3.1 Activity driven modeling of time varying networks

The vast majority of recent modeling approaches are connectivity driven [New10; BBV08; AB02; Boc+06; Bol13; Ves12; MR95; HL81; FS86; WP96; BAJ99; BA99b; DMS00; DM13; FFM06; BP03], where the network's topology is at the core of the model's algorithmic definition. Connectivity-driven network models are well-suited for capturing the essential features of systems such as the Internet, where connections among nodes are long-lived elements [PV07; AJB99]. However, in many cases, the interactions between the elements of the system are rapidly changing and are characterized by processes whose timing and duration are defined on a very short time scale [HS12; GH06]. Connectivity driven models necessarily provide a time-aggregated representation that may fail to describe the instantaneous and fluctuating dynamics of many networks, including the network which we are dealing with. The motivation of using an activity driven model as the base of our model is defined by the capability of the model proposed by [Per+12] to encode the instantaneous time description of the network dynamics. The work of [Per+12] introduces a framework which can analytically describe highly dynamical networks. Their models intend to capture the process of accumulating connections over time and the resulting degree distribution and other topological properties merely represent a time-integrated perspective of the system. They conclude that the dynamics of the network can be encoded by the activity potential distribution function from which it is possible to derive the appropriate interaction rate among nodes. Proposing a process model for the generation of random dynamic networks can be used as a general base model for rapidly evolving networks.

3.1.1 The Activity Potential

The activity potential characterizes the individual activity of every agent, in our case, the number of Bitcoin transactions made. Activity potential x_i of the agent i is the number of interactions that he performs in a given particular time window of length Δt, divided by the total number of connections made by all agents during that same time window. It estimates the probability that the agent i was involved in any given interaction in the system. Interaction dynamics of the system is defined statistically by the probability distribution $F(x)$ that a randomly chosen agent i has an activity potential x.

3.1.2 General Activity driven network model

We describe the general activity driven model for time-varying networks proposed by [Per+12]. The general model uses the activity distribution to drive the formation of a dynamic system.

For each node (agent) i we assign an activity firing rate $a_i = \eta x_i$, where x_i are bound by the interval $\mathcal{E} \leq x_i \leq 1$. The activity firing rate is the probability per unit time to create new edges with other nodes, which is assigned according to a given probability distribution $F(x)$ that may be chosen arbitrarily or given by empirical data. η is a rescaling factor. The work of [Per+12] proposes a simple generative process according to the following rules:

1. At each discrete time step t the network G_t starts with N disconnected nodes.

2. With probability $a_i \Delta t$ each node i becomes active and generates m links that are connected to m other randomly selected nodes. Non-active nodes can still receive connections from other active nodes.

3. At the next time step $t + \Delta t$, all the edges in the network G_t are deleted.

Figure 3.1 illustrates the process of the general model from [Per+12]. Considering an example of 13 nodes and $m = 3$. We plot the resulting networks for three different time steps. The red nodes represent the active nodes and the grey nodes represent the inactive nodes. The dashed lines represent the past connections. From this definition we can see that all interactions have a constant duration $T_i = \Delta t$. The agents do not have a memory of the previous time steps. The full dynamics of the network and its resulting structure is therefore entirely encoded in the activity potential distribution function $F(x)$. This function allows the definition of a simple dynamical process based on the nodes' activity rate, providing a time dependent description of the network's connectivity pattern.

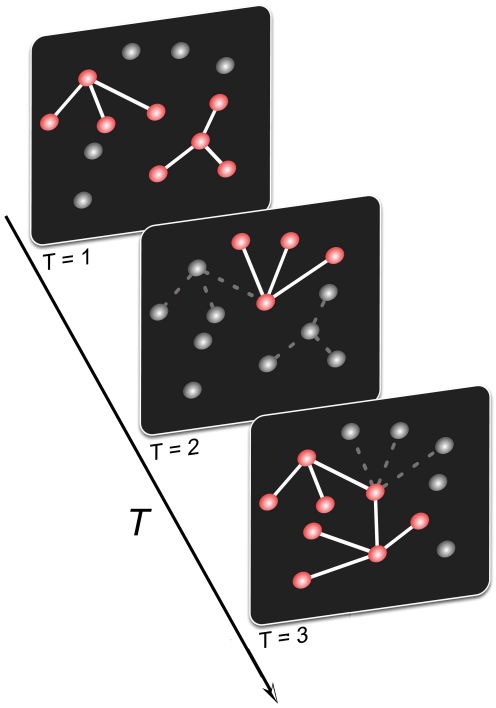


Figure 3.1: Illustration of the general model process from 3.1.3. Considering 13 nodes and $m = 3$, we visualize the resulting networks for three different time steps. The red nodes represent the active nodes and the grey nodes represent the not active nodes. The dash lines represent the past connections.

3.1.3 Applying the General Activity Driven Model to simulate the Bitcoin transactions temporal network

Despite its simplicity, the model stresses that the actors' activity rate plays a significant role in the understanding of dynamical networks. With this feature, we see the potential of this model for simulating the Bitcoin transactions temporal network where, only the agents, the ones who make transactions, define the network. We now investigate how suitable this general model is for capturing the dynamics of our network over our data sample.

In Section 2.4, we selected three sample days to be our golden models: *Day A) May 4 2017; Day B) June 16 2017; Day C) August 1 2017;* Where, considering only the 6 most relevant motifs: $M_{1,6}$,$M_{6,6}$, $M_{1,1}$, $M_{2,5}$, $M_{1,5}$ and $M_{2,6}$. Day *A)* has a lower motifs counts, day *B)* has a higher motifs counts and day *C)* has high volatility, marked by the Bitcoin break of consensus [1, 2, 3, 4, 5]. Our goal now is to use the general model described in 3.1.3 to simulate one day of Bitcoin transactions. We chose this model because we are interested in discovering how the edges are made and not how the network grows. Thus, the model proposed by [Per+12] provides for us a ground zero model for generating random edges. Afterwards, we run the algorithm from 2.3.2 to count the temporal motifs with $\delta = 1$ hour as the parameter. We count the motifs on the simulated temporal network output and on the selected samples days. Then, we normalize the counts to compare the distribution of the motifs counts. We compare them using the root-mean-square error function:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(P_i - O_i)^2} \tag{3.1}$$

An error close to one means the simulation did not capture the dynamics. We run our first experiment of with no modifications on the general model of Section 3.1.3, moreover with the initial parameters: Number of nodes $N = 100$; Δt= 1, Number of Connections $m = 10$, Activity Distribution Function $F(x)$ = Pareto Distribution with $\alpha = 2$, 5000 steps and Temporal Motif $\delta = 1$ hour.

