Tampere University

Jaakko Järvi

COMMUNITY EVOLUTION IN BITCOIN INVESTOR NETWORKS

Master of Science Thesis
Faculty of Engineering and Natural Sciences
Examiners: Postdoctoral researcher Kęstutis Baltakys
Professor Juho Kanniainen
October 2020

ABSTRACT

Jaakko Järvi: Community Evolution in Bitcoin Investor Networks
Master of Science Thesis
Tampere University
Master's Programme in Industrial Engineering and Management
October 2020

The rise of cryptocurrencies is one of the phenomena characterizing the past decade. What sets cryptocurrencies apart from the traditional ones is that no central party is required for enforcing the transaction rules and the transactions are also publicly available. Meanwhile, network analysis tools have become widely popular for explaining the complex world shaped by social interaction. Even though the Complex Networks approach has been used for inspecting Bitcoin, the most widely adapted cryptocurrency, no prior study investigates the dynamics of investor communities in Bitcoin networks. The existing studies mostly focus on directed Bitcoin transfer networks while behavioural synchronization networks have not been sufficiently addressed.

This thesis sheds a light on the social aspect of Bitcoin by exploring the dynamics of clusters of investors who time their trades similarly. To conduct such a research, we retrieve the public ledger of Bitcoin transactions and extract over 170 million Bitcoin wallets from the anonymous data. A network of active wallets is formed for each month from 2009 until the end of 2019, and two wallets are connected if their trade timing passes a statistical similarity test. Network analysis tools are used for detecting communities in the formed networks, and community evolution analysis is performed by analyzing the community structure of subsequent monthly networks.

Our results show that Bitcoin investor communities are mostly short-lived but some persist for months or even years. We also find out that the long-lived investor communities prefer splitting over merging when it comes to persistence methods. This research not only produces novel information, which is valuable as such, but also lays a solid basis for future studies concerned with the evolution of Bitcoin communities by bringing together best practices of varying disciplines.

Keywords: Bitcoin, Complex Networks, Network Analysis, Community Evolution, Community Detection, Statistically Validated Networks

TIIVISTELMÄ

Jaakko Järvi: Yhteisöjen kehittyminen Bitcoin-sijoittajien verkostoissa
Diplomityö
Tampereen yliopisto
Tuotantotalouden DI-ohjelma
Lokakuu 2020

Kryptovaluuttojen suosion räjähdysmäinen nousu on eräs menneen vuosikymmenen merkittävistä ilmiöistä. Krypto- ja perinteisten valuuttojen merkittävin ero on se, että kryptovaluutoissa ei tarvita pankin kaltaista keskitettyä tahoa varmistamaan transaktion turvallisuutta, ja lisäksi kryptovaluuttojen transaktiotiedot ovat julkisesti saatavilla. Toinen tiedemaailmaa muovaava ilmiö on maailman alati kasvava monimutkaisuus, jonka tutkimiseen ja selittämiseen käytetään yhä useammin verkostoanalyysin työkaluja. Verkostoanalyysin menetelmiä on käytetty myös tutkittaessa kaikkein suosituinta kryptovaluuttaa Bitcoinia, mutta yksikään aiempi tutkimus ei ole perehtynyt Bitcoin-verkoston sijoittajayhteisöjen kehittymiseen. Aiemmat tutkimukset keskittyvät ennen kaikkea suunnattuihin, Bitcoin-siirtojen verkostoihin, kun taas käyttäytymisen synkroniaa mallintavia verkostoja ei ole tutkittu riittävissä määrin.

Tämä diplomityö lisää tietoisuutta Bitcoinin sosiaalisesta ulottuvuudesta tutkimalla sellaisten sijoittajien ryhmiä, jotka ajoittavat kaupankäyntinsä samankaltaisesti. Tutkimus toteutetaan hakemalla Bitcoin-transaktioiden julkinen pääkirja ja tunnistamalla anonyymistä datasta yli 170 miljoonaa sijoittajalompakkoa. Aktiivisista lompakoista rakennetaan verkosto kullekin kuukaudelle vuoden 2009 alusta vuoden 2019 loppuun, ja verkostoissa lompakoiden välille asetetaan linkki, mikäli kaupankäyntiajoitusten samankaltaisuus on tilastollisesti merkittävä. Kunkin kuukauden verkostosta tunnistetaan sijoittajayhteisöjä verkostoanalyysin keinoin, minkä jälkeen yhteisöjen kehittymistä tutkitaan vertailemalla peräkkäisten kuukausien verkostoja.

Tutkimuksen tulokset osoittavat, että pääosa sijoittajayhteisöistä on hyvin lyhytikäisiä, mutta kuitenkin osa säilyy hengissä useita kuukausia, jopa vuosia. Tutkimus myös näyttää, että pitkäikäiset sijoittajayhteisöt suosivat jakautumista ennemmin kuin yhdistymistä selviytymiskeinona. Täysin uuden tiedon tuottamisen lisäksi tämä diplomityö luo perustan tuleville Bitcoin-yhteisöjen tutkimuksille yhdistelemällä eri tieteenhaarojen parhaita käytänteitä.

Avainsanat: Bitcoin, Monimutkaiset verkostot, Verkostoanalyysi, Yhteisöjen kehittyminen, Yhteisöjen tunnistaminen, Tilastollisesti vahvistetut verkostot

PREFACE

It was December 2018. Having barely survived an overloaded semester of work and studies, there I was sitting in an unevenly lighted meeting room, chit-chatting with my employer about the past, the present and the future. As there were only a couple of courses and a Master's thesis between me and my graduation, it was only natural for him to say that I would be young-yet-experienced when graduating. Little did he know...

So here I am, on a different decade and on a different quarter of life, turning in a thesis the completion of which has not always been as certain as death and taxes. Therefore, it is time to express gratitude towards the brilliant people who have made this happen. First, I would like to thank Kęstutis Baltakys, the main supervisor of this thesis, who has shared his expertise throughout the process. What is more, Kęstutis could have made me feel guilty for the slight inconsistency of my commitment and the challenges in time allocation but instead he decided to make the collaboration work. I would also like to thank professor Juho Kanniainen not only for his advice and help, but also for his great courses which led me to this research topic.

Even though having a great pair of supervisors did smoothen the thesis process, the journey involved many more people. I would like to thank my wonderful colleagues at work and at school – and especially the handful of people at the intersection of those two – who have been there to shape the path and share the highs and lows of it. As this is most likely the end of my journey in the Finnish school system, I will also thank my family for guiding me and being supportive through the whole 18-year cruise.

Just as Finnish sports legend Lasse Urpalainen said right after becoming a World Champion, right now it mostly feels empty, the journey is the meaningful part.

Espoo, 28th October 2020

Jaakko Järvi

CONTENTS

1 INTRODUCTION

Cryptocurrencies have become widely popular over the past decade and Bitcoin, launched in 2009, is the first and most important driver of the boom. Cryptocurrencies enforce the transaction rules without a central party verifying each transfer of currency (Narayanan et al. 2016). Such an unconventional system is fascinating from a technological point of view but, what is more, regulation becomes rather challenging. The rise of Bitcoin price from less than $0,10 to almost $20 000 has created a new class of crypto millionaires (Investopedia 2020) but Bitcoin is also the favorite currency for various criminal activities (Brown 2016). The Bitcoin drama simply offers something for everyone's appetite.

Even though technological innovations like Bitcoin shape the world we live in, the world is much more complex than that. The social interaction between humans forms complex social networks, and the social structure affects our behavior and decision-making (Baltakys, Baltakienė et al. 2019; Siikanen et al. 2018). Thus, unsurprisingly, the field of network analysis has developed rapidly in terms of popularity and methodology. What is more, complex networks methods have proved to be highly useful in several domains, including but not limited to economy, communication, biology and many other (Costa et al. 2011; Emmert-Streib et al. 2018). Other studies, for example, highlight the systemic risk in the financial system (Battiston et al. 2012), explain the topological collapse of interbank networks causing the financial crisis of 2008 (Squartini et al. 2013), hint a trial-stage drug to be toxic (Asur et al. 2009), recommend products and services (Baltakiene et al. 2018) and extract clusters of investors with similar trading strategies from the stock market (Baltakienė et al. 2019; Baltakys, Kanniainen et al. 2018; Bohlin and Rosvall 2014; Musciotto et al. 2016).

As complex networks approach has helped researchers obtain novel, meaningful insights on topics of varying nature, the idea of utilizing such methods on Bitcoin data is intriguing. Therefore, several studies have studied the properties of Bitcoin user network from different perspectives. For example, Ron and Shamir (2013) create a Bitcoin user network and compute basic network properties and other statistics from it, whereas Tasca et al. (2018) extract and investigate super clusters — massive groups of addresses belonging to a single party. Vallarano et al. (2020) compare the properties of Bitcoin user network to Bitcoin price and analyze the evolution of basic properties over time.

However, no previous research has investigated the evolution of community structure in Bitcoin investor networks over a period of time. Bohlin and Rosvall (2014), Baltakienė et al. (2019) and Musciotto et al. (2016) have obtained interesting results when carrying out similar studies on stock market investors. This thesis enters the uncharted territory and aims to gain novel information regarding Bitcoin. The research questions are defined as follows:

1. Are there communities of Bitcoin investors who follow a similar trading strategy?
2. If there are investor communities, how do they evolve over time.

To answer the questions in a structured manner, Chapter 2 presents the theory required for understanding the phenomena. Chapter 3 describes how the Bitcoin transaction data is retrieved from the launch of Bitcoin until the end of 2019 and how the raw, anonymous transactions can be transformed into a wallet-level transaction log. Having created the wallet-level transaction log, we can apply the network analysis tools on the data. Chapter 4 explains how trading similarity is evaluated and how the trader communities are detected. In addition, Chapter 4 describes the framework used for characterizing community evolution. Chapter 5 presents the results whereas Chapter 6 concludes the thesis.

2 THEORY AND PHENOMENA

This section aims to explain the techonologies and the phenomena relevant for understanding the content and methodological choices of this research. As this thesis investigates a network formed by the users of cryptocurrency named Bitcoin, the following chapters will give an overview of cryptocurrencies and Blockchain before diving deeper into Bitcoin. The latter part of this section presents the core concepts of complex networks and a few popular techniques for quantifying and analyzing them.

2.1 Cryptocurrencies

There are several suitable approaches for defining and describing cryptocurrencies. This paper provides a brief explanation by comparing cryptocurrencies with traditional fiat currencies which are widely used in today's world.

Currencies are primarily used as medias of exchange (Investopedia 2019). As a fiat currency itself has no value, unlike a silver coin for example, the system relies on common rules that are to be accepted by different parties. The enforcement of such common rules is what sets traditional and cryptocurrencies apart. In contrast to fiat currencies where a central party enforces the rules, cryptocurrencies apply a set of cryptographic techniques to store the rules in the system itself (Narayanan et al. 2016). In other words, cryptocurrencies offer an alternative, secure channel for strangers to exchange services or products for currency without relying on financial institutions (Vigna and Casey 2016).

Even though Bitcoin is the invention that made cryptocurrencies known for the public, it is not the first cryptocurrency nor it claims to be that. In fact, when Nakamoto et al. (2008) published Bitcoin in a white paper, the paper credited several earlier innovations, such as b-money (Dai 1998) and HashCash (Back et al. 2002), and combined the earlier achievements with their own innovations. Thus, Bitcoin is merely an implementation of a cryptocurrency. However, Bitcoin did pave way for other cryptocurrencies as the creators also invented a public-yet-secure data storage protocol, which builds on top of innovations like the Blockchain.

2.2 Blockchain

Blockchain is a technology, or a group of technologies, which can be used to create distributed ledgers (Lewis 2018). The nature of events stored in the ledger may differ but the basic concept remains the same: new events are bundled in a block of data which is then inserted on top of previous data blocks (Antonopoulos 2017). Even though Bitcoin blockchain is only one implementation of blockchain technologies, let us take a look at how it solves the challenges commonly related to public ledgers.

The security of Bitcoin blockchain, or generally any blockchain, relies heavily on cryptography. Each block calculates its identifier by using a cryptographic hash function and, therefore, it is necessary to be familiar with the very basics of hash functions to understand blockchain. Cryptographic hash functions receive a message of varying length as an input and convert it to a fixed-length digest, known as the hash (Preneel 1993). Such a function must be deterministic, computing the hash should be effortless and reversing the computation should be difficult, or at least impractical (Lewis 2018; Preneel 1993). One part of the difficulty of reversing the computation is the design where a slight change to the input message results in a drastically different output digest (Lewis 2018). The principle was originally proposed by Feistel (1973) and it is also known as the avalanche effect. Table 2.1 presents example inputs and outputs for MD5 hash function.

Table 2.1. *Two messages hashed with MD5 hash function. The avalance effect produces greatly different cryptographic hashes even though the messages are almost identical.*

Message	Digest
10 BTC from Alice to Bob	`e3ea6484c72671d69d4aadd677ea2e7c`
50 BTC from Alice to Bob	`6c029eea9185371952af240ce284ac28`

As can be discovered from the table, the two result hashes have barely any similarities, apart from the length, even though the inputs were almost identical. The avalanche effect is one of the reasons why cryptographic hash functions are extremely useful in blockchain – the public ledger would be meaningless if the transaction data could be tampered afterwards. For example, Bitcoin blockchain builds on the earlier work of Haber and Stornetta (1990) who have described how a digital document can be timestamped unforgeably by using hashes. Bitcoin blockchain uses a similar approach: each block links to its predecessor, the parent block, by calculating a hash value of the parent block's content and then storing the hash in its own header (Antonopoulos 2017; Nakamoto et al. 2008). Figure 2.1 visualizes the principle of chaining new blocks on top of existing ones.

Let us imagine a situation in which Bob becomes greedy and tries to modify the ledger. If, for example, Bob modifies the contents of the second block to receive Carla's funds, the block hash – which is the result of applying a hash function on the contents – will not

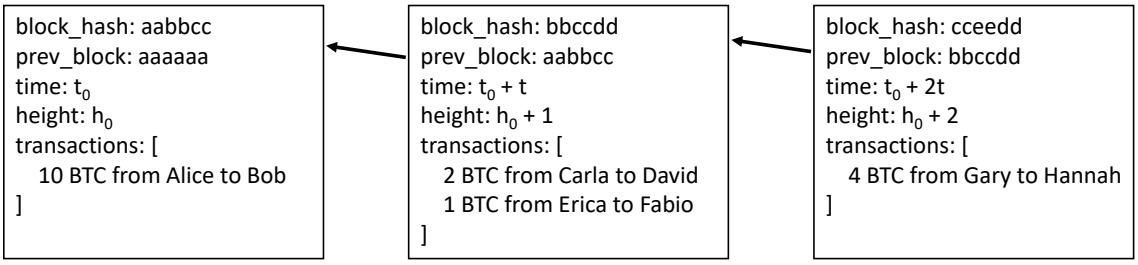


Figure 2.1. *Each blockchain block is linked to its predecessor, and the prev_block reference must match the hash of the parent block.*

remain unchanged either. As a result, the third block's `prev_block` is now pointing to a block that does not exist. Figure 2.2 shows the new, erroneous state.

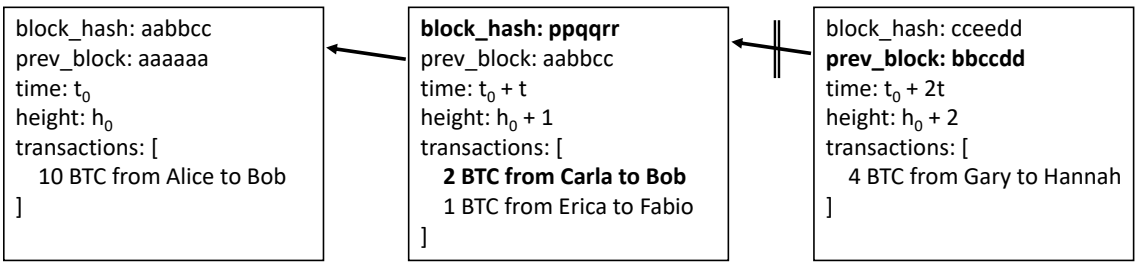


Figure 2.2. *Invalid blockchain due to data tampering*

As the ledger is distributed, meaning it is stored on a remarkable amount of Bitcoin users' computers, Bob's version of the ledger will not be considered the universal truth. To convince the honest users of Bitcoin network, Bob would have to recalculate the hashes for all blocks that have been inserted into blockchain after the block he modified (Nakamoto et al. 2008). As trivial as that may sound, the creation of a new block, which will be explained in more detail in the following sections, requires solving a cryptographic problem which happens to be computationally expensive. The sky-high electricity bill would make it demotivating to rebuild the chain (Antonopoulos 2017) and, what is more, the task of catching and surpassing the chain built by honest users would be next to impossible unless Bob has more computational power than all the honest nodes combined (Nakamoto et al. 2008). Due to the described blockchain feature, it is often stated that each new block reinforces the existing ones (Nakamoto et al. 2008).

Even though the many features of blockchain technologies are most efficiently described by using an actual implementation as an example, the concept of blockchain exists independently of cryptocurrencies. Therefore, distributed blockchain ledgers could and should be used in different domains. For example, proving the authenticity of various documents is necessary to guarantee the security and fairness of international trade. Similarly, global supply chains could benefit from a system where the true origin of a product can be tracked effortlessly. With the described system in place, one would have

no need to worry whether the content of an expensive wine bottle actually matches the etiquette.

2.3 Bitcoin transactions

Bitcoin is a cryptocurrency making use of the blockchain technology. However, there is much more to Bitcoin than the general principles of cryptocurrencies and blockchains. Let us begin by explaining the paradox of having a publicly available yet anonymous transaction history.

Similarly as with traditional currencies, a typical Bitcoin transaction is an event in which some amount of Bitcoin is transferred from one party to another. Contrary to the traditional model where a third party, a bank for example, acts as a middle man to hide sensitive details, Bitcoin provides privacy by using pseudonyms (Nakamoto et al. 2008). In other words, the sender's address, transacted amount and the recipient's address are known but the public are unable to link the data to any real-life identities. Even though the from-alice-to-bob notation was practical while explaining the fundamentals of blockchain, Figure 2.3 presents a more realistic Bitcoin transaction.