Our first simulation comparing to day A) was able to achieve error $E = 0.583254$. Comparing to day B) $E = 0.389751$ and comparing to day C) $E = 0.642245$. Although we achieved a good distribution, the motifs $M_{2,4}$(triangle), $M_{3,5}$(triangle), $M_{5,1}$(cycle-2-nodes), $M_{5,2}$(cycle-2-nodes), $M_{6,2}$(cycle-2-nodes) were not generated by the simulations. However, those motifs are quite irrelevant in the real data. Figure 3.2 helps to visualize the distribution of the motifs between the selected days and the simulation.

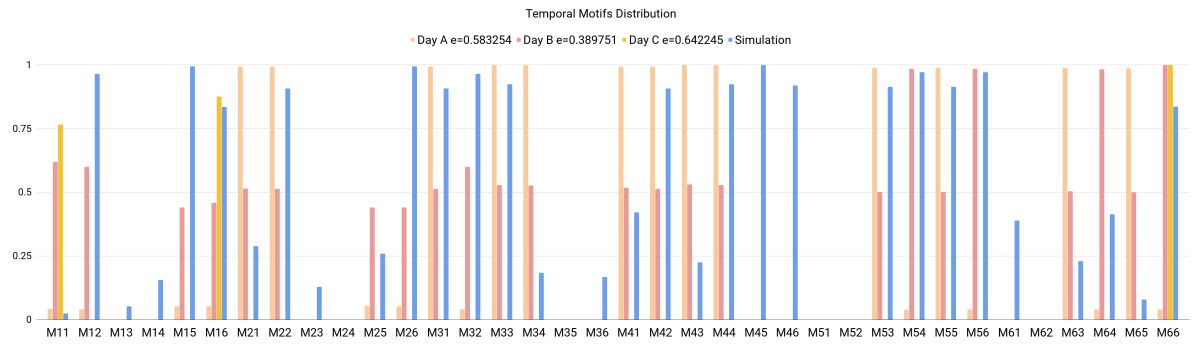


Figure 3.2: Distribution of the motifs normalized between the selected days and the activity driven simulation, using the parameters: Number of nodes $N = 100$; Δt= 1, Number of Connections $m = 10$, Activity Distribution as Pareto with $\alpha = 2$, 5000 steps and Temporal Motif $\delta = 1$ hour. Comparing the simulation to day A) the simulation was able to achieve error $E = 0.583254$. Comparing to day B) $E = 0.389751$ and comparing to day C) $E = 0.642245$. Additionally, the motifs $M_{2,4}$(triangle), $M_{3,5}$(triangle), $M_{5,1}$(cycle-2-nodes), $M_{5,2}$(cycle-2-nodes), $M_{6,2}$(cycle-2-nodes) were never generated by the simulation.

Several attempts with different parameters were made to get better results but none was relevant. We know that the General Model is a *Markovian* model, where the future states depend only on the current state. Although, in order to model the Bitcoin dynamics, we intuitively assume: agents that have traded in the past are more likely to trade again among them. Thus, this assumption adds memory to our agents. Additionally, memory feature is not covered by the general model of subsection 3.1.3. This lack of memory encourages us to modify the general activity driven model by allowing memory on the agents in order to look for better results.

3.1.4 Modifing the Activity driven network model

We pointed out the lack of memory on the general model proposed by [Per+12] in Section 3.1. In this section we investigate a modification of the general model to support memory. We expect this change generates better motif distributions. We start by modifying the general activity driven model by adding a memory queue with fixed size S, following the *First-In-First-Out* pattern on every node. Every node will have this queue which keeps track of who sent a transaction to him. In other words, the agents hold a record of from who they received Bitcoins. Figure 3.3 illustrates a temporal network with a memory queue of size $S = 3$ on every node. The memory of Node 3 holds Node 1 at the head of the queue because the transaction T1 came first. Node 4 stores in its memory Node 3 at the head because the transaction T3 arrived early and Node 2 at the tail because the transaction T5 came later.

Our model has two phases at each time step where edges are generated. The first phase is the activity driven edge generation described in the previous section, and the second phase is edge generation from the memory pool. During this phase, every node

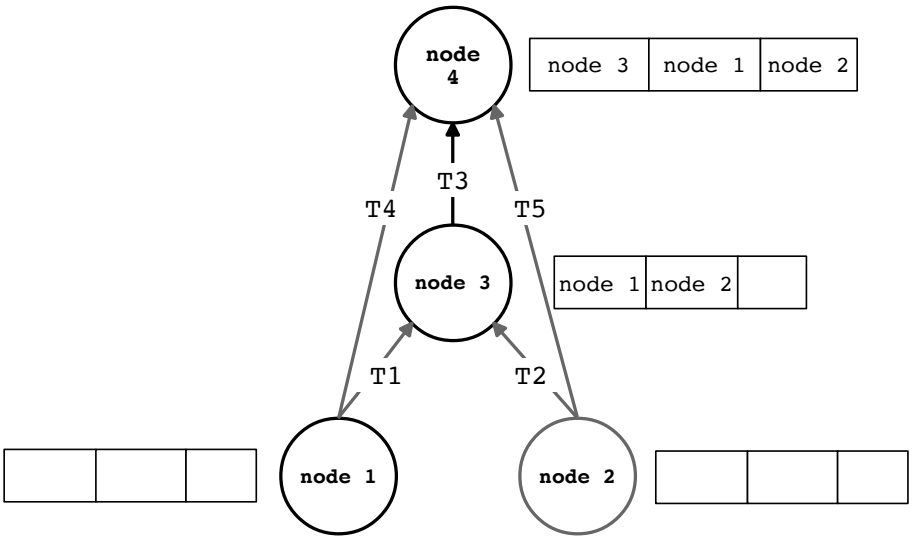


Figure 3.3: Example of a temporal network with a memory queue on every node. In this figure, the memory queue has size $S = 3$ and follows the First-In-First-Out pattern. Every node will have a queue which keeps track of receiver nodes. In other words, the agents hold a record of from who they got Bitcoins. The memory of Node 3 holds Node 1 at the head of the queue because the transaction T1 came first. Node 4 store in its memory Node 3 at the head because the transaction T3 arrived early and Node 2 at the tail because the transaction T5 came later.

with a non-empty memory pool generates an edge to each of the nodes stored in its memory pool. This simulates persistance. Additionally, we set the current runtime timestamp in seconds as time value of the generated temporal edges. For simplicity, we will call this modified model "model A".

3.1.5 Attempts to simulate the Bitcoin transactions temporal network

We propose an extension to the general activity driven model in order to simulate the Bitcoin transactions temporal network. In this extension only the agents, the ones who make connections, define the network by following basic intuition agent rules. In this section we investigate how suitable this model is for capturing the dynamics of our network over our data sample.

For all of the above simulations we used the same initial parameters from the above subsection 3.1.3: Number of nodes $N = 100$; Δt= 1, Number of Connections $m = 10$, Activity Distribution Function $F(x)$ = Pareto Distribution with $\alpha = 2$, 5000 steps, Temporal Motif $\delta = 1$ hour.

In model A, we simulated the network with a small memory size (size = 5 nodes) and followed the same methodology described in the previous subsection. Also, we simulated the same model but with a big memory size (size = 50 nodes).

By adding a small memory to our model we were able to achieve a better error rate than before. Compared to the day A) an error of $E = 0.53157$, $E = 0.440523$ for day B) and $E = 0.465206$ for day C). Compared to the Activity Driven model, we reduced the error in 8.86% for day A and 27.56% for day C. However, for day B our model increased the error by 13%.

For the big memory case, we were able to achieve compared to the day A) an error of $E = 0.512707$, $E = 0.368788$ for the day B) and $E = 0.536415$ for the day C). The big memory model reproduced slight lower error results than the small memory on the day A and B. Additionally, the big memory model compared to the Activity Driven model, decreased the error by 12% for day A and 5.37% for day C and by 16.47%.

However, that modification was not satisfactory; we thus create a new type of memory for our model: at every time step, for every node with memory, we force those nodes to connect with nodes on their memory besides the most recently added to the memory node. Our motivation with this type of memory would avoid on the same time step a node return the just received Bitcoins to its previous owner. We named this modification as model B. We simulated model B following the same procedure described above for model A. We simulate model B with small memory (size = 5 nodes) and with big memory (size = 50 nodes) as well.

For model B small memory, we were able to achieve compared to the day A) an error of $E = 0.573973$, $E = 0.42058$ for day B) and $E = 0.390031$ for day C). From those results, only at day C a achieved significantly improved results. For the case of model B with big memory, the simulation produced similar results as model A with a big memory.

Even though we made two modification to the model, those adjustments were not able

Model	Day A	Day B	Day C
Activity Driven	0.583254	0.389751	0.642245
Model A memory size=5	0.53157	0.440523	0.465206
Model A memory size=50	0.512707	0.368788	0.536415
Model B memory size=5	0.573973	0.42058	0.390031
Model B memory size=50	0.519995	0.365183	0.421879
Model C memory size=5	0.64976	0.533083	0.51391
Model C memory size=50	0.596023	0.398518	0.585966
Model D memory size=5	0.64987	0.549157	0.417396
Model D memory size=50	0.540905	0.315639	0.575851

Table 3.1: All the computed error results for the Models A, B, C, D with memory size=5 and size=50 and the Additionally Activity Driven model, all compared to the selected days. Model A with memory size=50 as the best model for day A. Model D with memory size=50 as the best for the day B. Finally, model B with memory size=5 as the best model for day C. Besides, the models with just memory(model A and B), were able to overcome the Activity Driven model 12% on average.

to produce satisfactory results. Thus, we investigate another modification to the model. We modified again our model based on the intuition that new agents are added to the network at every time step. Thus, we propose two more models: model C) which copies the mechanism of model A plus adds 1 new node to the network at every time step, and model D) which reproduces the model B and adds 1 new node to the system at every time step.

Following the same procedure of model A and B, we simulate model C and D with small and big memory. Unfortunately, model C was not capable of improving over model A nor model B and neither Activity Driven model. However, model D was able to overcome model A, model B and Activity Driven model on day C. We provide in table 3.1, all the computed error results for the Models A, B, C, D with small and big memory and additionally the Activity Driven model, all compared to the selected days for better understanding.

We conclude, from several attempts of simulations, we end up with model A with big memory as the best model for day A. Model D with big memory as the best for the day B. Finally, model B with small memory as the best model for day C. Besides, the models with just memory (model A and B), were able to overcome the Activity Driven model 12% on average. All the simulations were computed on a Macbook Pro with processor 2,7 GHz Intel Core i7 8 cores and with 16 GB 2133 MHz DDR3 memory ram and took on average 40 minutes to be generated. All the simulation data and source code is hosted at https://github.com/ernaneluis/master-thesis.

4 Learning the Parameters

We concluded from the previous section that only analytically generating a temporal network was not enough to adequately model our Bitcoin temporal network samples. Consequently, this result motivates us to investigate the possibility of learning how the edges are created by looking at the real data. This layer of machine learning into our model could infer the correct parameters for the simulation. We propose in this section a strategy towards learning the probability of one node u_i being connected to another v_i node from our Bitcoin temporal network data.

4.1 Bigclam

We start by selecting the work from Yang and Leskovec "Overlapping community detection at scale: a nonnegative matrix factorization approach"[YL13] as our learning algorithm. We believe this is the best state of the art algorithm for Community Detection on large-scale networks; it is a relatively new algorithm to be proven be quite efficient. However, we are not interested in the detection of communities but rather the weights of affiliations outputted by this algorithm. The work proposed by Yang and Leskovec[YL13] presents the Cluster Affiliation Model for Big Networks (BIGCLAM), a probabilistic generative model for graphs that presumably captures the organization of networks based on community affiliations. The proposed model has three main components:

1. The first part acknowledges that communities emerge due to shared group associations.

2. The second ingredient arises from the fact that people tend to be involved in communities through various degrees. Therefore, they assume that each affiliation edge has a non-negative weight. The higher the node's weight of the affiliation to the community the more likely is the node to be connected to other members of the same community.