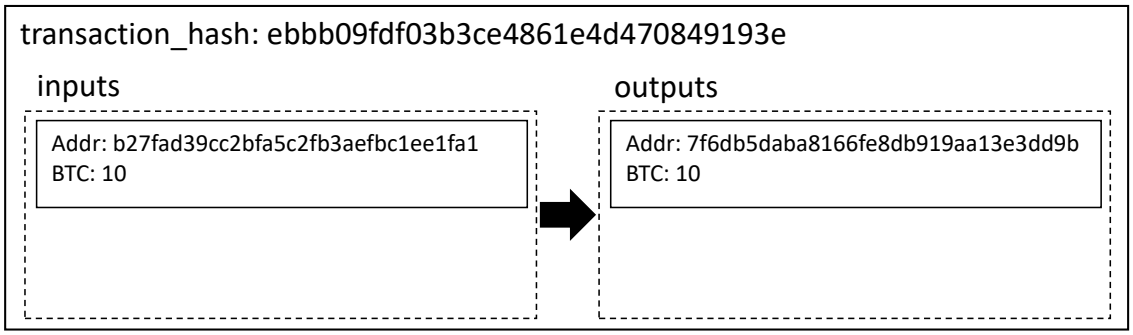


Figure 2.3. *Alice sending 10 Bitcoin from her address to Bob's address*

When the transaction is broadcast to the Bitcoin network – the peer-to-peer network consisting of Bitcoin users – the network nodes will verify that Alice's digital signature permits the spending of Bitcoin which were previously sent to the address starting with `b27fad39`. After a miner node has verified the contents and included the transaction in a new blockchain block, anyone can search the public ledger and see that 10 Bitcoin was transferred from `b27fad39` to another address beginning with `7f6db5da`. However, the public information does not expose Alice and Bob's identities, only their pseudonyms.

Let us imagine that Bob owes Carla 7 Bitcoin and is willing to erase the debt with his recently acquired funds. Bitcoin transactions require the sender to specify a destination address for the whole amount to be sent (Antonopoulos 2017; Nakamoto et al. 2008). In other words, if Bob happened to broadcast a transaction that has Bob's address `7f6db5da` as an input and the output section has only Carla's address receiving

7 Bitcoin, the difference of 3 Bitcoin would be considered transaction fee (Antonopoulos 2017). Simply put, Bob would compensate the block miner with an extremely generous fee. To avoid donating his Bitcoin, Bob has to define an output address which belongs to him and, assuming Bob values security and privacy, he follows the best practice of generating a new address for the change (Nakamoto et al. 2008). Figure 2.4 visualizes the transaction from Bob to Carla where Bob receives 3 Bitcoin as change.

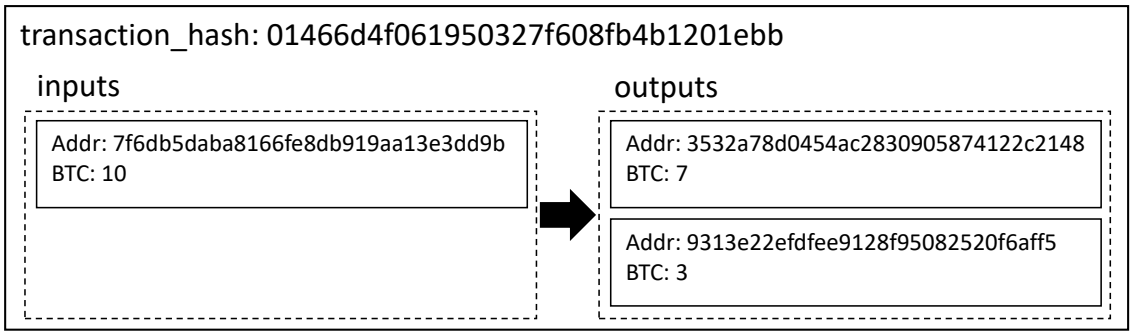


Figure 2.4. *Bob transferring 7 Bitcoin to Carla, receiving 3 Bitcoin as a change*

In addition to receiving change, one can also combine multiple inputs if they do not have sufficient funds in a single address (Nakamoto et al. 2008). If, for example, Carla has received 2 Bitcoin to another address prior to the transaction with Bob, she can send 8 Bitcoin to David in one transaction by combining the inputs as shown in Figure 2.5.

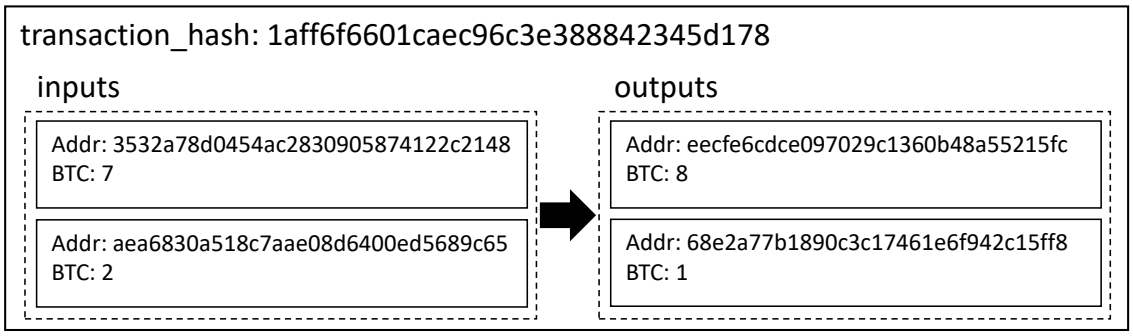


Figure 2.5. *Carla combining inputs to send 8 Bitcoin to David, receiving 1 Bitcoin as a change*

To conclude the transaction logic, each transaction input is an output from an earlier transaction and each transaction can have multiple inputs and multiple outputs. Contrary to a common misconception, Bitcoin is not moved from wallet to another but it actually remains in the blockchain (Antonopoulos 2017). The public can see which addresses hold the unspent transaction outputs but the only one who can spend the currency is the user who has the corresponding private key in their wallet.

2.4 Bitcoin wallets

Bitcoin wallets perform many tasks. On one hand, they are merely a storage for a Bitcoin user's public and private keys (Antonopoulos 2017; Narayanan et al. 2016). As stated earlier, the actual currency is stored in the blockchain and the users are supposed to store their keys which are needed for sending and receiving Bitcoin. On the other hand, from a regular user's point of view, Bitcoin wallets are software applications which provide a user interface for managing one's Bitcoin, keys, and transactions (Antonopoulos 2017; Narayanan et al. 2016). Thus, they hide the majority of the more complex parts of Bitcoin system and allow the users to focus on exchanging Bitcoin.

What is relevant for this research is the relation between users, wallets and addresses. Each user must have a wallet of some sort to participate in transactions. However, nothing prevents a user from having multiple wallets – the situation is actually quite similar to having multiple bank accounts. In such a case, we do not have a single wallet representing the user in the constructed wallet network. Therefore, users and wallets have a one-to-n relationship.

As already stated briefly, the best practice is to generate new addresses instead of reusing the old ones (Antonopoulos 2017; Nakamoto et al. 2008). As a result, it is not only possible but also very likely that a wallet contains multiple addresses and, therefore, there is a one-to-n relationship between wallets and addresses, too. Studying the Bitcoin network on an address level would be of little use considering the topic of this thesis and, hence, we will have to cluster addresses which most likely belong to the same wallet. Section 3.3 presents the methodology for extracting wallets from transaction data.

2.5 Block mining in Bitcoin blockchain

Even though this thesis focuses on Bitcoin transactions, understanding the dynamics of Bitcoin mining is relevant for some of the logic needed to process the transaction data. What is more, it is an essential part of Bitcoin and, thus, a fascinating product of engineering. As Nakamoto et al. (2008) describe in the original Bitcoin white paper, new transactions are broadcast to all nodes and then, following their own preference, each node bundles the transactions into a block and starts working on a cryptographic puzzle which must be solved for a block to be included in the blockchain. That is a plenty to process at a time and, thus, a light should be shed on each of those three steps.

Let us return to Alice and Bob's transaction. As Alice is the sending party, she is the one with the keys needed to sign the transaction. As a result, Alice – or the wallet software Alice is using – is the one to broadcast the transaction to other nodes in the Bitcoin network (Antonopoulos 2017). Alice's wallet is connected to some amount of other active Bitcoin users, just as the concept of peer-to-peer-network suggests, and when these

other nodes receive the transaction, they will verify the content and forward the verified transaction to their neighbour nodes (Antonopoulos 2017; Nakamoto et al. 2008). As all nodes forward the information, Alice's transaction data will spread quickly in the network, even if Alice herself broadcast the information to a small number of nodes. Now Alice has created a valid transaction and the network has been acknowledged, which means that Alice's mandatory tasks are done.

A subset of the Bitcoin network nodes is interested in participating in a computationally expensive task known as block mining. When a miner node is selecting transactions to be included in the block, it has to take into account that there is a limit for the amount of data that can be stored in one block (Antonopoulos 2017). A direct consequence of that is the miners' preference to pick transactions where the ratio of fee, which is the gap between the sum of inputs and the sum of outputs, and the amount of data is relative high. If Alice and Bob's transaction is not lucrative enough, it may not be included in the very next block. Even though the network is conscious of Alice and Bob's willingness to commit a transaction, the transaction will remain unconfirmed until included in a blockchain block (Antonopoulos 2017).

Now that the miner node has selected which transactions it wants to include in the block, the only task remaining is solving the cryptographic puzzle, known as the proof-of-work (Antonopoulos 2017; Nakamoto et al. 2008). The block contents must be hashed so that the result begins with a certain amount of 0 bits (Nakamoto et al. 2008). However, hash functions are deterministic and, thus, the resulting hash is always the same unless the block contents are changed. Hence, a varying component must be introduced to modify the result hash. This variable is simply a number, known as the nonce, and now the task of mining becomes functionally rather trivial: alter the nonce until the proof-of-work condition is met (Antonopoulos 2017; Nakamoto et al. 2008).

The basic idea of mining new blocks can be simulated with a fairly straightforward Python 3 program. To further simplify the concept, let us imagine that the block content to be hashed is merely a simple text `"5 BTC from Alice to Bob 2020-04-24 00:00:00"`. May the proof-of-work condition be that the resulting hash begins with six 0s. By writing a few lines of code, we have a program that runs until it finds a suitable nonce.

Listing 2.1. *The concept of Bitcoin mining simulated with a simple Python 3 program.*

```
1  import hashlib
2
3  message = '5 BTC from Alice to Bob 2020-04-24 00:00:00'
4  nonce = 1
5  nonce_found = False
6
7  while not nonce_found:
8
9      # Concatenate message and nonce, convert into bytes
10     message_as_bytes = (message + str(nonce)).encode()
11
12     # Calculate the hash and switch from bytes
13     # to hexadecimal representation
14     hash_hex = hashlib.md5(message_as_bytes).hexdigest()
15
16     # If the hash meets our "proof-of-work" condition,
17     # we have found a solution
18     if hash_hex[0:6] == '000000':
19         nonce_found = True
20         print('Nonce {}, hash {}'.format(nonce, hash_hex))
21
22     else:
23         nonce += 1
```

Simply put, we will start from number 1 and try nonces one by one until finding a solution. There is no such thing as an optimal starting nonce as the avalanche effect makes it impossible to estimate which nonce would be close to a correct solution. Table 2.2 presents the first few hashes and the first hash which meets our proof-of-work condition. In addition, a few other solutions are shown to emphasize the fact there are multiple solutions, all of which are of equal quality.

The last row of Table 2.2 demonstrates how the difficulty of mining can be adjusted. By requiring the first seven digits to be 0s, it takes our mining algorithm over 730 million attempts to meet the proof-of-work condition. The basic principle of Bitcoin block mining is the same as in our trivial example, though, Bitcoin mining difficulty is adjusted on a bit level whereas our example applied the same principle on a hexadecimal digit level. Thus, the slightest adjustment our hexadecimal example model permits is a change of four bits, as $2^4 = 16$. The Bitcoin network adjusts the difficulty so that a new block is mined approximately every 10 minutes (Antonopoulos 2017), and considering that the electricity consumption of the Bitcoin network was the same magnitude as that of Ireland in 2018 (Economist 2018), the difficulty of finding a solution is absurd. However, when someone finds a solution and broadcasts it to other nodes in the network, it is effortless for the others to verify the correctness of the successful miner's block. By presenting a correct

Table 2.2. *Simulating the concept of Bitcoin mining by hashing a simple message and a varying nonce with MD5.*

Nonce	Hash
1	`128d9b4e32e4f2b4b4552f6f6215fa47`
2	`13f0c1920da3cfe22dab03fd794bedfe`
3	`44d14a0b9e32b4957ed23dfb182a33fe`
...	
6 403 901	`000000f9915a652c58ef99c518a6515d`
40 501 148	`000000acb79a84ec4a23a83eaeba17b8`
69 956 735	`000000b337e3ded717d5917b3c138d95`
...	
732 420 615	`000000048d6125a5c249428b8ae875c6`

solution to the cryptographic problem, the miner has proven their hard computational work, thus the concept is called proof-of-work.

The mining process may also be considered a very specific type of voting system. When the majority of Bitcoin network agrees on something, that is considered the truth. However, as explained in the original Bitcoin paper, allowing each computer or IP address to vote once would give the power to someone who has the ability to generate such addresses (Nakamoto et al. 2008). As anyone can create nodes in the Bitcoin network and, hence, have over 50 % of Bitcoin network nodes in their possession, a more sophisticated and secure protocol is needed. As Nakamoto et al. (2008) explain, Bitcoin solves the complex problem by relying on the proof-of-work: the longest chain of blocks corresponds to the greatest amount of computational work, the greatest number of votes that is. When a node receives information that there exists a new version of the Bitcoin blockchain, it verifies the content and compares its local version with the fresh information. If the new version of Blockchain is the longer one, the node considers it the new truth and shows its acceptance by starting to work on finding a new block to extend the chain.

2.6 Networks & graph theory

A simple network consists of individual points which are connected by lines. What makes networks intriguing is the fact that systems of varying complexity and nature can be considered networks and analyzed accordingly. (Newman 2018) Network science bases on graph theory, which is a branch of mathematics (Barabási et al. 2016). The most basic terms of network science are nodes and links, also known as vertices and edges (Barabási et al. 2016; Newman 2018). Figure 2.6 presents a network where the nodes are computers and the links are connections – be it wired or wireless – between the computers.

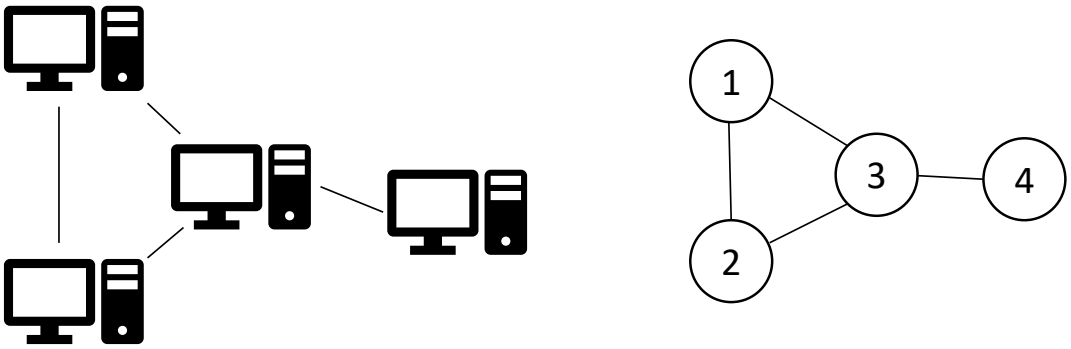


Figure 2.6. *A computer network and its graph representation, edited from Barabási et al. (2016)*

The links of a network may also be directed or weighted (Barabási et al. 2016; Newman 2018), even though the links in Figure 2.6 are neither. For example, a supply chain could be presented as a directed network and, depending on the research topic, weights could be added to represent delivery times, transaction volume or some other property. If we were to construct a Bitcoin user network or a Bitcoin address network, the transactions would be directed links from sender to receiver. However, as this thesis aims to identify similar wallets, the links represent similarity and, thus, are undirected.

Nodes and links are the core of any network but what is of great interest to this research is community detection. Barabási et al. (2016) define community as a group of nodes that have a higher probability of connecting to other nodes inside the group than from outside the group. Generally, the definition states that there are many links between the nodes of the community – or at least they are less connected to other nodes of the network. Figure 2.7 presents a network of two communities. Even though nodes 7 and 8 are linked, the network does not merge into one community but merely has a bridge between two communities.

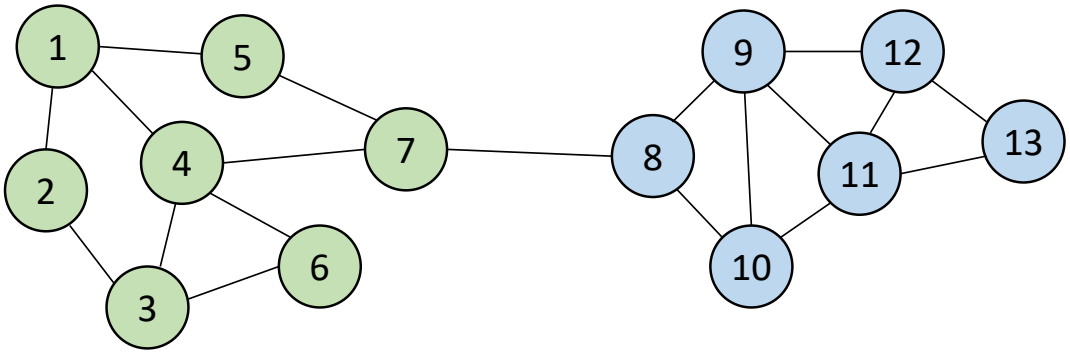


Figure 2.7. *A network with two communities. Even though nodes 7 and 8 are linked, they belong to different communities.*

Even though using common sense is a sufficient community detection algorithm for the

tiny network presented in Figure 2.7, a more sophisticated method is needed for extracting communities from larger, more complex networks. Several algorithms have been proposed to tackle the challenge which, after all, is rather vaguely defined. For example, the Louvain algorithm aims to maximize modularity (Blondel et al. 2008), the DEMON algorithm lets each node vote which community would be the most fitting for its neighbours (Coscia et al. 2012) and the Infomap algorithm uses a random walker to simulate information flow in a network (Rosvall et al. 2009). No algorithm is perfect but most of them are highly useful for discovering groups of similar nodes.

3 DATA

This chapter describes how the Bitcoin blockchain was retrieved and how we verified the retrieved data set. What is more, this chapter also explains the process of extracting wallets from the anonymous Bitcoin transaction log.