3. The last ingredient of the Bigclam model is based on the fact that, when people share multiple community connections (e.g., students who attended the same class), the links between them derive from the predominant reason that they shared a community. That means that for each community a pair of nodes shares, they get an independent chance of being connected to each other. Thus, naturally, the more communities a pair of nodes shares, the higher the probability of them being connected to each other.

The model uses a bipartite graph to keep track of the relation nodes and communities, where the nodes at the bottom of the graph represent the nodes of a network G, the nodes on the top represent communities C, and the edges M indicate the relation of node affiliates to a community. They express this bipartite affiliation network as $B(V, C, M)$.

Therefore, the algorithm is able to generate a network $G(V, E)$ given a bipartite community affiliation $B(V, C, M)$. The model considers a simple parameterization where they assign a non-negative weight F_{uc} between node $u \in V$ and community $c \in C$. Where $F_{uc} = 0$ means no affiliation. Given F, they assume that each community c connects its member nodes depending on the value of F. Formally, each community c connects independently its member nodes u, v with probability

$$1 - e^{-F_{uc}F_{vc}} \tag{4.1}$$

Thus, the edge probability between u and v is

$$1 - e^{-\sum_c F_{uc}F_{vc}} \tag{4.2}$$

Where the higher the number of shared communities the higher the probability. Summarizing, given F, a non-negative matrix where F_{uc} is a weight between node $u \in V$ and community $c \in C$, the BIGCLAM algorithm generates a graph $G(V, E)$ by creating an edge between a pair of nodes $u, v \in V$ with the probability 4.2. Note that by using this process, nodes that share multiple communities memberships receive many chances to create a link.

Additionally, to allow edges between nodes that do not share any community affiliations, the model assumes an additional community, called the ε-community, which connects any pair of nodes with a very small probability ε, where from their experiments they set $\varepsilon \approx 10^{-8}$.

Now that we defined the BIGCLAM model, we explain how to learn the affiliation weights between nodes using the model. Given an unlabelled static undirected network $G(V, E)$, we aim to detect $K = 1$ community by fitting the BIGCLAM.

The algorithm keeps generating G' until G' fits G, by modifying the F matrix. In order words, it will look for the most likely affiliation factor matrix $\hat{F} \in R^{N \times K}$ to the network G. This is done by maximizing the likelihood $l(F) = logP(G|F)$:

$$\hat{F} = \underset{F \geq 0}{\operatorname{argmax}}\, l(F) \tag{4.3}$$

Thus, BIGCLAM can learn $F \in R^{N \times K}$ that best approximates the adjacency matrix A of a given static network G. For more details on How Yang and Leskovec solved the optimization problem of equation 4.3 see [YL13].

4.2 Computing Bigclam

Although the BIGCLAM algorithm output is the set of nodes belonging to a community, we are interested in the learned $\hat{F}$. Consequentially, we had to slightly modify the official

BIGCLAM implementation provided at Snap-stanford github page in order to output the learned $\hat{F}$.

To understand what the algorithm can provide for us, we run the BIGCLAM with our 3 Bitcoin data selected days as selected in the section 2.4. Additionally, we induce our temporal network day samples into a static network by removing the time from the edges list. Thus, our day samples are now suitable as inputs for the BIGCLAM implementation. The result contains around 30 megabytes of static edges list for the day samples. We run BIGCLAM on those days on a cluster with Xeon E7 v3 36 cores and 128 GB of ram with the K=1 community. The algorithm took on average 20 minutes on each day sample to output its $\hat{F}$.

After that, we run the Kolmogorov–Smirnov test [Mas51] to discover which distribution fits best the $\hat{F}$. Testing on normal, log-normal, Pareto and Gamma distribution, the best fit was the log-normal distribution for all the selected days $\hat{F}$. We have done this procedure for days A, B, and C. We then plot the histogram of the result $\hat{F}$ and the result of the Kolmogorov–Smirnov test for the day A at figure 4.1, for the day B at figure 4.2 and the day C at figure 4.3.

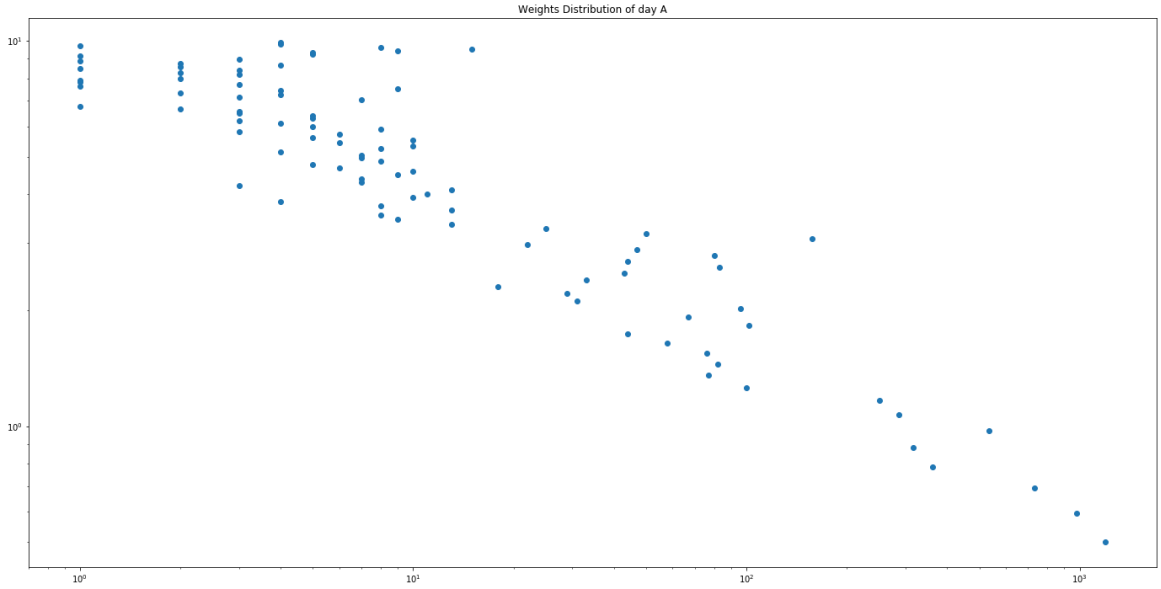


Figure 4.1: Histogram of the result $\hat{F}$ learned by the BIGCLAM algorithm for the **day A** $\hat{F}$. The result of the Kolmogorov–Smirnov test for the **day A** $\hat{F}$ has an error of $E = 0.39$ for log-normal. $E = 0.44$ for normal. $E = 0.79$ for Pareto and $E = 0.98$ for Gamma distribution.

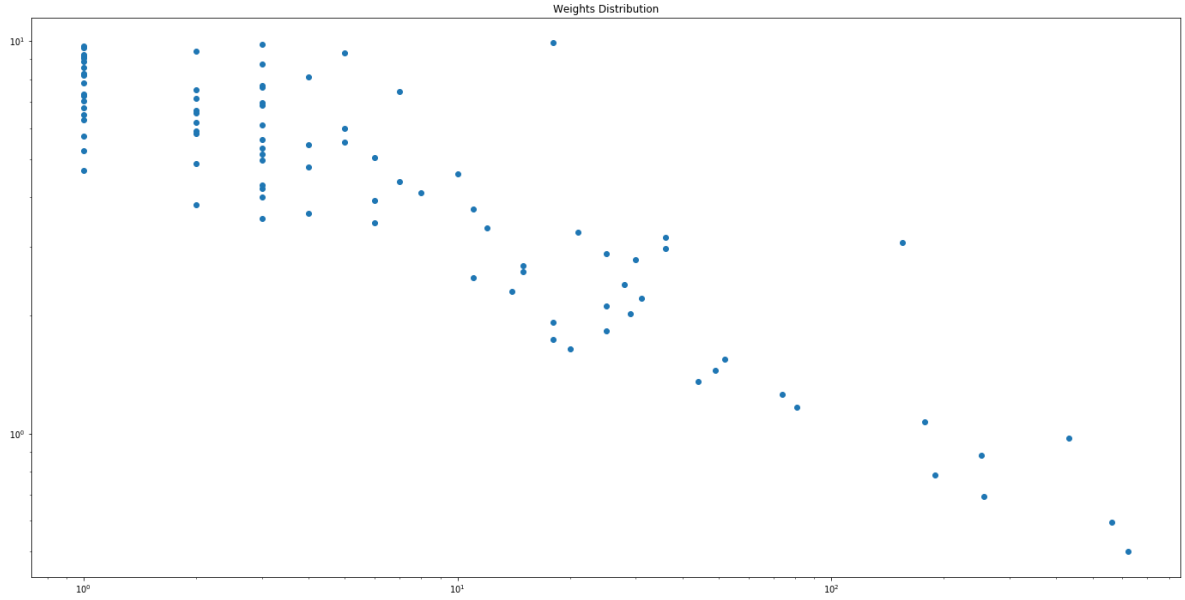


Figure 4.2: Histogram of the result $\hat{F}$ learned by the BIGCLAM algorithm for the **day B** $\hat{F}$. The result of the Kolmogorov–Smirnov test for the **day B** $\hat{F}$ has an error of $E = 0.37$ for log-normal. $E = 0.46$ for normal. $E = 0.77$ for Pareto and $E = 0.98$ for Gamma distribution.

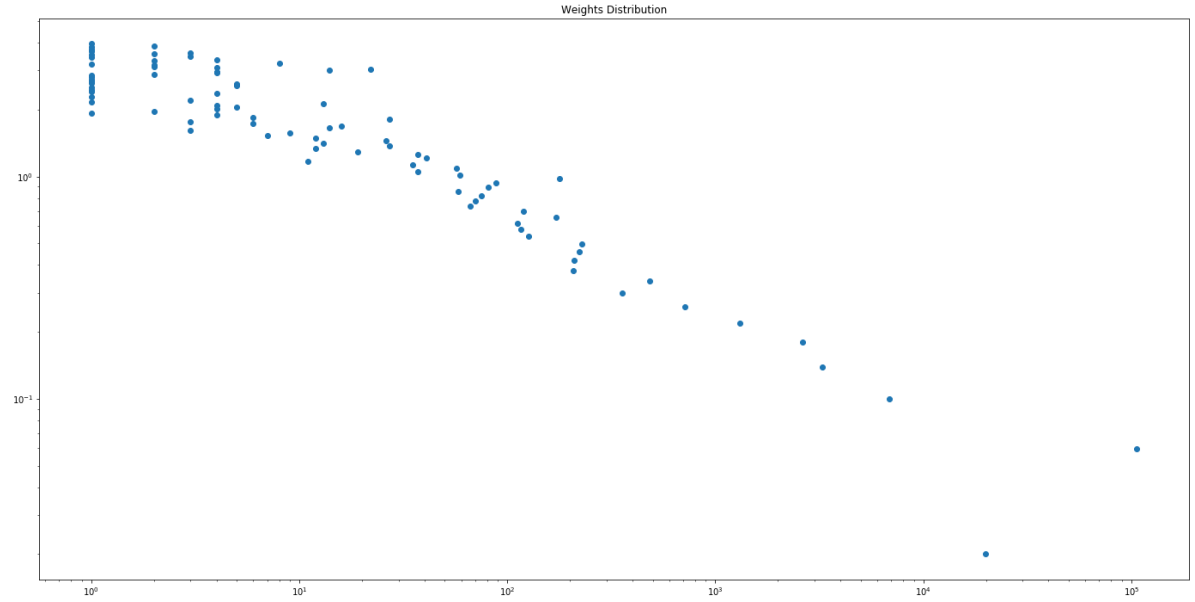


Figure 4.3: Histogram of the result $\hat{F}$ learned by the BIGCLAM algorithm for the **day C** $\hat{F}$. The result of the Kolmogorov–Smirnov test for the **day C** $\hat{F}$ has an error of $E = 0.37$ for log-normal. $E = 0.45$ for normal. $E = 0.77$ for Pareto and $E = 0.96$ for Gamma distribution.