3.1 Retrieving the blockchain

This thesis analyzes Bitcoin transactions from the very beginning, which is January 2009, until the end of 2019. Each block within the 11-year timespan is of our interest, as well as each transaction included in those blocks. The last block of the timespan is the one with a height of 610 690 and a hash ending with `fb33aeb93`. In other words, our data consists of the first 610 691 blocks, one of which is the genesis block.

The interval is roughly 96 341 hours long and, as a new block is to be mined every ten minutes, there should be approximately 580 000 blocks. Though, the slight offset is only logical when we take into consideration the fact that the mining difficulty is calibrated every 2 016 blocks, around two weeks, and bases on the actual performance of mining the previous batch of blocks (Antonopoulos 2017). Consequently, if the computational power of network nodes grows constantly, at the latter stages of a 2 016-block batch the miners are expected to find a proof-of-work in slightly under 10 minutes.

As stated several times, Bitcoin is based on a public ledger. However, there is no single correct method for obtaining the ledger. The obvious method is to join the Bitcoin network and receive the whole blockchain from other network nodes. The other approach is to use some of the third-party services on the Internet. This thesis fetched the blockchain data from a RESTful API provided by blockchain.info. We consumed these two end-points:

1. https://blockchain.info/blocks/{timestamp}?format=json
2. https://blockchain.info/rawblock/{block_hash}

The first end-point allows us to get the hash of each block added to the blockchain on a certain day, which is passed in parameter `timestamp`. It should be noted that using this this end-point is not functionally mandatory as we could just fetch the most recent block and then, as each block contains a reference to its parent, fetch the parent block. However, it would be time-consuming and tiresome to query the API block by block. To

solve the issue, the first end-point is used to fetch all block hashes for each day spanning from 2009 until the end of 2019. Listing 3.1 presents a JSON-format response from the first end-point, though, for practical reasons, not all 145 blocks of December 31st 2019 are included.

Listing 3.1. *Daily Bitcoin blocks. JSON-formatted response from RESTful API.*

```
{
    "blocks": [
        {
            "height": 610546,
            "hash": "000000000000000000fa3b65e43e4240d...",
            "time": 1577754335
        },
        ...
        {
            "height": 610690,
            "hash": "000000000000000000005f0cc9b56533624...",
            "time": 1577835692
        }
    ]
}
```

By passing a block hash as an argument, the second end-point gives us the contents of a blockchain block, including transaction details. Even though including full transaction details is of great help, the large block size results in the API responding rather slowly. However, having used the first end-point to push the block hashes to a queue, we can have multiple asynchronous workers using the blockchain.info API and storing the results in a document database. Simply put, a program which mostly waits for an external API and a local database to do their work, is a lightweight-yet-slow one and, considering the amount of data, rather inconvenient. Therefore, it makes sense to work on multiple blocks simultaneously.

At this point, no data is modified – the JSON documents are stored in a local MongoDB instance as such. Overall, this thesis tries to follow the same ideology of performing the time-consuming tasks in a way which prevents the need for re-doing every tasks if an error happens to occur at some of the later stages. In the sense of managing risks, it would be a sub-optimal approach to modify the data at this point. Listing 3.2 contains a JSON-format representation of the last block in our data set.

Listing 3.2. *API response payload: blockchain block in JSON format label*

```
{
    "ver" : 1073733632,
    "next_block" : [ ],
    "time" : 1577835692,
    "bits" : 387300560,
    "fee" : 2255310,
    "nonce" : 281267184,
    "n_tx" : 744,
    "size" : 283571,
    "height" : 610690,
    "hash" : "000000000000000000005f0cc9b565336243d08e65f...",
    "prev_block" : "000000000000000000001292920f327724599f...",
    "mrkl_root" : "d0245fe9ffa995cac9bccf16ecb17028a751e...",
    "tx" : [ ... ]
}
```

The `tx` property holds a list of transactions, each of which is structured as shown in Listing 3.3. Some properties have been omitted to save space and to emphasize the ones which are of greater significance for our needs.

Listing 3.3. *API response payload: JSON representation of a Bitcoin transaction.*

```
{
    "hash" : "1e34fd3391f4c7b55bdbfc4ae...",
    "block_height" : 610690,
    "inputs" : [
        {
            "script" : "473044022070826cf227b3066a6e4a...",
            "prev_out" : {
                "value" : 153042875,
                "addr" : "1BaYzjAbQfvdyu1ZVb2hoiFzonPY4GTH7v"
            }
        }
    ],
    "out" : [
        {
            "value" : 1279990,
            "addr" : "3JprvwhSrR83wmBHjWRDKTstsGDwWzurQZ"
        },
        ...
        {
            "value" : 147540115,
            "addr" : "19rNsMmTXoYcRHfuA7gyMUdMYZ758siNnm"
        }
    ]
}
```

The input, output and fee amounts are expressed as satoshis, which is a denomination of Bitcoin. One Bitcoin consists of 100 000 000 satoshis (Antonopoulos 2017), similarly as a euro is equal to 100 cents.

3.2 Verifying the retrieved data

As a third-party service is used for retrieving the data, there is a clear need for verifying the content. In addition, it is necessary to confirm that our own program functions as desired. To ensure the quality of our data, certain tests may be carried out.

First, as each block references the previous block, it is rather trivial to confirm that the retrieved data set contains the parent block of each retrieved block. Running such a test revealed that there were 14 blocks which the blockchain.info service was unable to return, despite re-trying multiple times. To overcome the shortage, the Bitcoin Core software (Nakamoto et al. 2009) is used to retrieve the missing blocks. Bitcoin Core is a reference implementation of Bitcoin node, originally developed by Satoshi Nakamoto and, since 2011, maintained by the open-source community (Antonopoulos 2017). By running a Bitcoin node with Bitcoin Core, the missing blocks were acquired and, as the data included transaction hashes, a separate blockchain.info end-point was used to fetch the transactions in an identical format to the other other blocks.

Having filled the void of 14 blocks, the parent block hash test passes. The integrity of blockchain is also verified by inspecting the block heights. Each block contains an ordinal number indicating its place in the blockchain. In the retrieved set of 610 691 blocks, the block heights range from 0 to 610 690 with no duplicates, further confirming the assumption that the blockchain has been fetched successfully.

However, as this study focuses on the transaction data, verifying block headers is not sufficient, even though necessary. Thus, the retrieved transaction data is verified by comparing the blockchain.info data to the blockchain data obtained via Bitcoin Core. The test script runs three tests on the contents of each block. First, it is checked that the `n_tx` property has the same value in both sources. Then we ensure that the `tx` property, which holds an array of transactions, has the same length as `n_tx` indicates. Finally, we compare the transaction hashes contained in the two sources: the list of transactions must be the same regardless of the source.

The final test dives one level deeper and compares the addresses that participate in a transaction. Our two data sources have a vastly different approach to presenting transaction inputs and, therefore, we will mostly focus on comparing outputs. The test shows that some more complex transactions have been decoded in a different way in our two data sources, especially around block height 350 000. Therefore, some addresses are missing from our main source, though the amount is relatively low and allows us to go proceed

with the study.

3.3 Extracting wallets from transaction data

As the Bitcoin network relies on the use of pseudonyms, it is not possible to link transactions to real-life identities without off-chain data sources. However, as this thesis studies the transaction network as a phenomenon, it is sufficient to identify which addresses belong to the same user, regardless of the user's actual identity. Several studies have carried out a similar task of mapping Bitcoin addresses to user wallets, commonly known as address clustering (Androulaki et al. 2013; Bovet, Campajola, Lazo et al. 2018; Bovet, Campajola, Mottes et al. 2019; Ermilov et al. 2017; Harrigan and Fretter 2016; Meiklejohn et al. 2013; Ron and Shamir 2013; Tasca et al. 2018; Vallarano et al. 2020).

Each study has a slightly different approach for extracting wallets. Overall, most studies take advantage of these two heuristics, both of which are based on Bitcoin features:

1. Common spending
2. One-time change address

The first behavior pattern, common spending, suggests that whenever multiple input addresses are included in a transaction, all those addresses belong to the same wallet. As a consequence, if addresses A and B are the inputs in one transaction and addresses B and C are inputs in another one, all three addresses are considered to belong to the same user. As the sender is the one whose digital signature is needed for the transaction, having multiple users' inputs in one transaction can be considered rare, or at least non-trivial for an average user.

However, it should be noted that there are third-party mixing services for creating multi-party transactions, due to which a transaction can have multiple different users' inputs (Antonopoulos 2017). After all, the multi-input heuristic is used very often in address-to-wallet mapping and, as Harrigan and Fretter (2016) describe it, the technique is very practical even if not completely accurate. Harrigan and Fretter (2016) also show in their research that utilizing the common spending heuristic does not create a suspicious amount of superclusters, stating that the amount of false positives in wallet merging is fairly limited.

The first heuristic is quite straightforward to implement. The address-to-wallet mapper iterates over the blockchain blocks in chronological order and associates each observed address with a wallet ID. Input addresses always belong to the same wallet and, hence, these will be associated with the same wallet ID. There is no heuristic for output addresses and it would be too bold to assume they all belong to the same user. Hence, each previously unobserved output address is given a new wallet ID from the wallet ID sequence. It

should be noted that even those addresses which are observed together as transaction inputs have previously been observed as transaction outputs – as every input is a former output. Consequently, wallets are merged frequently. Figure 3.1 shows the concept of associating input addresses with the same wallet ID and updating the address-wallet pairs accordingly.

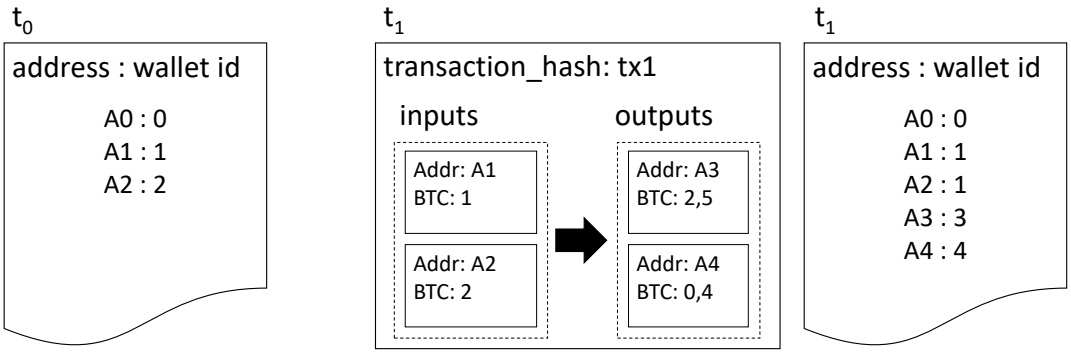


Figure 3.1. *Address clustering heuristic I: all inputs belong to a single wallet.*

In Figure 3.1, addresses A3 and A4 were observed for the very first time and, thus, they were associated with completely new wallet IDs. Having processed the transaction, the wallet mapper continues analyzing transactions one by one. If we happen to observe address A2 again as an input, the other input addresses are associated with wallet 1. Figure 3.2 presents a transaction which results in merging wallets 1 and 4.

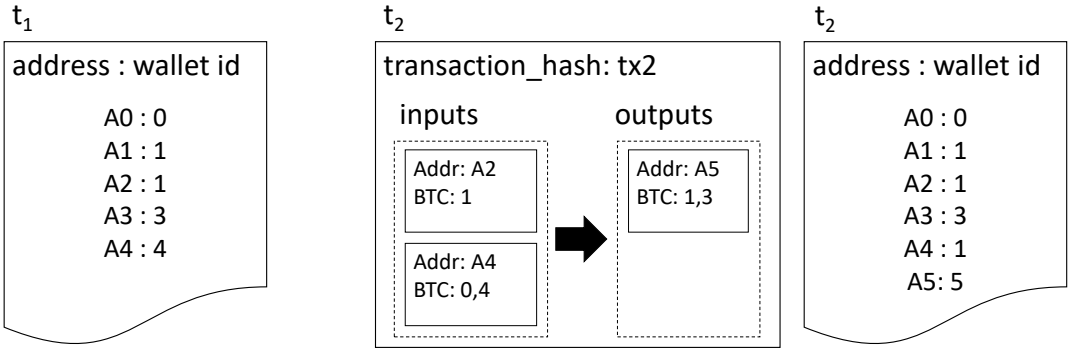


Figure 3.2. *Address clustering heuristic I: address A2 enables further merging.*

Even though heuristic I is powerful, it is not perfect. In addition to false positives, the multi-party inputs discussed earlier, this heuristic also results in false negatives: contents of an actual wallet may be split into several wallets. As it is generally recommended to generate new addresses instead of re-using the existing ones (Antonopoulos 2017; Nakamoto et al. 2008), it is possible that address A2 occurs only once as an input. Consequently, address A4 in Figure 3.2 would be paired with some other input than A2 and, therefore, A4 would not be associated with the same wallet as addresses A1 and A2.

To tackle the limitations of heuristic I, we will introduce heuristic II. Androulaki et al. (2013)

were the first ones to implement such a heuristic and, as the research was carried out at an early stage of Bitcoin, they had a rather simple condition for the change address: if transaction has exactly two outputs and exactly one of them has been observed earlier, the new address is the change address. Meiklejohn et al. (2013) continue the work and propose a more detailed set of rules. For a transaction to have a change address, they require it to have exactly one new output address and none of the input addresses is allowed to be an output address, called a self-change address. They also clarify that the transaction must not be a coin generation one. In their study, Bovet, Campajola, Mottes et al. (2019) consider an output to be a change address if the address is a new one and the amount transferred to it is lower than the smallest input. Guided by these studies, this thesis defines the change address heuristic conditions as follows:

0. The transaction has at least two output addresses
1. This is the first observation of the address
2. The address receives fewer Bitcoin than the smallest input sends
3. Conditions 1 and 2 are met by exactly one address
4. The transaction is not a coin generation
5. The outputs do not contain self-change addresses

Let us walk through the conditions one by one. Condition 0 is a trivial one but rather easy to forget while writing code. However, it must not be forgotten as otherwise most transactions with one input and one output would be classified as pointless transactions transferring Bitcoin inside the wallet.

Condition 1 bases on the idea of Bitcoin wallets automatically generating a new address for receiving the change. Most wallets do not bother the user with the concept of using each address only once – instead they follow the best practice of automatically generating a new address for the change. Even though there are some wallet applications which allow the user to manually define the change address, avoiding false positives is of great importance and motivates us to enforce condition 1.

Condition 2, the address must receive fewer Bitcoin than the smallest input sends, is derived from the concept of transaction fees being based on the amount of data. Therefore, including redundant input addresses in transactions would make little sense. Figure 3.3 visualizes a transaction where the second input address is merely a waste of bytes.

Our condition 3, conditions 1 and 2 must not be met by more than one wallet, has its roots in the rules proposed by Meiklejohn et al. (2013) but has been slightly modified from the original version. Whereas Meiklejohn et al. (2013) require the transaction to have exactly one new input address, this thesis combines the requirement with the idea that a change address must receive a lower value than the smallest input sends. Overall, the aim is

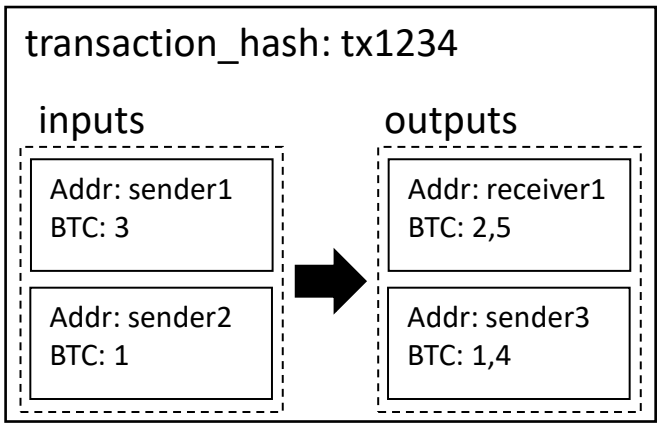


Figure 3.3. *Redundant transaction input. If the sender wants to send 2,5 Bitcoin to the receiver, there is no need to include the second input address. In fact, it increases the size of the transaction and, therefore, raises the transaction fee.*

to reduce false positives by skipping transactions with more than one change address candidate and, when put that way, the condition is actually quite identical with the original one. Figure 3.4 presents a transaction which has two new output addresses but only one of them meets the condition of receiving fewer Bitcoin than the smallest input sends. The re-defined condition 3 allows us to identify a change address even though the version proposed by Meiklejohn et al. (2013) would have skipped this one.

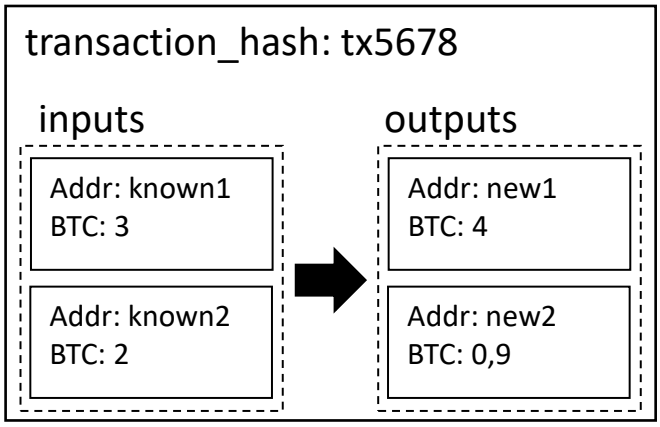


Figure 3.4. *Heuristic II, condition 3: only output* `new2` *meets both conditions 1 and 2. Assuming the sender would rather not pay extra fees, including the second input address indicates that* `new1` *is the receiver address and* `new2` *is for collection the change.*

Conditions 4 and 5 are fairly trivial. First of all, coin generation transactions do not have an input address and, consequently, trying to identify the change address is not relevant in those transactions. Condition 5 states that if any of the input addresses occurs also as an output in the same transaction, the safest bet is to assume the change is sent to that address.

With these rules in place, the address-to-wallet mapper can be made to apply both heuris-

tics. Now the address clustering program functions as shown in Figure 3.5.