We now do the same above procedure as well, for the model A and Activity driven model produced in Section 3.1.5. The result of $\hat{F}$ from model A memory size=5 can be seen in figure 4.4 and for Activity driven model in figure 4.5. For the simulations, the BIGCLAM result in a log-norm distribution of $\hat{F}$ as well. However, for the model A memory size=5, the test results produced quite close errors for the normal $E = 0.13231$, log-normal $E = 0.13186$ and Gamma $E = 0.13189$. Surprisingly, we can see that the memory model produced different BIGCLAM $\hat{F}$ comparing to the Active Driven model.

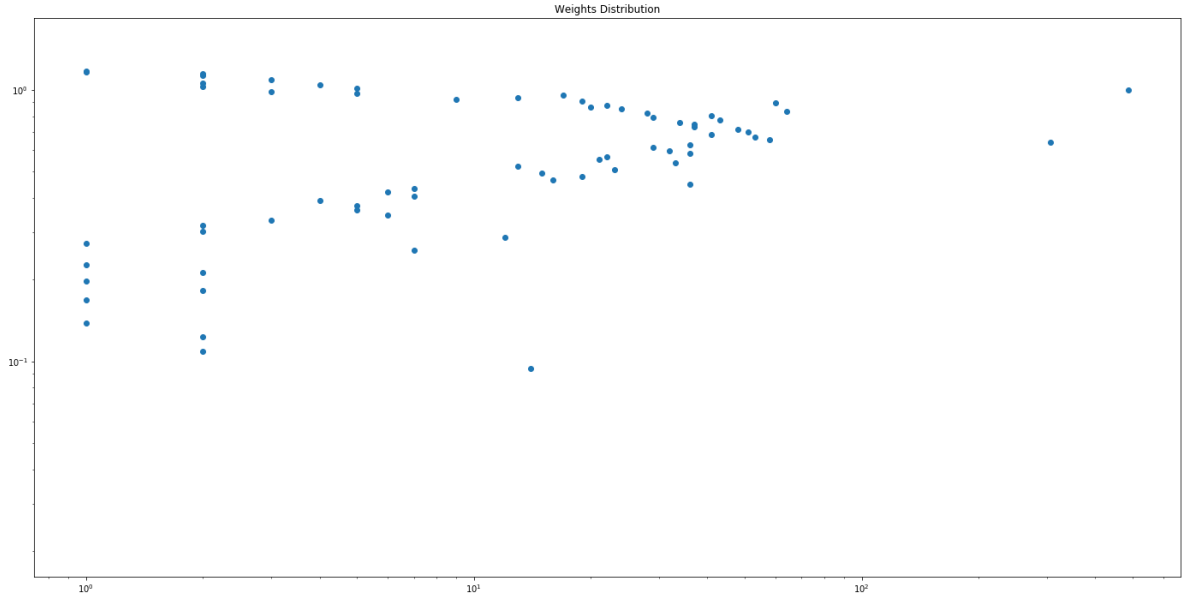


Figure 4.4: Histogram of the result $\hat{F}$ learned by the BIGCLAM algorithm for the **model A memory size=5** $\hat{F}$. The result of the Kolmogorov–Smirnov test for the **model A memory size=5** $\hat{F}$ has an error of $E = 0.13186$ for log-normal. $E = 0.13231$ for normal. $E = 0.38$ for Pareto and $E = 0.13189$ for Gamma distribution. See that the results produced were quite close for log-normal and Gamma.

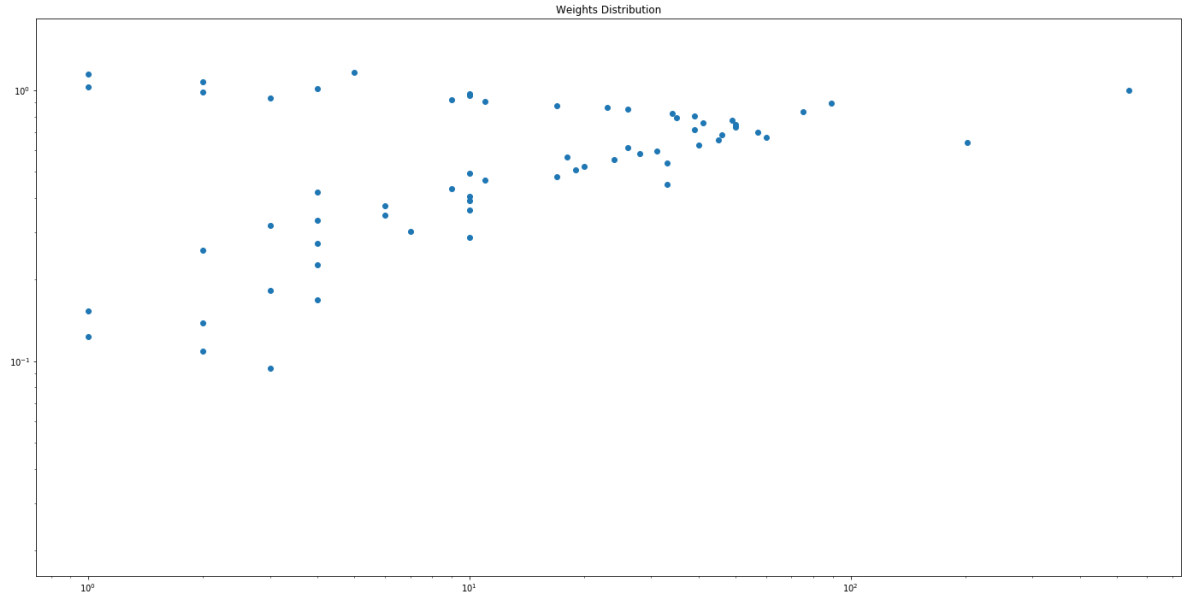


Figure 4.5: Histogram of the result $\hat{F}$ learned by the BIGCLAM algorithm for the **Activity driven model** $\hat{F}$. The result of the Kolmogorov–Smirnov test for the **Activity driven model** $\hat{F}$ has an error of $E = 0.17$ for log-normal. $E = 0.30$ for normal. $E = 0.38$ for Pareto and $E = 0.48$ for Gamma distribution.

We believe that the learned $\hat{F}$ from BIGCLAM can be used to derive the Activity potential distribution of the general model of Section 3.1, which could, in theory, be applied for predicting the Bitcoin temporal network. However, this is not a trivial task. Thus, we push this assignment for future work.

5 Conclusion and Future Work

5.1 Conclusion

In this study, we have analyzed the Temporal Network dynamics of the Bitcoin transactions. We achieved that by using only a subset of the Blockchain to demonstrate a proof of concept. Then, we translate, the not so consistent, Bitcoin transaction data into a Temporal Network. From this Temporal Network, we characterize the dynamics of the agents by identifying 36 types of temporal motifs. This characterization of motifs proves to be very efficient to detect significant changes in the behavior of our Temporal Network. Using this metric, we tried to model the dynamics embedded in the Bitcoin Temporal Network subset. We used a state of the art model as our ground zero to generate networks based on sole a Distribution, capable enough of creating complex dynamics. From our several modeling trials, we generated a slightly improved model, compared to the ground zero model, by adding memory to the agents. However, we were not able to create a convincing model which could capture the whole dynamics of the Bitcoin transactions. For that reason, we pushed our investigation further to catch the mentioned dynamics by using a learning algorithm which, in theory, could derive a better model by learning the probability of edges connected from our data sample. Thus, if our intuition is correct, this would allow us to predict the dynamics of the Bitcoin Transactions Temporal Network.

5.2 Future Work

Given the models and results presented in this work, we propose some directions for future work:

1. Derive a better comprehensive analysis of cause/effects of the temporal motifs related to the parameters of the simulation. We believe that there must exist a relation and this relation could be beneficial when building a distinct model.

2. We suppose that a set of small rules is not enough to capture the whole Bitcoin dynamics. Thus, we believe a mix of generative models could significantly improve the quality of the dynamics.

3. Last but not least, we think it is possible to connect the learned $\hat{F}$, provided by the BIGCLAM algorithm, into our generative model as initial values for the Activity firing rates of the nodes. Despite, this connection is not a trivial task, this could bring a promising ability to predict future dynamics of the live Bitcoin Blockchain.

Bibliography

[Nak08] S. Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008 (cit. on pp. 5, 11).

[PBL17] A. Paranjape, A. R. Benson, and J. Leskovec. "Motifs in temporal networks". In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM. 2017, pp. 601–610 (cit. on pp. 5, 22, 23, 25, 27, 31).

[Per+12] N. Perra, B. Gonçalves, R. Pastor-Satorras, and A. Vespignani. "Activity driven modeling of time varying networks". In: *Scientific reports* 2 (2012) (cit. on pp. 5, 33, 34, 36, 37).

[YL13] J. Yang and J. Leskovec. "Overlapping community detection at scale: a nonnegative matrix factorization approach". In: *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM. 2013, pp. 587–596 (cit. on pp. 5, 41, 42).

[Nak] S. Nakamoto. *I've been working on a new electronic cash system that's fully peer-to-peer, with no trusted third party*. URL: http://article.gmane.org/gmane.comp.encryption.general/12588/ (cit. on p. 11).

[cry] cryptography@metzdowd.com. URL: https://www.%20mail-archive.%20com/search?l=%20cryptography@metzdowd.com%20&q=from:%22Satoshi+%20Nakamoto%22 (cit. on p. 11).

[bloa] blockexplorer.com. URL: http://web.%20archive.%20org/web/%20201310%2015154613/%20http://blockexplorer.com%20:80/block/%200000%20000000%2019d6689c085%20ae165831e93%204ff763ae4%206a2a6c17%202b3f1b60%20a8ce26f (cit. on p. 11).

[mai] mail-archive.com. URL: http://web.%20archive.org%20/web/20140326174921/%20http://www.%20mail-archive.%20com/cryptography%20@metzdowd.com/%20msg10142.html (cit. on p. 11).

[sou] sourceforge.net. URL: http://web.%20archive.org%20/web/20130316013625/%20http://sourceforge.net%20:80/news/?%20group_id=244765 (cit. on p. 11).

[tim] times.co.uk. URL: https://www.%20thetimes.%20co.%20uk/%20article/%20chancellor-alistair-darling-%20on-brink-of-second-bailout-%20for-banks-n9l382mn62h (cit. on p. 11).

[blob] blockchain.info. URL: https://blockchain.%20info/tx/%204a5e1e4ba%20ab89f3a325%2018a88%20c31bc87f%20618f76673%20e2cc77ab21%2027b7a%20fdeda33b?%20show_adv=true (cit. on p. 11).

[Rot09] M. N. Rothbard. *Economic depressions: Their cause and cure*. Ludwig von Mises Institute, 2009 (cit. on p. 11).

[Sza08] N. Szabo. "Bit gold". In: *Website/Blog* (2008) (cit. on p. 12).

[Dai98] W. Dai. "b-money, 1998". In: *URL http://www. weidai. com/bmoney. txt* (1998) (cit. on p. 12).

[Cha95] D. Chaum. "An Introduction to ecash". In: *DigiCash, http://www. digicash. com* (1995) (cit. on p. 12).

[Kon+14a] D. Kondor, I. Csabai, J. Szüle, M. Pósfai, and G. Vattay. "Inferring the interplay between network structure and market effects in Bitcoin". In: *New Journal of Physics* 16.12 (2014), p. 125003. URL: http://stacks.iop.org/1367-2630/16/i=12/a=125003 (cit. on p. 20).

[Kon+14b] D. Kondor, M. Pósfai, I. Csabai, and G. Vattay. "Do the Rich Get Richer? An Empirical Analysis of the Bitcoin Transaction Network". In: *PLOS ONE* 9.2 (Feb. 2014), pp. 1–10. DOI: 10.1371/journal.pone.0086197. URL: https://doi.org/10.1371/journal.pone.0086197 (cit. on p. 20).

[HS12] P. Holme and J. Saramäki. "Temporal networks". In: *Physics reports* 519.3 (2012), pp. 97–125 (cit. on pp. 22, 33).

[BGL16] A. R. Benson, D. F. Gleich, and J. Leskovec. "Higher-order organization of complex networks". In: *Science* 353.6295 (2016), pp. 163–166 (cit. on p. 22).

[Mil+02] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. "Network motifs: simple building blocks of complex networks". In: *Science* 298.5594 (2002), pp. 824–827 (cit. on p. 22).

[Wu+10] Y. Wu, C. Zhou, J. Xiao, J. Kurths, and H. J. Schellnhuber. "Evidence for a bimodal distribution in human communication". In: *Proceedings of the national academy of sciences* 107.44 (2010), pp. 18803–18808 (cit. on p. 22).