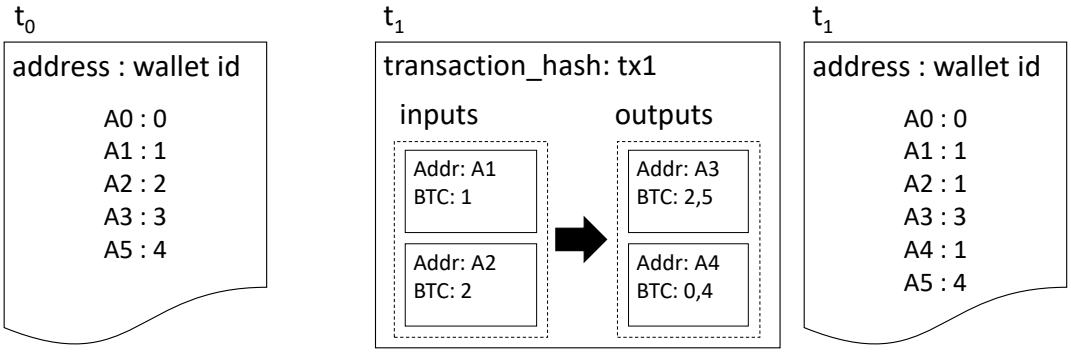


Figure 3.5. *Heuristics I and II in action. Addresses A1 and A2 belong to the same wallet as they are inputs in the same transaction. A4 meets the conditions of a one-time change address.*

Our one-time change address conditions state that there must be exactly one address which both is a new one and receives an amount lower than the smallest input. These conditions are met only by address A4 and, thereafter, the wallet mapping algorithm associates address A4 with the same user ID as addresses A1 and A2. What is more, if A4 is seen together with address A5, it is safe to state that the same user controls addresses A1, A2, A4 and A5. Thus, heuristic II enables us to detect a single wallet with four addresses whereas heuristic I alone would have resulted in two separate wallets containing two addresses. Figure 3.6 shows the described transaction and the changes in address-wallet pairings.

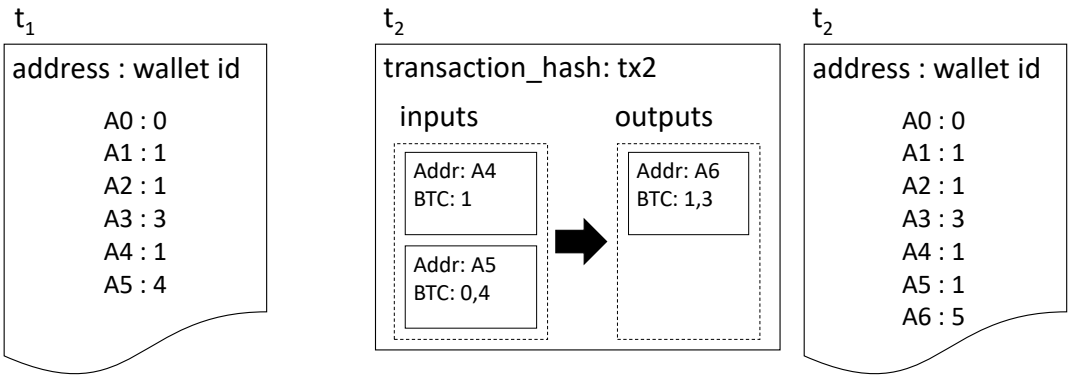


Figure 3.6. *Former change address A4 is used in a multi-input transaction, A5 can be assigned to the same wallet as A4. As heuristic II identified A4 as the one-time change address in the previous transaction, we can assign A1, A2, A4 and A5 to the same wallet.*

Considering that our goal is not to track any specific users but to construct a network for studying the phenomenon, our address clustering script should be accurate enough. Even though neither this thesis nor any of the previous researches manages to construct a completely accurate representation of the network, a sanity check may be performed.

Table 3.1 compares the results of some existing papers to our algorithm when applied to the same timespan.

Table 3.1. *Benchmarking address clustering results with several other studies, some of which only apply heuristic I.*

Research	H. I	H. II	Addresses	Wallets	Ratio
Androulaki et al. (2013)	X		1 632 648	1 069 699	1,5
This paper	X		1 632 252	1 224 451	1,3
Androulaki et al. (2013)	X	X	1 632 648	693 051	2,5
This paper	X	X	1 632 252	660 047	2,5
Ron and Shamir (2013)	X		3 730 218	2 460 814	1,5
This paper	X		3 730 472	2 461 002	1,5
This paper	X	X	3 730 472	1 150 931	3,2
Meiklejohn et al. (2013)	X	X	12 056 684	3 354 831	3,6
This paper	X	X	12 055 131	2 710 944	4,4
Bovet, Campajola, Mottes et al. (2019)	X	X	304 111 529	16 749 939	18,2
This paper	X	X	345 851 379	96 921 461	3,6

As Table 3.1 shows, both the number of addresses and the clustering results are aligned to a great extent. The research carried out by Bovet, Campajola, Mottes et al. (2019) seems to be an outlier when considering the ratio of addresses and wallets. The same results are reported in an overview article written by mostly the same authors (Vallarano et al. 2020) and, thus, the authors most likely use a more aggressive algorithm for address clustering. Another option is a programming error – after all, the authors state the algorithm aims to avoid false positives (Bovet, Campajola, Mottes et al. 2019; Vallarano et al. 2020).

Even though the results presented in Table 3.1 indicate that the used algorithm clusters addresses quite successfully, it should be noted that other techniques and approaches have been proposed as well. For example, machine learning can be used for classifying users – such as exchanges, mixing services, mining pools and personal users – and, for example, mixing services can be identified with around 95 % accuracy (Burks et al. 2017; Ermilov et al. 2017). With such a technique, the multi-input heuristic could be skipped on transactions where mixing service addresses seem to be present. However, the authors used proprietary data sets to train their machine learning models which makes the technique a less-intriguing option for this thesis.

Other methods have been applied, too. Meiklejohn et al. (2013) associate addresses to real-life identities by collecting off-chain data and participating in several transactions with known exchanges. The authors use the collected data to merge wallets that their clustering algorithm was unable to associate with each other. Regardless of the clustering

algorithm, the method can be used to further enhance clustering results, even though collecting such data requires some work.

3.4 Creating a wallet-level transaction log

Having performed address clustering, we can transform the transaction data to wallet level. Figure 3.7 presents two simultaneous transactions in which addresses have been mapped to wallet IDs.

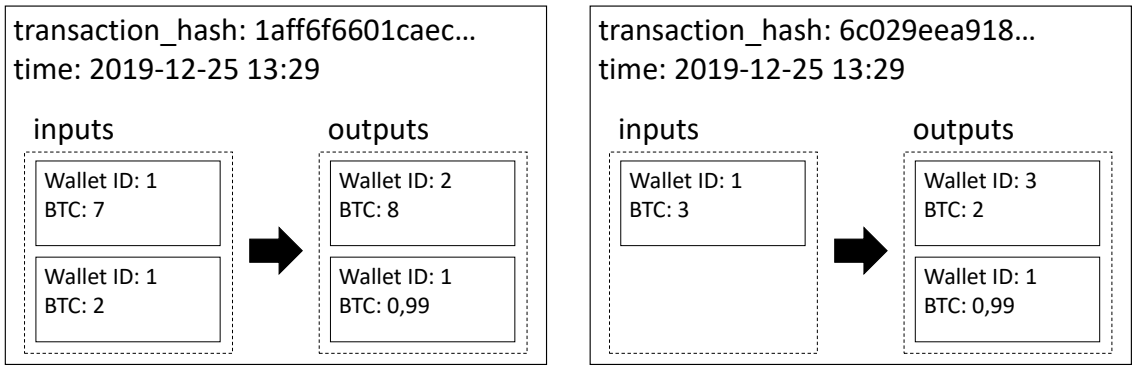


Figure 3.7. *Transaction log can be enriched with the newly obtained Wallet IDs. This way we can actually see which part of the input was sent to another wallet and which is the change.*

Now we can form a transaction log. When creating the log, coin generation transactions are omitted, as they form a very specific type of transaction which is fairly distant from the general idea of using currency as a medium for exchange. In addition, all within-wallet transfers are ignored as they would lead to problems in some of the latter stages. Table 3.2 presents a complete transaction log based on the transactions of Figure 3.7.

Table 3.2. *Transaction log for the transactions presented in Figure 3.7*

Wallet ID	Timestamp	Role	BTC
1	2019-12-25 13:29	Sender	10
2	2019-12-25 13:29	Receiver	8
3	2019-12-25 13:29	Receiver	2

What should be noted is that wallet 1 has only one entry in the table: the rows are grouped by wallet ID, timestamp and transaction role. In addition, the sent amount does not necessarily match the sum of inputs but the outputs to other wallets. After these steps, the transaction log is in rather universal format and, hence, we are able to use non-Bitcoin-specific methods from this point onwards.

4 METHODOLOGY

This section introduces the methodology used for analysing the wallet-level transaction log, the output of data pre-processing. The first subsection presents how a transaction log – be it Bitcoin or some other asset – can be converted into investor states and investor networks. The latter parts describe how this thesis extracts the underlying community structure of the formed investor networks and how the evolution of such communities can be quantified.

4.1 Deriving investor networks from transaction log

Having clustered Bitcoin addresses to wallets, we could effortlessly construct a Bitcoin user network where transactions form the links between users. However, as we are trying identify clusters of wallets with similar behavior patterns, a different approach is needed. Tumminello et al. (2011) introduce the method of statistically validated networks in which nodes are linked if the similarity is too high to be explained by randomness. The authors demonstrate the usefulness of statistically validated networks by applying the method to three systems of varying sizes and domains. The method has also proved itself useful when, for example, comparing stock portfolio structure and trading behavior (Bohlin and Rosvall 2014) and analyzing the existence of investor clusters in the stock market (Baltakienė et al. 2019; Baltakys, Kanniainen et al. 2018; Musciotto et al. 2016). Given the similarity of this thesis' objectives, our methodology is aligned with the aforementioned studies to a great extent.

To make use of the methodology of statistically validated networks, a metric is needed for evaluating wallet similarity. First, we re-sample the wallet-level transaction log to hourly slots and, having done that, we proceed to assign a categorical state variable for each active wallet for each one-hour slot. To assign the categorical variable characterizing trading behavior, a scaled net volume ratio is calculated for each wallet *w* and time span *t*, similarly as Musciotto et al. (2016) and Baltakienė et al. (2019):

$$r(w,t) = \frac{V_b(w,t) - V_s(w,t)}{V_b(w,t) + V_s(w,t)} \tag{4.1}$$

The variables V_b and V_s represent the buying (receiving) and selling (sending) volume.

Even though Bitcoin can also be used as a currency, this thesis chooses to use term buying (selling) to represent a transaction where a user receives (sends) Bitcoin. When Bitcoin is sent from person A to person B, it is highly likely that person B sends some amount of fiat currency the other way, though, it is possible that B offers services or physical goods. For the scaled net volume ratio to be of any relevance, it is vital to remove self-transactions from the transaction log, as described in Chapter 3.4. As our transaction log is composed of inter-wallet transactions, we can proceed and assign the categorical state variables by defining a threshold value θ as follows:

$$\begin{cases} b \text{ (primarily buying)}, & \text{if r(w,t)} > \theta \\ s \text{ (primarily selling)}, & \text{if r(w,t)} < -\theta \\ bs \text{ (buying and selling)}, & \text{if } -\theta \leq \text{r(w,t)} \leq \theta \end{cases}$$

As this research focuses on the alignment of wallet behavior, we are mainly interested in events where two wallets *i* and *j* both are in buying state or alternatively both in selling state. To compare trading states, an hourly trading vector is constructed for each wallet and for each month. In other words, a 30-day month is represented by a vector of length $T = 720$, as $30 * 24 = 720$ hours, where zeros represent inactive hours and ones indicate activity. Then, the similarity can be assessed by performing a hypergeometric test (Rice 2006). If wallet *i* is N_i^P times in state *P*, wallet *j* is N_j^Q times in state *Q* and the whole span is of length *T*, the probability of having X random co-occurrences can be obtained from the hypergeometric distribution as follows (Baltakienė et al. 2019; Rice 2006; Tumminello et al. 2011):

$$H(X|T, N_i^P, N_j^Q) = \frac{\binom{N_i^P}{X}\binom{T-N_i^P}{N_j^Q-X}}{\binom{T}{N_j^Q}} \tag{4.2}$$

Given that we are mainly interested in the event where the wallets are in the same state, the Q may be replaced with P. Furthermore, if we use $N_{i,j}^P$ notation to represent the number of hours the wallets i and j are in the same state, a p-value can be calculated as follows (Tumminello et al. 2011):

$$p(N_{i,j}^P) = 1 - \sum_{X=0}^{N_{i,j}^P - 1} H(X|T, N_i^P, N_j^P) \tag{4.3}$$

As we apply Equation 4.3 to a pair of active wallets in a given month, we will obtain a p-value which can then be compared to a certain threshold. The fundamental concept is to have a null hypothesis which states that the co-occurrences are due to randomness and, if the obtained p-value is very low, the null hypothesis may be discarded. A typical

threshold for statistical significance is either 0.05 or 0.01 (Rice 2006), which some studies further lower by performing a multiple test correction (Baltakienė et al. 2019; Tumminello et al. 2011), such as the Bonferroni correction (Bonferroni 1936) or the False-Discovery Rate correction (Benjamini and Hochberg 1995).

However, there is no clear consensus of when and how the multiple-test correction should be used, if at all (Cabin and Mitchell 2000; Moran 2003; Perneger 1998). Perneger (1998) argues that the multiple test correction is relevant only when the research is focused on testing the global null hypothesis and, by lowering the threshold of statistical significance, the amount of false negatives will rocket, which can be considered problematic, too. According to Perneger, the use of Bonferroni correction in our wallet similarity testing would indicate that we are actually investigating if there are any wallets that trade similarly. However, we take the existence of such aligning trading patterns for granted and rather aim to extract clusters of wallets behaving similarly. Considering our goal, false negatives are equally harmful as false positives and, thus, this research uses a fixed threshold of $\alpha = 0.001$ for each month of data.

To conclude, we construct a similarity network for each month, having a separate network for buying and selling state alignment. Two wallets are linked in the network if the hypergeometric test suggests that the wallets take the same trading position too often to be explained by randomness.

4.2 Grouping identical wallets to reduce testing

When there are *n* active wallets in a given month, the number of wallet pairs to be tested is $n * (n-1)/2$. Therefore, the number of tests to be performed increases exponentially as the number of active wallets grows. As the amount of active wallets has been relatively high for the last few years, we reduce the number of tests needed by grouping identical trading vectors, similarly as Bohlin and Rosvall (2014). For example, if we have six wallets, as presented in Table 4.1, the number of tests needed is $6 * 5/2 = 15$.

Table 4.1. *Six wallets with two unique trading position vectors*

Wallet ID	Activity vector
W1	['2018-05-17 23:00','2018-05-18 21:00']
W2	['2018-05-17 23:00','2018-05-18 21:00']
W3	['2018-05-17 23:00','2018-05-18 03:00']
W4	['2018-05-17 23:00','2018-05-18 03:00']
W5	['2018-05-17 23:00','2018-05-18 03:00']
W6	['2018-05-17 23:00','2018-05-18 03:00']

Wallets 1 and 2 have identical buying states, as do wallets 3, 4, 5 and 6. We can group

the wallets as shown in Table 4.2.

Table 4.2. *Wallet groups representing unique trading vectors*

Wallet group ID	Activity vector
G1	['2018-05-17 23:00','2018-05-18 21:00']
G2	['2018-05-17 23:00','2018-05-18 03:00']

By comparing wallet groups, which represent unique trading vectors, the number of tests required decreases significantly. For example, we would have only one inter-group pair-test with the groups presented in Table 4.2, though, it should be noted that now there is a need to test the self-link as well: if pair $(G1, G1)$ is statistically significant, there is a link between each pair of wallets belonging to group *G1*.

4.3 Community detection

The pair-wise testing helps us determine whether two nodes behave similarly enough for them to be linked in our statistically validated network of Bitcoin users. However, as Bitcoin is largely an investment asset and not only a currency, it would be fascinating to identify not only groups of similar investors in addition to pairs of individuals. As demonstrated by Bohlin and Rosvall (2014), the complex networks toolbox provides methods for extracting community structure, namely the map equation algorithm Infomap.

As the Infomap algorithm is based on simulating information flow within a network (Bohlin, Edler et al. 2014), it needs the nodes and links of a network to function as intended. To cope with our hardware limitations and the fact our monthly similarity networks may have tens of millions of links even on the wallet group level, the community detection algorithm is run for the said wallet groups. The resulting community structure would most likely be largely similar was Infomap run on wallet level. After all, the authors of the algorithm have detected communities from a grouped set of data in their stock market research (Bohlin and Rosvall 2014).

Infomap provides a fairly straight-forward, user-friendly API, whereas some other community detection algorithms require the user to define a set of parameters. Infomap ignores self-links by default but, other than that, there are few decisions to make before and arguments to pass in while running the algorithm. By writing the monthly group-level similarity network nodes and links to a text file in Pajek's `.net` format (Bohlin, Edler et al. 2014) and running Infomap, we will obtain a list of nodes and their communities as an output. It is worth mentioning that Infomap associates each node with exactly one community (Bohlin, Edler et al. 2014) while, for example, the Clique Percolation Method is likely to produce communities which may overlap and some nodes are left out altogether (Barabási et al. 2016; Palla et al. 2007). When our database is enriched with the newly acquired commu-

nity IDs, we have data similar to the format presented in Table 4.3.

Table 4.3. *Community detection results on group level*

Timestamp	Buying or selling	Group ID	Community ID
2018-01	B	G1	1
2018-02	B	G1	28
2018-02	B	G2	104

However, we must go back to wallet level to be able to compare months and the evolution of communities. Even though the groups in Table 4.3 represent nodes which have an identical trading state vector, the groups are only concerned with the data of that particular month. For example, the wallet-level representation of Table 4.3 could be as shown in Table 4.4.

Table 4.4. *Community detection results on wallet level*

Timestamp	Buying or selling	Wallet ID	Group ID	Community ID
2018-01	B	W1	G1	1
2018-01	B	W2	G1	1
2018-02	B	W1	G1	28
2018-02	B	W2	G2	104

From the wallet-level representation of Table 4.4 we notice that wallets W1 and W2 belonged to the same community in January 2018 but had timed their trades differently enough in February to end up in separate communities. Therefore, when comparing similarity networks of different months, the analysis must be done on wallet level.

4.4 Community evolution

Most often network analyses focus on single snapshots of networks. However, while such analyses may be able to derive useful results regarding the complex structure, they fail to provide insights on how the network evolves over time (Greene et al. 2010; Hopcroft et al. 2004). Thus, another type of approach is needed for characterizing the properties of a temporal network.

As community evolution research focuses on dynamic networks, a set of change events must be defined. As Dakiche et al. (2019) state, most studies (Asur et al. 2009; Bródka et al. 2013; Chen et al. 2010; Greene et al. 2010; Palla et al. 2007; Takaffoli et al. 2010) define events in a fairly similar manner. Generally, the events related to communities in a dynamic network can be listed as follows (Dakiche et al. 2019):

- Birth

- Death
- Growth
- Shrinking
- Merging
- Splitting

While most studies agree on the events related to the lifespan of a community, there are several different sets of methods for detecting such events. Dakiche et al. (2019) classify the techniques into 4 main categories:

1. Detecting communities independently and then matching
2. Detecting communities based on previous network snapshots
3. Simultaneously detecting communities for all snapshots
4. Dynamically detecting communities based on previously found ones and changes in network

This thesis takes the first approach by building the network for each month and then using Infomap to detect communities for the given month. Methodologically we are close to Bródka et al. (2013), Asur et al. (2009) and Greene et al. (2010). However, to be able to match the communities of subsequent snapshots — to detect that community i of snapshot t_1 is the same as community j in snapshot t_2 — a set of metrics must be defined to quantify the similarity of two communities.

Metrics for defining community evolution events

Before defining any similarity metrics, we should agree on some of the network-related notations. As we are interested in groups of nodes, we will refer to the members of a community with C. Moreover, a specific community i at time t is denoted with C_t^i and, thus, $|C_t^i|$ is the number of Bitcoin wallets included in the community. When there is a need to refer to all active wallets at time t, we will do that with G_t.

Having introduced the common language, we can move on to discussing similarity measures. Firstly, Bródka et al. (2013), Asur et al. (2009) and Greene et al. (2010) all rely on node overlap while comparing communities. Asur et al. (2009) do not have a specific similarity function but instead alter the sets of nodes for which the overlap is computed. In other words, they perform different calculations while extracting merging events than, for example, when analyzing whether a community has dissolved. Contrary to that approach, the other two studies have defined a similarity function which will be applied on pairs of communities.

Greene et al. (2010) make use of Jaccard coefficient for binary sets and calculate the similarity of two communities as follows:

$$sim(C_t^i, C_{t+1}^j) = \frac{|C_t^i \cap C_{t+1}^j|}{|C_t^i \cup C_{t+1}^j|} \tag{4.4}$$

From the definition it follows that the Jaccard coefficient similarity is a symmetric function. What is more, it is very robust method for comparing how identical two communities are.

Bródka et al. (2013) have a slightly different approach. Where the Jaccard coefficient similarity focuses on the number of matching nodes, Bródka et al. (2013) have defined a similarity function which which also is also concerned with the relevance of overlapping nodes. Thus, in addition to comparing the quantity of overlap, they aim to evaluate the quality of overlap. Their metric $inclusion\ I$ is calculated as the product of quantity and quality, and the authors use their own Social Position function SP for the quality measure:

$$I(C_t^i, C_{t+1}^j) = \frac{|C_t^i \cap C_{t+1}^j|}{|C_t^i|} * \frac{\sum_{x \in (C_t^i \cap C_{t+1}^j)} SP_{C_t^i}(x)}{\sum_{x \in (C_t^i)} SP_{C_t^i}(x)} \tag{4.5}$$

The $Inclusion$ function is slightly less readable but the main idea behind it is clear. First, the function inspects which nodes of community C_t^i are present in community C_{t+1}^j. Having done that, the sum of importance is calculated for the nodes present in both communities and the obtained value is then compared to the original importance of the original C_t^i. As both quantity and quality receive values from range [0,1], the values of $inclusion$ range from 0 to 1, too.

Given the definitions, it should be noted that $sim(C_t^i, C_{t+1}^j) = sim(C_{t+1}^j, C_t^i)$ but most often $I(C_t^i, C_{t+1}^j) \neq I(C_{t+1}^j, C_t^i)$. The difference of these functions is best demonstrated by applying them on a pair of communities (C_t^i, C_{t+1}^j) where $|C_t^i| \ll |C_{t+1}^j|$. Even if all nodes of C_t^i are included in C_{t+1}^j, the Jaccard coefficient similarity states that these communities are similar only to a small extent. However, the inclusion metric returns value 1, as all nodes of C_t^i have ended up in C_{t+1}^j.

This thesis does not copy any of the approaches described above but follows them quite closely, especially the one by Bródka et al. (2013). Firstly, it should be remembered that the networks of active Bitcoin wallets vary significantly from month to month. To be able to compare relevant numbers, we will define a separate similarity function for comparing the overlap of those nodes which were active both months:

$$ActiveOverlap(C_t^i, C_{t+1}^j) = \frac{|C_t^i \cap C_{t+1}^j|}{|C_t^i \cap G_{t+1}|} \tag{4.6}$$

However, the intersection $C_t^i \cap G_{t+1}$ returns an empty set when none of C_t^i members are active at time t+1. To handle the risks related to relative values, another helper function is defined to identify dissolving communities prior to applying Active Overlap on them. Active Size return the relative amount of members of community C_t^i which are active at time T:

$$ActiveSize(C_t^i, T) = \frac{|C_t^i \cap G_T|}{|C_t^i|} \tag{4.7}$$

Having defined these functions, we can go on to define the boundaries for different community evolution events.

Dissolving

Dissolving, the end of lifespan for a community, occurs when a community is considered not to be present in the next snapshot. Asur et al. (2009) have the strictest rule: community C_t^i dissolves if none of its members are active in the subsequent snapshot. Bródka et al. (2013) have a similar idea but do allow some overlap. They consider community C_t^i to dissolve if there is no C_{t+1}^j so that $I(C_t^i, C_{t+1}^j) \geq 10\%$ or $I(C_{t+1}^j, C_t^i) \geq 10\%$.

Greene et al. (2010) have a slightly more relaxed definition. They do not require a community to be active at every step, but instead consider it dissolved only when a similar-enough community has not been observed for d successive snapshots. However, running their algorithm with a parameter value $d = 1$ would result in a very similar event log as the other definitions. With $d = 1$, a given community C_t^i dissolves if there is no such a pair of communities (C_t^i, C_{t+1}^j) for which $sim(C_t^i, C_{t+1}^j) > \theta$.

This thesis follows the other researches quite closely. A community C_t^i is considered a dissolving one when $ActiveSize(C_t^i, t+1) < \theta_{ActiveSize}$ or if there is no community C_{t+1}^j which meets the following two conditions:

$$ActiveOverlap(C_t^i, C_{t+1}^j) \geq \theta_{overlap}$$

and

$$ActiveOverlap(C_{t+1}^j, C_t^i) \geq \theta_{overlap\_reverse}$$

In other words, there is no need to compare the overlap of active wallets if only a tiny fraction of the members of C_t^i are active at t+1. However, if a reasonable share are active, the community is compared with each of the t+1 communities. We are mostly interested in the forward-looking $ActiveOverlap(C_t^i, C_{t+1}^j)$, which answers the question where the nodes of C_t^i are at $t + 1$. Moreover, a looser sanity check $ActiveOverlap(C_{t+1}^j, C_t^i)$ is performed to evaluate whether C_t^i is of any significance as a source of nodes for C_t^j. The

reverse-order inspection is done to add safety to situations where $|C_t^i| \ll |C_{t+1}^j|$. If a 28-node community matches with a 9000-node one, intuition suggests we will have a more realistic community evolution timeline by considering C_t^i a dissolving community.

Merging

Merging, the event where multiple communities become one, is quite different from dissolving but still the same metrics are useful. For example, Greene et al. (2010) use the Jaccard coefficient similarity to test each pair of communities (C_t^i, C_{t+1}^j), and if multiple communities match with the same C_{t+1}^j, it is classified as a merging event. Bródka et al. (2013) mine merging events in a similar fashion. They consider (C_t^i, C_{t+1}^j) a merging candidate if $I(C_t^i, C_{t+1}^j) \geq \alpha$ and $I(C_{t+1}^j, C_t^i) < \beta$, where α and β are process parameters. Having tested each pair, a merging event is registered if more than 1 C_t^i have matched with the same C_{t+1}^j.

As noted earlier, Asur et al. (2009) do not have a similarity function but define specific calculations for each event. While extracting merging events, they compare pairs of communities at time t to a community at time $t+1$. According to their definition, communities C_t^i and C_t^j merge into C_{t+1}^k if the following condition is met:

$$\frac{|(C_t^i \cup C_t^j) \cap C_{t+1}^k|}{max(|C_t^i \cup C_t^j|, |C_{t+1}^k)|} > \theta_{merge}$$

The definition by Asur et al. (2009) might be the most accurate when focusing solely on merging events. However, our goal is to define events in such a robust way that each community is assigned one event. To achieve the goal, it is preferable option to use aligning metrics for defining each event, and, thus, we will perform the same tests as when extracting dissolving events. If a community does not meet the ActiveSize threshold $ActiveSize(C_t^i, t+1) \geq \theta_{ActiveSize}$, it will not exist in snapshot $t+1$. If the size threshold is met, the non-symmetric ActiveOverlap function will be applied on C_t^i and each C_{t+1}^j with both argument orders. If the pair survives both ActiveOverlap thresholds, it is considered a match. If multiple communities of snapshot t match the same community of $t+1$, the communities merge.

Splitting

Splitting event is basically the opposite of a merge: one community continues as multiple new communities. Thus, for example, Greene et al. (2010) compare each pair (C_t^i, C_{t+1}^j) similarly as when testing for merging events. If the similarity threshold θ is exceeded, the pair is considered a match, and then the count of pairs determines the exact event type. When it comes to observing splitting events, the same C_t^i must match with multiple

communities C_{t+1}^j of the next snapshot. Bródka et al. (2013) take a comparable approach by applying their $inclusion$ function on all pairs, counting matches and then requiring a splitting event to have multiple communities matching with the same destination. Contrary to the pair-count approach, Asur et al. (2009) once again define the splitting event by testing both communities simultaneously:

$$\frac{|(C_{t+1}^j \cup C_{t+1}^k) \cap C_t^i|}{max(|C_{t+1}^j \cup C_{t+1}^k|, |C_t^i)|} > \theta_{split}$$

Moreover, the authors require that for both resulting communities C_{t+1}^j and C_{t+1}^k over 50 % of their nodes must exist in the source community C_t^i. Overall, even though the three definitions differ, they can not be considered conflicting. The basic idea behind the definitions is the same, and after that the question is about how much value is given to each property.

Our split definition follows the one of merging events. A community must pass the activity test $ActiveSize(C_t^i, t+1) \geq \theta_{ActiveSize}$ to continue to pair-wise testing. Just as with merging events, the pair is considered a match if $ActiveOverlap(C_t^i, C_{t+1}^j)$ and $ActiveOverlap(C_{t+1}^j, C_t^i)$ meet the thresholds. A community is said to split if it matches with multiple C_{t+1}^j communities.

Continuing, growing and shrinking

Continuing, growing and shrinking are very similar events in the sense that each of them has one input and one output community. As finding a pair of matching communities is still fundamentally the same task as with the above-mentioned events, the events can be extracted with largely similar rules. However, contrary to split and merge events which have a one-to-many relation between source and destination communities or vice versa, now C_t^i must match with exactly one C_{t+1}^j and that particular C_{t+1}^j is allowed to match with only one C_t^i. Both Bródka et al. (2013) and Greene et al. (2010) follow this same logic.

After identifying a pair where C_t^i and C_{t+1}^j only match with each other, the only task remaining is determining whether we have a continuing, growing or shrinking event. The relative size change is calculated as follows:

$$\Delta s = \frac{|C_{t+1}^j| - |C_t^i|}{|C_t^i|}$$

and then the obtained value is compared to a threshold value which could be, for example, in the region of 10 per cent.

$$\begin{cases} \text{growing}, & \text{if } \Delta s > \theta_{continue} \\ \text{shrinking}, & \text{if } \Delta s < -\theta_{continue} \\ \text{continuing}, & \text{if } -\theta_{continue} \leq \Delta s \leq \theta_{continue} \end{cases}$$

Even though continuing events may seem trivial and not worth mentioning, considering our highly dynamic data set and the non-transparent nature of Bitcoin investor movements, all aforementioned survival events are of great interest for us. If the research were to focus on some other topic, forming and dissolving events could be the most interesting ones, but for Bitcoin similarity networks it is likely the other way around.

Forming

As forming events are basically the opposite of dissolving events, the definition is also very similar. However, when extracting forming events, we are looking backwards to answer the question, whether any C_{t-1}^i matches with this month's C_t^j. Mathematically expressed, C_t^i forms at time t if there is no C_{t-1}^j meeting these three conditions:

1. $ActiveSize(C_{t-1}^j, t) \geq \theta_{ActiveSize}$
2. $ActiveOverlap(C_{t-1}^j, C_t^i) \geq \theta_{ActiveOverlap}$
3. $ActiveOverlap(C_t^i, C_{t-1}^j) \geq \theta_{ActiveOverlapReverse}$

In addition, we will define that each community active at t_0 has a forming event at that point. The first snapshot has no predecessor and, thus, the same logic can not be applied. However, it is natural to have a forming event at the start of a timeline.

Extracting events and creating community timelines

As all our event definitions are based on the logic of testing each pair (C_t^i, C_{t+1}^j) and the definitions are supposed not to overlap, the set of rules can be presented as a decision tree. Such a decision tree is shown in Figure 4.1. This thesis also chooses to perform the event extraction only for communities with 10 or more members. The absolute size threshold is used to improve the validity of our relative metrics.

In addition to using the proposed framework for extracting community evolution events, we can use the events to build community evolution timelines. For this task, we take a similar approach as Greene et al. (2010): new steps are added at the end of a community evolution vector until the community dissolves. It should be noted that when a community splits, the original community vector is duplicated so that each of the split destinations can be added to a different vector. Figure 4.2 presents a four-timeframe community evolution and Table 4.5 shows how the vectors are built from the event log.

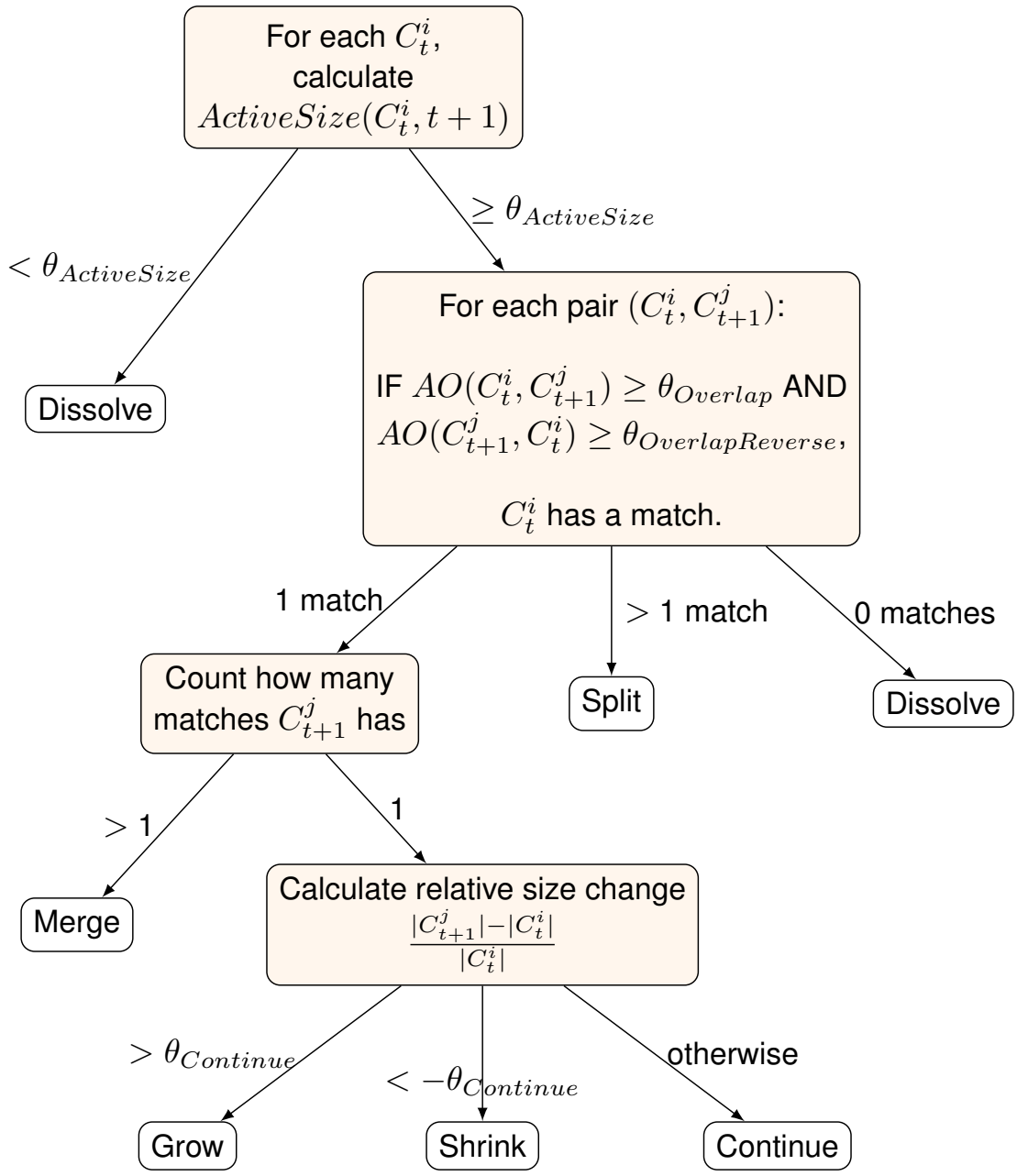


Figure 4.1. *Community evolution event definitions presented as a decision tree. If C_{t+1}^j is not the target community for any C_t^i, the C_{t+1}^j community is born at time t+1. We also discard communities with less than 10 members. $ActiveOverlap$ function is abbreviated to AO.*

Having defined the events and the creation of community evolution vectors in a systematic manner, we can carry out multiple tests which must pass regardless of the choice of parameters. First, each monthly community must be assigned a community evolution event. If the inspected month is the forming month of a community, the community must be associated with a forward-looking event, too. The rule goes both ways: if a community has two events, one of them must be a forming one. In no situation is a community allowed to have more than two events in a single month.