[BA99a] A.-L. Barabási and R. Albert. "Emergence of scaling in random networks". In: *science* 286.5439 (1999), pp. 509–512 (cit. on p. 22).

[Jac+15] A. Z. Jacobs, S. F. Way, J. Ugander, and A. Clauset. "Assembling thefacebook: Using heterogeneity to understand online social network assembly". In: *Proceedings of the ACM Web Science Conference*. ACM. 2015, p. 18 (cit. on p. 22).

[LKF07] J. Leskovec, J. Kleinberg, and C. Faloutsos. "Graph evolution: Densification and shrinking diameters". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.1 (2007), p. 2 (cit. on p. 22).

[Ara+14] M. Araujo, S. Papadimitriou, S. Günnemann, C. Faloutsos, P. Basu, A. Swami, E. E. Papalexakis, and D. Koutra. "Com2: fast automatic discovery of temporal ('comet') communities". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2014, pp. 271–283 (cit. on p. 22).

[DKA11] D. M. Dunlavy, T. G. Kolda, and E. Acar. "Temporal link prediction using matrix and tensor factorizations". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 5.2 (2011), p. 10 (cit. on p. 22).

[TBK07] C. Tantipathananandh, T. Berger-Wolf, and D. Kempe. "A framework for community identification in dynamic social networks". In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2007, pp. 717–726 (cit. on p. 22).

[Lat08] M. Latapy. "Main-memory triangle computations for very large (sparse (power-law)) graphs". In: *Theoretical Computer Science* 407.1-3 (2008), pp. 458–473 (cit. on p. 26).

[Mil+04] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon. "Superfamilies of evolved and designed networks". In: *Science* 303.5663 (2004), pp. 1538–1542 (cit. on p. 28).

[Vaz+04] A. Vazquez, R. Dobrin, D. Sergi, J.-P. Eckmann, Z. Oltvai, and A.-L. Barabási. "The topological relationship between the large-scale attributes and local interaction patterns of complex networks". In: *Proceedings of the National Academy of Sciences* 101.52 (2004), pp. 17940–17945 (cit. on p. 28).

[Yav+14] Ö. N. Yaveroğlu, N. Malod-Dognin, D. Davis, Z. Levnajic, V. Janjic, R. Karapandza, A. Stojmirovic, and N. Pržulj. "Revealing the hidden language of complex networks". In: *Scientific reports* 4 (2014) (cit. on p. 28).

[New10] M. Newman. *Networks: an introduction*. Oxford university press, 2010 (cit. on p. 33).

[BBV08] A. Barrat, M. Barthelemy, and A. Vespignani. *Dynamical processes on complex networks*. Cambridge university press, 2008 (cit. on p. 33).

[AB02] R. Albert and A.-L. Barabási. "Statistical mechanics of complex networks". In: *Reviews of modern physics* 74.1 (2002), p. 47 (cit. on p. 33).

[Boc+06] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. "Complex networks: Structure and dynamics". In: *Physics reports* 424.4 (2006), pp. 175–308 (cit. on p. 33).

[Bol13] B. Bollobás. *Modern graph theory*. Vol. 184. Springer Science & Business Media, 2013 (cit. on p. 33).

[Ves12] A. Vespignani. "Modelling dynamical processes in complex socio-technical systems". In: *Nature physics* 8.1 (2012), pp. 32–39 (cit. on p. 33).

[MR95] M. Molloy and B. Reed. "A critical point for random graphs with a given degree sequence". In: *Random structures & algorithms* 6.2-3 (1995), pp. 161–180 (cit. on p. 33).

[HL81] P. W. Holland and S. Leinhardt. "An exponential family of probability distributions for directed graphs". In: *Journal of the american Statistical association* 76.373 (1981), pp. 33–50 (cit. on p. 33).

[FS86] O. Frank and D. Strauss. "Markov graphs". In: *Journal of the american Statistical association* 81.395 (1986), pp. 832–842 (cit. on p. 33).

[WP96] S. Wasserman and P. Pattison. "Logit models and logistic regressions for social networks: I. An introduction to Markov graphs andp". In: *Psychometrika* 61.3 (1996), pp. 401–425 (cit. on p. 33).

[BAJ99] A.-L. Barabási, R. Albert, and H. Jeong. "Mean-field theory for scale-free random networks". In: *Physica A: Statistical Mechanics and its Applications* 272.1 (1999), pp. 173–187 (cit. on p. 33).

[BA99b] A.-L. Barabási and R. Albert. "Emergence of scaling in random networks". In: *science* 286.5439 (1999), pp. 509–512 (cit. on p. 33).

[DMS00] S. N. Dorogovtsev, J. F. F. Mendes, and A. N. Samukhin. "Structure of growing networks with preferential linking". In: *Physical review letters* 85.21 (2000), p. 4633 (cit. on p. 33).

[DM13] S. N. Dorogovtsev and J. F. Mendes. *Evolution of networks: From biological nets to the Internet and WWW*. OUP Oxford, 2013 (cit. on p. 33).

[FFM06] S. Fortunato, A. Flammini, and F. Menczer. "Scale-free network growth by ranking". In: *Physical review letters* 96.21 (2006), p. 218701 (cit. on p. 33).

[BP03] M. Boguná and R. Pastor-Satorras. "Class of correlated random networks with hidden variables". In: *Physical Review E* 68.3 (2003), p. 036112 (cit. on p. 33).

[PV07] R. Pastor-Satorras and A. Vespignani. *Evolution and structure of the Internet: A statistical physics approach*. Cambridge University Press, 2007 (cit. on p. 33).

[AJB99] R. Albert, H. Jeong, and A.-L. Barabási. "Internet: Diameter of the world-wide web". In: *nature* 401.6749 (1999), pp. 130–131 (cit. on p. 33).

[GH06] G. Ghoshal and P. Holme. "Attractiveness and activity in internet communities". In: *Physica A: Statistical Mechanics and its Applications* 364 (2006), pp. 603–609 (cit. on p. 33).

[Mas51] F. J. Massey Jr. "The Kolmogorov-Smirnov test for goodness of fit". In: *Journal of the American statistical Association* 46.253 (1951), pp. 68–78 (cit. on p. 43).