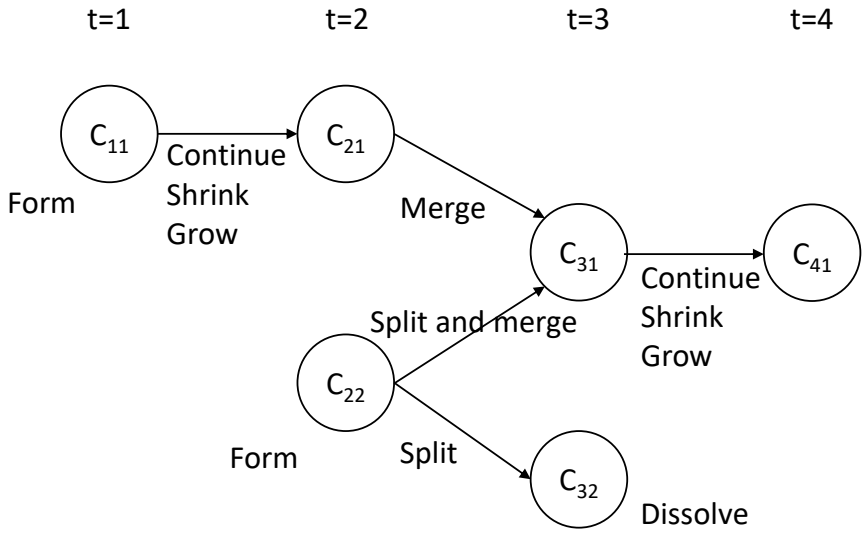


Figure 4.2. *Different community evolution events demonstrated in a span of 4 snapshots, edited from Greene et al. (2010). Due to our event definitions, C_{22} splits into C_{31} and C_{32}, but C_{21} makes the event also a merge. To avoid problems at following stages, only a split event is registered for C_{22}, even though C_{21} is still considered to merge into C_{31}.*

Similar tests can be run for the program creating the community evolution vectors. First, each extracted community event must be included in a community evolution vector. In addition, if an event is included in more than one vector, the vector must also contain at least one split or merge event.

Table 4.5. *Parsing community evolution vectors from the events of Figure 4.2. The split event also splits the vector into two separate branches which share the same initial step. Merging event does not reduce the number of vectors but makes the mergers share the following steps.*

Vector ID	Start of span	Last active snapshot	Communities
V1	t=1	t=4	$[C_{11}, C_{21}, C_{31}, C_{41}]$
V2	t=2	t=4	$[C_{22}, C_{31}, C_{41}]$
V3	t=2	t=3	$[C_{22}, C_{22}]$

The last set of tests verifies the sanity of community evolution vectors. Without comparing the events of a vector, the following two tests can be run. First of all, each vector must have exactly one forming event and there must not be another event prior to that. The same logic can be applied the other way, too: each vector must have a dissolving event and all the other events must have a timestamp earlier than the one of dissolving event. Though, the end of our timespan, December 2019, is slightly different as there is no next month. If we assign a dissolving event for each community alive at December 2019, the test can be run as-is.

5 RESULTS

This section introduces the results obtained by applying the research methodology described in Chapter 4. The first part presents the results of community evolution analysis, and the latter part displays key observations related to Bitcoin network snapshots and similarity networks.

5.1 Community evolution

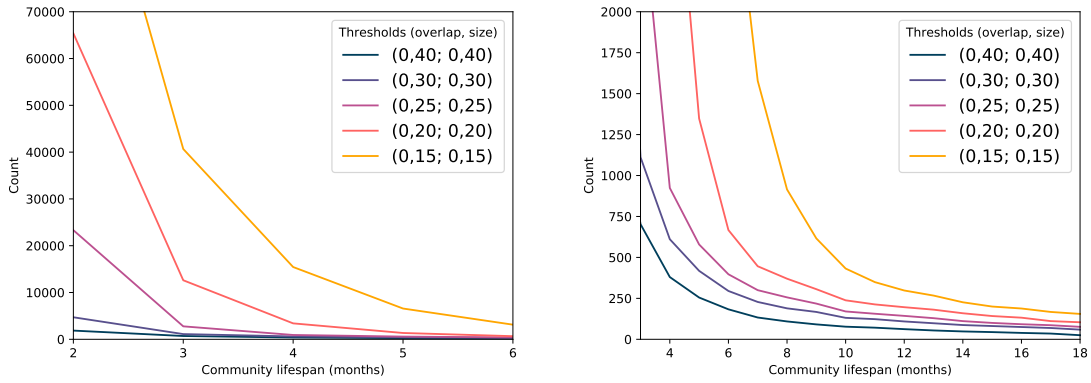
The community evolution results are analyzed from different perspectives. First, the sensitivity of parameter choices is inspected by analyzing different result sets. The second part focuses on the events obtained by running the event extraction algorithm on a given parameter set. The third subsection investigates long-lived communities from several perspectives.

Sensitivity analysis on process parameters

To be able to apply the community event extraction algorithm to our monthly snapshots, we will need to choose a set of threshold parameters. To understand how the parameters affect the event extraction results, a sensitivity analysis is carried out. Figure 5.1 visualizes how varying $\theta_{ActiveSize}$ and $\theta_{ActiveOverlap}$ are related to the amount of communities surviving at least n months. As the sanity check parameter $\theta_{ActiveOverlapReverse}$ is not of similar interest to us, it is fixed at 0.10.

Based on Figure 5.1, it seems that $(0.30; 0.30)$ is too strict of a threshold to have any reasonable amount of surviving communities. That itself can be considered an interesting result if we consider the traditional thresholds by Bródka et al. (2013), Greene et al. (2010) and other studies. It is quite common to require the similarity to be from range $[0.5; 1]$, but our Bitcoin trader communities would not survive such a requirement, as the active overlap threshold of 0.30 is already eliminating most of the persistence.

Even though other studies use higher thresholds, it does not necessarily indicate that the same values should be applied on the Bitcoin network. It should be remembered that the communities in phone call, email or collaboration networks are of completely different nature than the one we are analyzing. For example, changes in email network might be a



(a) *At least 2 months* ***(b)*** *At least 3 months*

Figure 5.1. *Experimenting different ($\theta_{ActiveOverlap}$, $\theta_{ActiveSize}$) pairs. Parameter set (0.30; 0.30) eliminates almost all continuity whereas (0.20; 0.20) is not strict enough.*

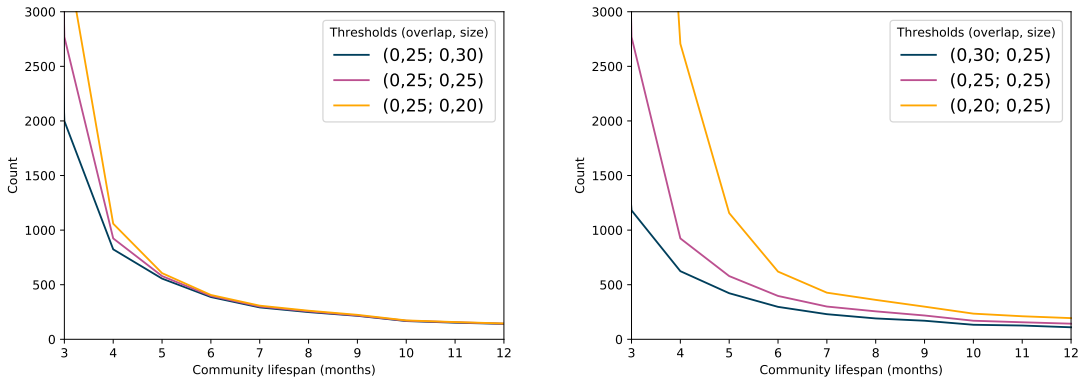
result of organizational changes, which often occur infrequently. Phone calls are common between family members, and families remain relatively unchanged when analyzing a few weeks or months of data. On the other hand, the timing of Bitcoin trades might base on following a specific group of information sources. For example, a single news article or a change in Bitcoin price may result in significant changes in the structure of Bitcoin investor communities.

The other side of the spectrum shows that a threshold of $(0.15; 0.15)$ would result in a drastically different distribution of community life spans. Intuition and Figure 5.1 suggest that $(0.15; 0.15)$, and most likely $(0.20; 0.20)$, are too forgiving. Choosing a threshold that low would basically mean that our results would have a plenty of persisting communities, even though the similarity between snapshots would be so limited that it would be uncomfortable to call it the same community. After all, the goal of this thesis is not to maximize the number of persisting communities but to quantify a real-world phenomenon.

If we continue to inspect the region of $(0.25; 0.25)$ and vary one parameter at a time, we can analyze how sensitive the event extraction algorithm is to changing each of the parameters. Figure 5.2 presents the results of that test.

Figure 5.2 shows that altering $\theta_{ActiveSize}$ within range $[0.20; 0.30]$ has little effect on the lifespan distribution. However, lowering $\theta_{ActiveOverlap}$ to 0.20 would result in clearly different distribution, whereas a stricter threshold of 0.30 would return a result set roughly half the size of the one with $\theta_{ActiveOverlap} = 0.25$. Both parameter choices would probably be equally correct but we will continue with the value 0.25 to have a larger set of community lifespans to analyze.

Community evolution event distribution



(a) $\theta_{ActiveOverlap} = 0.25$ ***(b)*** $\theta_{ActiveSize} = 0.25$

Figure 5.2. *Sensitivity analysis on the two main thresholds. Adjusting* $\theta_{ActiveSize}$ *has little effect, whereas* $\theta_{ActiveOverlap}$ *is much more sensitive to changes.*

The following section focuses on a result set obtained by running community evolution event extraction with the parameters presented in Table 5.1. Communities with less than 10 members are excluded from the analysis due to the challenges related to determining significant overlap for tiny communities.

Table 5.1. *Thresholds for community evolution analysis.*

Threshold	Value
$\theta_{ActiveSize}$	0.25
$\theta_{ActiveOverlap}$	0.25
$\theta_{ActiveOverlapReverse}$	0.10
$\theta_{ContinueSize}$	0.10
Community size at least	10

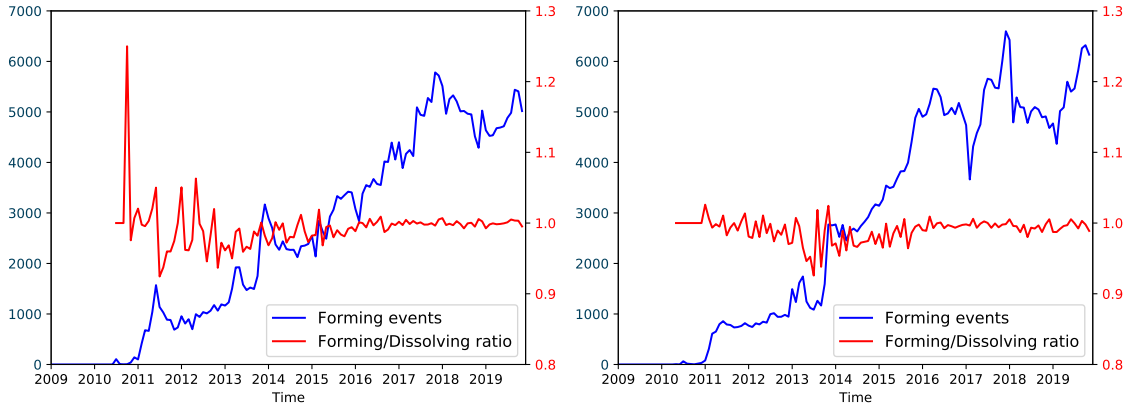
When the algorithm is run with the given parameters, the vast majority of events are forming and dissolving ones. The results also show that splitting is a more common method for surviving than the opposite event, merging. In addition, the total count of growing events is over twice the sum of continuing and shrinking events. Table 5.2 presents the event count on a yearly level. It should be noted that the count of forming and dissolving events does not necessarily have to match even if each community timeline starts with a forming event and ends in dissolving one. Due to merging events, two or more communities may form independently, become one community at some point and then dissolve together. Split events are the opposite: form once, dissolve at least twice.

When the monthly forming and dissolving events are plotted as a function of time, we can observe multiple characteristics of Bitcoin investor community evolution. First of all, the trend lines for buying and selling networks have a fairly similar shape. Secondly, the amount of forming communities has an inclining trend and peaks at the end of 2017.

Table 5.2. *Yearly number of community evolution event types. Forming and dissolving events dominate the table. Splits are more common than merges.*

Year	Form	Dissolve	Merge	Split	Grow	Continue	Shrink
2009	0	0	0	0	0	0	0
2010	455	454	0	0	2	0	0
2011	17 763	17 931	36	182	151	37	44
2012	22 463	22 749	62	435	393	96	102
2013	40 542	41 442	245	1 158	1 071	233	415
2014	61 923	63 222	273	1 184	1 410	424	591
2015	82 077	83 140	375	971	1 598	325	479
2016	104 945	105 294	103	259	1 149	207	197
2017	120 087	120 193	57	181	1 133	185	154
2018	121 153	121 490	144	522	1 652	310	284
2019	124 234	124 812	180	338	1 448	285	236
Total	695 642	700 727	1 475	5 230	10 007	2 102	2 502

The end of 2017 happens to be the point in time when Bitcoin flirted with the all-time-high $20k exchange rate, which resulted in a very high transaction volume for that month. Radical changes in Bitcoin price also tend to lure new investors to join the Bitcoin network, and both the existing and the newly-joined investors may interpret the market situation in various ways. Thus, it is only natural that the amount of communities at the end of 2017 was higher than other months. Figure 5.3 presents forming event count and the ratio of forming and dissolving events for buying and selling networks.

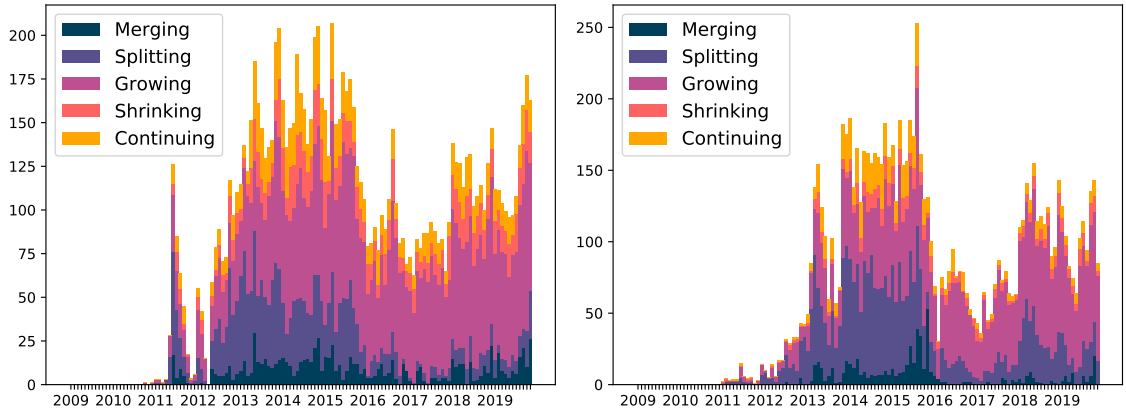


(a) *Buying* ***(b)*** *Selling*

Figure 5.3. *Forming events and forming/dissolving ratio as a function of time. The trend is inclining and reaches its peak at the end of 2017 when Bitcoin price peaked, too. The ratio of forming and dissolving events is close to 1.0 all the time.*

As the volume of forming and dissolving events is relatively high compared to so-called survival events, it is not possible to observe any meaningful survival patterns from Fig-

ure 5.3. Instead, we will plot the monthly distribution of survival events separately and show it in Figure 5.4. A few interesting observations can be made from the plot. First of all, the earliest survival event peaks occur in the summer of 2011 in the buying network whereas the selling communities do not reach a similar event volume until year 2013. It is highly likely that the temporal emergence of continuous buying communities is related to Bitcoin pricing climbing to a new high of $32 in the summer of 2011. The exchange rate did not reach similar heights until 2013.



(a) *Buying* ***(b)*** *Selling*

Figure 5.4. *Survival events as a function of time. Increasing community size increases over time, which favors growing events over shrinking ones. Splitting is more common than merging. Buying communities have their first survival peak almost 2 years earlier than the selling ones.*

Another observation is that the amount of survival events is higher between 2013 and 2015 than in the following years, even though the amount of communities increases quite steadily as discussed earlier and shown in Figure 5.3. In other words, the relative amount of surviving communities was remarkably higher during 2013–2015. That could indicate that the communities of 2013, 2014 and 2015 were fitter for survival or the environment was more favorable in one way or another. Another possibility is that the consistently grown number of investors adds significant layer of noise on top of the actual communities, which makes it challenging to track the evolution of communities. Of course, it may merely be that the communities are not as fit as they were in the earlier stages of Bitcoin.

The last thing to discuss of Figure 5.4 is the frequency of different event types. Even though we register community evolution events from the point of view of a source community, which most likely boosts the count of merging events, splitting seems to be a much more common survival method. In addition, as the pair-matching technique relies on finding a $t + 1$ community with high enough overlap, it would not even be possible to have any splitting events with $\theta_{ActiveOverlap} = 0.50$, whereas merging events can be found even when the parameter is assigned a value of 1.0. In that sense, the frequency of splitting events compared to merging ones is rather intriguing.

The three other event types — continuing, growing and shrinking — are responsible for the majority of survival events. With the continuing threshold set to 10 % change in size, which is actually a rather strict threshold, most of the events are classified as growing ones. As we earlier noted that splitting is a more popular survival method than merging, it is quite unintuitive that growing events are much more frequent than continuing and shrinking ones. However, as the average community size increases over time, it could also be that new traders join the communities regardless of the event type. For example, a splitting event does not necessarily mean that the new branches would have a lower amount of members than the original one. Instead, the definition merely states that the original community continues its life span as two or more independent communities.

Long-lived communities

While analyzing the distribution of community events gives a good overview of the evolution, it lacks the perspective of individual communities. We are especially interested in communities which exist for multiple consecutive snapshots. For example, if two communities each live for 2 months, we have two survival events. However, we may have a similar log of survival events if a community exists for 3 subsequent months. Therefore, it is useful to inspect the results from another point of view.

As trivial as that may sound, first we have to define how to count active communities. As we create community evolution vectors according to the methodology presented in Figure 4.2 and Table 4.5, the same community may be included in multiple vectors. Therefore, it is possible to either count the active vectors or the unique communities under such vectors. Table 5.3 presents the problem by showing two community evolution vectors which merge at $t = t_3$. Given the data presented in Table 5.3, one may say that we have two long-lived community evolution vectors active at t_3 while the other interpretation is that the number of active communities is 1, even though no community dissolves between t_2 and t_3.

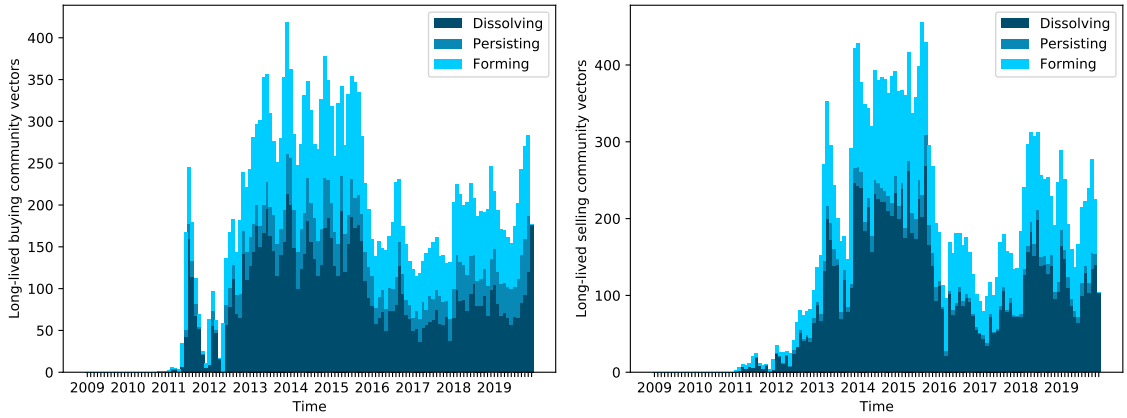
Table 5.3. *Two long-lived community evolution vectors.*

	Vector 1	Vector 2	Active vectors	Active communities
t_1	C_{11}	-	1	1
t_2	C_{21}	C_{22}	2	2
t_3	C_{31}	C_{31}	2	1
t_4	-	-	0	0

Of the two interpretations illustrated in Table 5.3, we will go forward with the latter one: counting unique communities under active vectors. Consequently, if we plot the number of active, long-lived vectors as a function of time, the sum of communities forming or staying alive at $t = t_1$ does not necessarily match the sum of communities staying alive

or dissolving at $t = t_2$. However, it would be at least equally weird to count a community twice at $t = t_1$ merely due to a split event at $t = t_1 + n$.

Figure 5.5 presents the amount of active, long-lived communities as a function of time. The figure considers community a long-lived one if it does not form and dissolve the same month. With such a relaxed threshold, the figure is dominated by forming and dissolving events. In other words, and as the color coding shows, around half of the active communities have formed that month and approximately similar amount of communities will dissolve the next month. Consequently, Figure 5.5 is very similarly shaped as Figure 5.4, which presents the distribution of survival events.

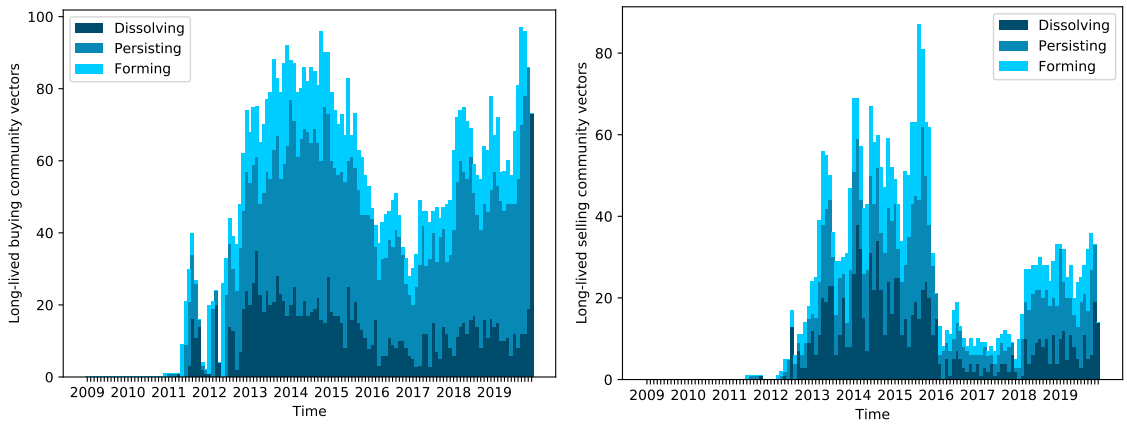


(a) *Buying* ***(b)*** *Selling*

Figure 5.5. *The number of long-lived community evolution vectors as a function of time. Every community living longer than 1 month is considered a long-lived one. The figure is dominated by two-month communities and, consequently, the shape follows the survival event distribution of Figure 5.4 very closely.*

As Figure 5.5 is so heavily dominated by two-month community evolution vectors, there is a clear motivation for plotting a similar graph but with the two-month communities excluded. If we focus only on those community vectors which live for three months or longer, the plot takes a slightly different shape. While the selling communities have a fairly similar shape as the buying ones in Figure 5.5, the three-or-more-months version shows that selling communities start to vanish with the stricter conditions, whereas the buying community graph still has a similar shape as earlier but it just covers a smaller area. Most months have around 50 to 80 active buying communities but the number of long-lived selling communities reaches similar heights only rarely. In addition, the selling communities never bounce back to the activity levels of year 2013—2015, even though the buying communities do. Figure 5.6 presents the number of active community evolution vectors, which live at least three months, as a function of time.

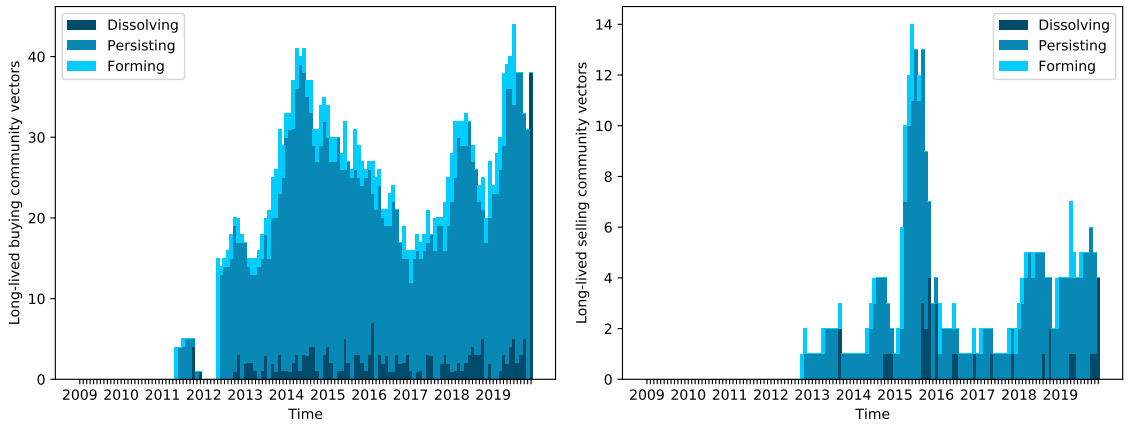
As we continue in a similar fashion and raise the threshold of lifespan length to six months, the results obtained by comparing the 3 and 6-month distributions are well aligned with the insights gathered by comparing 2 and 3-month distributions. Little remains of the selling



(a) *Buying* ***(b)*** *Selling*

Figure 5.6. *The number of long-lived community evolution vectors as a function of time, but with two-month vectors excluded.*

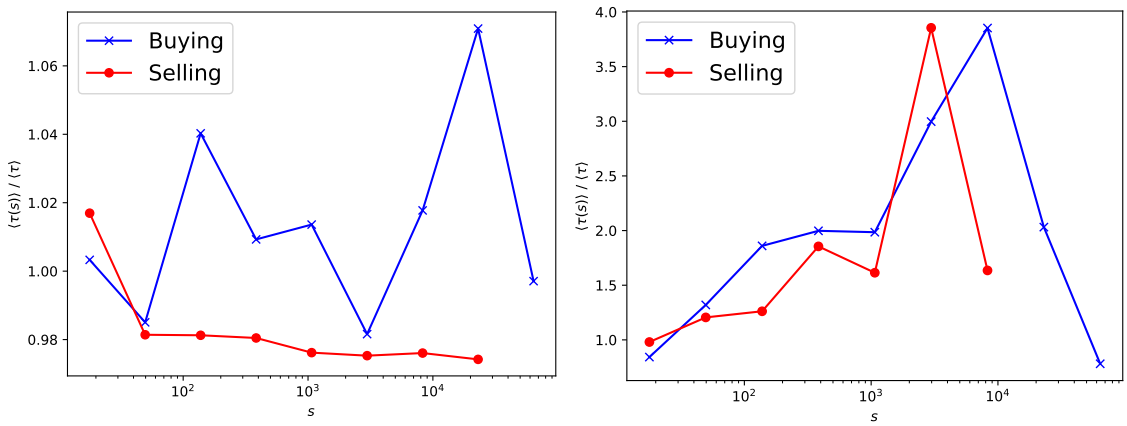
communities when the 6-month threshold is applied. Contrary to the selling communities, the stricter threshold does not affect the shape of the plot of buying communities. The amount of active communities remains quite stable from mid-2012 to the end of our data set. Figure 5.7 presents active community vectors, with a length of 6 or more months, as a function of time.



(a) *Buying* ***(b)*** *Selling*

Figure 5.7. *The number of long-lived community evolution vectors as a function of time, but with everything shorter than six months excluded.*

Another way to analyze the community lifespans is to carry out a similar test as Palla et al. (2007) do in their research. They compared the average lifespan of a community with a given size to the average lifespan of all communities. To be specific, they used the size of the first active month as the size of community. To perform a similar test, we pick each unique starting community and associate it with the longest community evolution vector. For example, if community C_1 splits into C_2 and C_3, and then only C_2 survives for another month, C_1 of size $|C_1|$ is considered to have a lifespan of 3 steps.



(a) *All communities*

(b) *Single-month lifespans excluded*

Figure 5.8. *The average lifespan for a given size $\langle \tau(s) \rangle$, compared to the average lifespan for all communities $\langle \tau \rangle$, as a function of community size s. Communities have been grouped by slicing the size range into ten buckets which are equidistant on a logarithmic scale.*

As our data consists of a wide range of community sizes, the validity of the test is improved by grouping the communities by size. This is done to avoid completely conflicting results for, let us say, communities of sizes 6126 and 6127. From our point of view, those are of identical size and, thus, the results would tell more about individual communities than the big picture if no grouping was done. Hence, we assign the communities to 10 buckets of the logarithmic space from 10 to 100 000. In other words, the bucket thresholds are equidistant on a logarithmic scale.

By performing the aforementioned test, we obtain a few new pieces of information. First of all, it seems that the community size has little to do with the expected lifespan. All observation points are within the range [0.97; 1.07] which means that no group has a significantly larger lifespan expectation than another. The result may be compared to the one obtained by Palla et al. (2007), who plot an increasing trend in which the higher values are in the region of 2.8. Simply put, in their study the larger communities have a tendency to live longer than the average, but the results can not be generalized to our Bitcoin investor communities.

However, when the same test is performed so that single-month communities are excluded from the data, it seems that size matters. Both buying and selling communities have higher values in the region from 1 000 to approximately 10 000 members. Figure 5.8 presents the results.

5.2 Similarity network snapshots

In addition to analyzing the evolution of communities, we can look at the results obtained while building the monthly similarity networks and detecting communities. For

each month, we have a network similar to the one presented in Figure 5.9 and we can, for example, analyze the number of active wallets and the distribution of similarity links between such wallets. Figure 5.9 also clarifies why network analysis relies on a wide range of algorithms. As the figure shows, the amount of links between communities is still significant even though the density is higher inside the community. Human eye and common sense are not able to perform community detection on such a granular level.

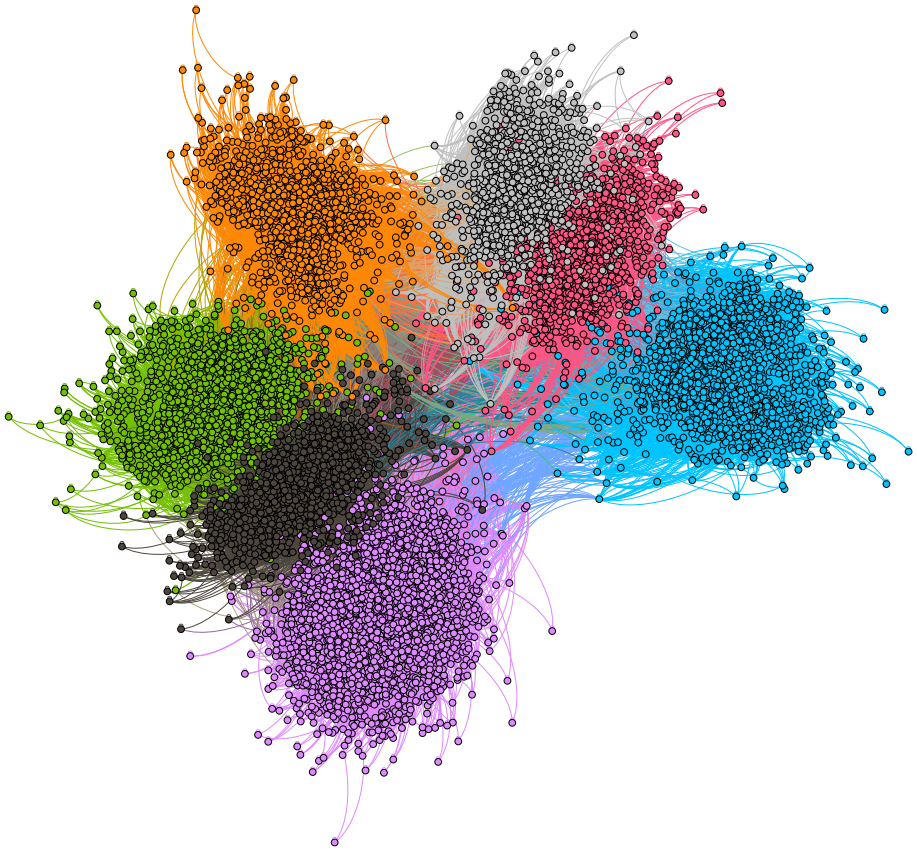
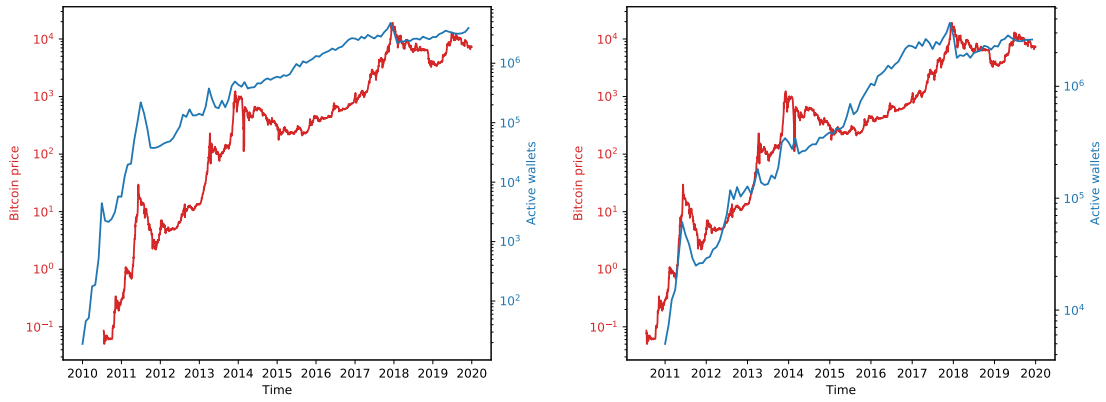


Figure 5.9. *Largest seller communities, May 2015. Infomap is able to detect multiple communities regardless of the inter-community links.*

If we merely look at the number of active wallets, buying and selling networks have a very similar shape. For both networks, the highest activity month is December 2017 when Bitcoin price reached its famous peak of almost $20k. Plotting the two series next to Bitcoin price series, both on a logarithmic scale, shows that the price series develops in a very similar fashion as the number of wallets does. That is actually a rather interesting observation as it is often said that Bitcoin price is not based on anything. Of course, it may still be that the speculative investors purely hope to get lucky, but at least there is a clear correlation between the number of active wallets and the Bitcoin price. Figure 5.10 presents the number of active wallets as a function of time.

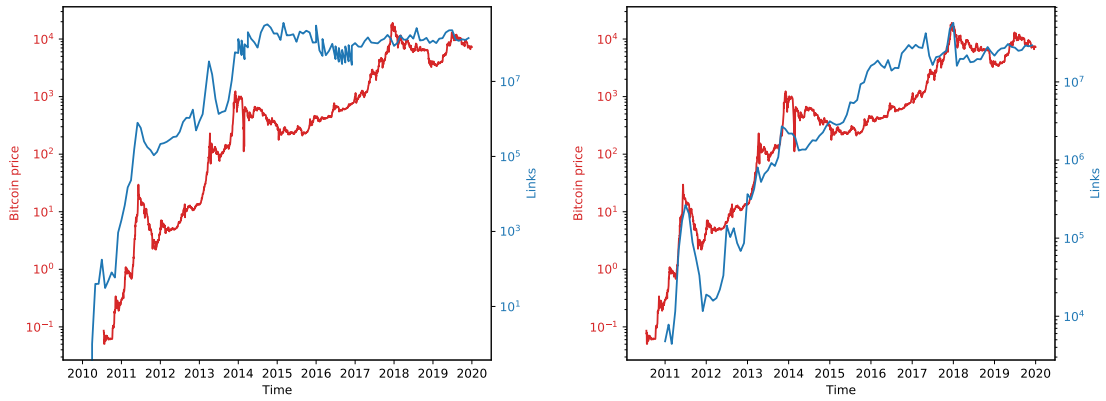
The other very basic property is the number of links, which here means similarity of two wallets when it comes to timing trades. The amount of links in buying networks peaks between 2014 and 2016, approximately the same span as the number of long-lived buying



(a) *Buying* ***(b)*** *Selling*

Figure 5.10. *Active wallets as a function of time. To be considered active, a wallet must be at least once (one hour) in buying or selling state*

communities reached its highest values. Figure 5.11 presents the number of similarity network links as a function of time.

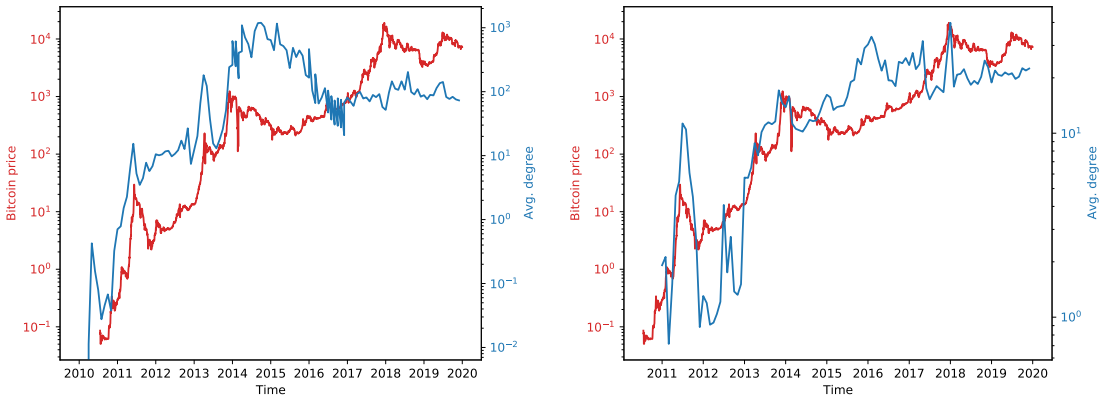


(a) *Buying* ***(b)*** *Selling*

Figure 5.11. *The number of similarity links in monthly similarity networks. The buying networks have their highest values from 2014 to 2016, approximately the same span as the number of long-lived buying communities peaked.*

Another way to look at the number of links is to calculate the average degree, which is basically the same as combining Figure 5.10 and Figure 5.11. The selling network snapshots have their highest average degrees in the region of 40 whereas the average degree in buying networks is frequently over 100, sometimes over 1 000. Figure 5.12 plots the average degree for each monthly snapshot.

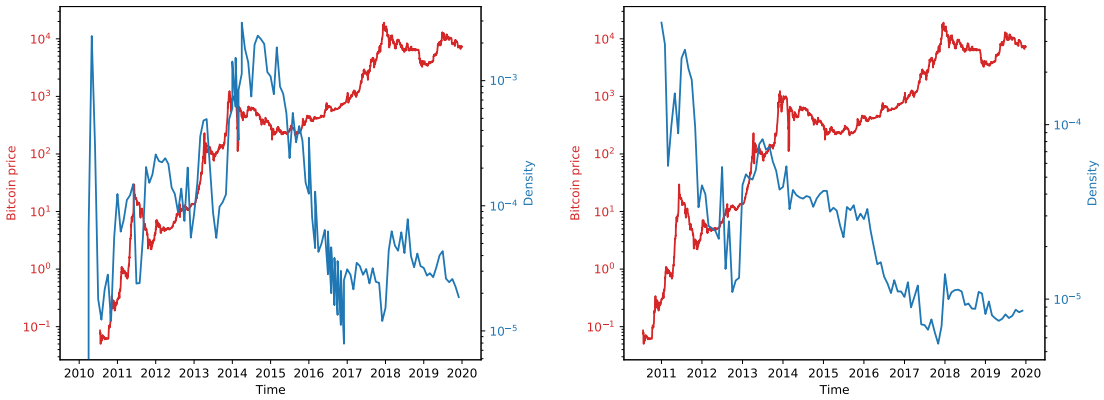
The actual number of links may also be compared to the theoretical maximum. In practice, the number of links is divided by $n * (n-1)/2$, where n is the number of nodes. This metric is known as density, and in this particular research, density also represents the relative amount of pairs passing the hypergeometric test while building the monthly similarity networks. Real-world networks tend to become sparser when growing (Newman 2018),



(a) *Buying* ***(b)*** *Selling*

Figure 5.12. *Average degree. Buyer network exceeds the threshold of 1 000 multiple times whereas the highest monthly averages are in the region of 40 in the seller network.*

and our similarity networks seem to follow that pattern to some extent. For the selling network snapshots, the trend is clearly declining. However, the density increases in the buying networks until mid-2015. Monthly network density is presented in Figure 5.13.

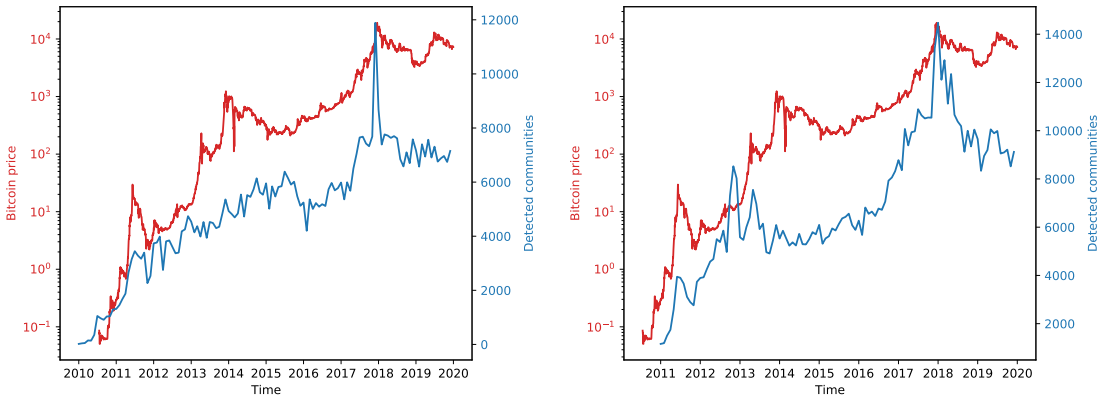


(a) *Buying* ***(b)*** *Selling*

Figure 5.13. *Network density as a function of time. The selling networks become sparser over time, but the density increases until the latter half of 2015 in the buying network.*

Even though we mostly focus on Bitcoin wallets, the nodes of our network, it should be remembered that the links determine the information flow in the network and, thus, are the basis of community detection. Therefore, they have a significant role in a research investigating community structure and the evolution of it. When it comes to the community detection performed for our monthly snapshots, the trend line seems to have a few local peaks but, other than that, it climbs quite steadily. Figure 5.14 shows the number of detected communities for each monthly snapshot.

It should be noted that the number of communities in Figure 5.14 are not necessarily aligned with the number of forming events in Figure 5.3. The main reason for that is the



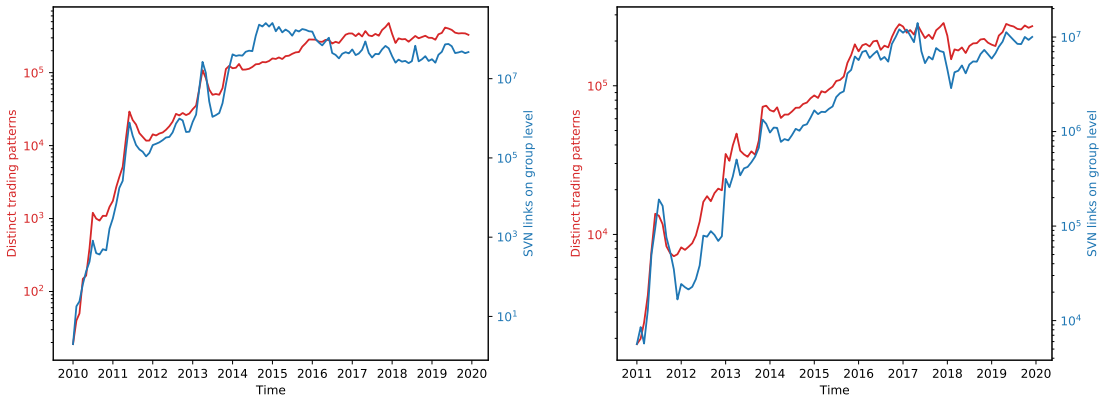
(a) *Buying* ***(b)*** *Selling*

Figure 5.14. *Detected communities*

decision to cut off communities with less than 10 members from the community evolution analysis.

The last piece of results to be presented is the number of distinct trading patterns – the wallet groups in pair-wise testing – per month. As Figure 5.10 shows, there are around 5 million active buying wallets in the busiest months. When we only look at unique trading patterns, the highest number of distinct buyer groups is slightly under 500 000. Basically the ten-fold difference means that the required CPU work would have been hundred-fold had we not grouped wallets for pair-wise testing.

If we look at the statistically validated links between wallet groups, the amount is very well aligned with the number of distinct trading patterns. Figure 5.15 shows the monthly number of distinct trading patterns and statistically validated similarity links between those patterns.



(a) *Buying* ***(b)*** *Selling*

Figure 5.15. *Pair-wise testing: wallet groups and statistically validated links between similar groups.*

The shape of the plot in Figure 5.15 basically indicates that the peaks in monthly average

degrees, presented in Figure 5.12, are resulted by wallets being unequally distributed to wallet groups. Some trading patterns are chosen by masses of traders and, even though there is only a couple of links on group level, there is a high-density cluster of wallets in the wallet-level similarity network.

6 CONCLUSIONS

With no prior research in place for analyzing the evolution of Bitcoin investor communities, this thesis penetrates the uncharted territory to provide novel information about Bitcoin as a social phenomenon. To conduct such a research, best practices of multiple disciplines are brought together. This paper extracts Bitcoin wallets from anonymous transaction data, builds a statistically validated network of Bitcoin users for each month and applies battle-tested network analysis tools on the created networks. What is more, the subsequent network snapshots are compared to extract events characterizing the evolution of dynamic communities.

The obtained results show that the vast majority of communities are short-lived but some communities survive for months, even years. We also find out that few selling communities persist for 6 months or longer whereas the corresponding number for buying communities is significantly higher, though still limited. When it comes to survival methods, communities prefer splitting over merging.

As this thesis presents some promising results regarding the underlying community structure of Bitcoin investor networks, the topic definitely deserves more attention. One possible route would be to further analyze the properties of long-lived communities. For example, while analyzing the investor clusters of the stock market, Baltakienė et al. (2019) perform a statistical test to find out if any investor attributes are overexpressed in the communities they detect. The anonymous nature of Bitcoin transactions undeniably complicates such a test but there are still some ways to enrich the data. For example, one could classify Bitcoin wallets with machine learning methods similarly as Ermilov et al. (2017). Another decent option would be to use off-chain data to bring a new dimension to the research as Meiklejohn et al. (2013) do.

It should also be remembered that this research constructs a multi-stage pipeline for transforming anonymous Bitcoin transaction data to dynamic Bitcoin investor community lifespans, which consist of a varying set of community members and events characterizing each step of the lifespan. The task is non-trivial and there are several points where one might decide to take a different approach. For example, the address-to-wallet mapping algorithm can always be improved. What is more, this thesis inspects monthly snapshots, and within those snapshots the timespan is sliced into hourly slots. The research could

even use a drastically different resolution as there is no single correct choice.

This paper also chooses to analyze buying and selling networks independently whereas, for example, Musciotto et al. (2016) construct a trading state vector so that the three individual state vectors – buying, selling and buying-and-selling – are concatenated and then the pair-wise testing is carried out for the concatenated trading state vectors. With our approach, it is possible to compare the structure of buying and selling networks but, on the other hand, we do not find out whether the synchronized buyers are also synchronized sellers later on. There are multiple cross-roads but one must choose a path.

REFERENCES

Androulaki, E., Karame, G. O., Roeschlin, M., Scherer, T. and Capkun, S. (2013). Evaluating user privacy in bitcoin. *International Conference on Financial Cryptography and Data Security*. Springer, 34–51.

Antonopoulos, A. (2017). *Mastering Bitcoin: Programming the Open Blockchain*. O'Reilly Media. ISBN: 9781491954348. URL: https://books.google.fi/books?id=tponDwAAQBAJ.

Asur, S., Parthasarathy, S. and Ucar, D. (2009). An event-based framework for characterizing the evolutionary behavior of interaction graphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3.4, 1–36.

Back, A. et al. (2002). Hashcash-a denial of service counter-measure.

Baltakiene, M., Baltakys, K., Cardamone, D., Parisi, F., Radicioni, T., Torricelli, M., Jeude, J. de and Saracco, F. (2018). Maximum entropy approach to link prediction in bipartite networks. *arXiv preprint arXiv:1805.04307*.

Baltakienė, M., Baltakys, K., Kanniainen, J., Pedreschi, D. and Lillo, F. (2019). Clusters of investors around initial public offering. *Palgrave Communications* 5.1, 1–14.

Baltakys, K., Baltakienė, M., Kärkkäinen, H. and Kanniainen, J. (2019). Neighbors matter: Geographical distance and trade timing in the stock market. *Finance Research Letters* 31.

Baltakys, K., Kanniainen, J. and Emmert-Streib, F. (2018). Multilayer aggregation with statistical validation: Application to investor networks. *Scientific reports* 8.1, 1–12.

Barabási, A.-L. et al. (2016). *Network science*. Cambridge university press.

Battiston, S., Puliga, M., Kaushik, R., Tasca, P. and Caldarelli, G. (2012). Debtrank: Too central to fail? financial networks, the fed and systemic risk. *Scientific reports* 2, 541.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* 57.1, 289–300.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R. and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008.10, P10008.

Bohlin, L., Edler, D., Lancichinetti, A. and Rosvall, M. (2014). Community detection and visualization of networks with the map equation framework. *Measuring scholarly impact*. Springer, 3–34.

Bohlin, L. and Rosvall, M. (2014). Stock portfolio structure of individual investors infers future trading behavior. *PloS one* 9.7, e103006.

Bonferroni, C. (1936). Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze* 8, 3–62.

Bovet, A., Campajola, C., Lazo, J. F., Mottes, F., Pozzana, I., Restocchi, V., Saggese, P., Vallarano, N., Squartini, T. and Tessone, C. J. (2018). Network-based indicators of Bitcoin bubbles. *arXiv preprint arXiv:1805.04460*.

Bovet, A., Campajola, C., Mottes, F., Restocchi, V., Vallarano, N., Squartini, T. and Tessone, C. J. (2019). The evolving liaisons between the transaction networks of Bitcoin and its price dynamics. *arXiv preprint arXiv:1907.03577*.

Bródka, P., Saganowski, S. and Kazienko, P. (2013). GED: the method for group evolution discovery in social networks. *Social Network Analysis and Mining* 3.1, 1–14.

Brown, S. D. (2016). Cryptocurrency and criminality: The Bitcoin opportunity. *The Police Journal* 89.4, 327–339.

Burks, L. S., Cox, A. E., Lakkaraju, K., Boyd, M. J. and Chan, E. (Aug. 2017). Bitcoin Address Classification.

Cabin, R. J. and Mitchell, R. J. (2000). To Bonferroni or not to Bonferroni: when and how are the questions. *Bulletin of the Ecological Society of America* 81.3, 246–248.

Chen, Z., Wilson, K. A., Jin, Y., Hendrix, W. and Samatova, N. F. (2010). Detecting and tracking community dynamics in evolutionary networks. *2010 IEEE International Conference on Data Mining Workshops*. IEEE, 318–327.

Coscia, M., Rossetti, G., Giannotti, F. and Pedreschi, D. (2012). Demon: a local-first discovery method for overlapping communities. *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 615–623.

Costa, L. d. F., Oliveira Jr, O. N., Travieso, G., Rodrigues, F. A., Villas Boas, P. R., Antiqueira, L., Viana, M. P. and Correa Rocha, L. E. (2011). Analyzing and modeling real-world phenomena with complex networks: a survey of applications. *Advances in Physics* 60.3, 329–412.

Dai, W. (1998). B-money. *Consulted* 1, 2012.

Dakiche, N., Tayeb, F. B.-S., Slimani, Y. and Benatchba, K. (2019). Tracking community evolution in social networks: A survey. *Information Processing & Management* 56.3, 1084–1102.

Economist, T. (2018). *Why bitcoin uses so much energy*. https://www.economist.com/the-economist-explains/2018/07/09/why-bitcoin-uses-so-much-energy. Accessed: 13.7.2020.

Emmert-Streib, F., Musa, A., Baltakys, K., Kanniainen, J., Tripathi, S., Yli-Harja, O., Jodlbauer, H. and Dehmer, M. (2018). Computational Analysis of the structural properties of Economic and Financial Networks. *Journal of Network Theory in Finance* 4.3, 1–32.

Ermilov, D., Panov, M. and Yanovich, Y. (2017). Automatic bitcoin address clustering. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 461–466.

Feistel, H. (1973). Cryptography and computer privacy. *Scientific american* 228.5, 15–23.

Greene, D., Doyle, D. and Cunningham, P. (2010). Tracking the evolution of communities in dynamic social networks. *2010 international conference on advances in social networks analysis and mining*. IEEE, 176–183.

Haber, S. and Stornetta, W. S. (1990). How to time-stamp a digital document. *Conference on the Theory and Application of Cryptography*. Springer, 437–455.

Harrigan, M. and Fretter, C. (2016). The unreasonable effectiveness of address clustering. *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*. IEEE, 368–373.

Hopcroft, J., Khan, O., Kulis, B. and Selman, B. (2004). Tracking evolving communities in large linked networks. *Proceedings of the National Academy of Sciences* 101.suppl 1, 5249–5253.

Investopedia (2019). *Currency*. https://www.investopedia.com/terms/c/currency.asp. Accessed: 9.7.2020.

– (2020). *Bitcoin's Price History*. https://www.investopedia.com/articles/forex/121815/bitcoins-price-history.asp. Accessed: 26.10.2020.

Lewis, A. (2018). *The basics of bitcoins and blockchains: an introduction to cryptocurrencies and the technology that powers them*. Mango Media Inc.

Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G. M. and Savage, S. (2013). A fistful of bitcoins: characterizing payments among men with no names. *Proceedings of the 2013 conference on Internet measurement conference*, 127–140.

Moran, M. D. (2003). Arguments for rejecting the sequential Bonferroni in ecological studies. *Oikos* 100.2, 403–405.

Musciotto, F., Marotta, L., Miccichè, S., Piilo, J. and Mantegna, R. N. (2016). Patterns of trading profiles at the Nordic Stock Exchange. A correlation-based approach. *Chaos, Solitons & Fractals* 88, 267–278.

Nakamoto, S. et al. (2008). *Bitcoin: A peer-to-peer electronic cash system.(2008)*.

– (2009). *Bitcoin Core*. `https://github.com/bitcoin/bitcoin`. Accessed: 15.7.2020.

Narayanan, A., Bonneau, J., Felten, E., Miller, A. and Goldfeder, S. (2016). *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press.

Newman, M. (2018). *Networks*. Oxford university press.

Palla, G., Barabási, A.-L. and Vicsek, T. (2007). Quantifying social group evolution. *Nature* 446.7136, 664–667.

Perneger, T. V. (1998). What's wrong with Bonferroni adjustments. *Bmj* 316.7139, 1236–1238.

Preneel, B. (1993). Analysis and design of cryptographic hash functions. PhD thesis. Katholieke Universiteit te Leuven.

Rice, J. A. (2006). *Mathematical statistics and data analysis*. Cengage Learning.

Ron, D. and Shamir, A. (2013). Quantitative analysis of the full bitcoin transaction graph. *International Conference on Financial Cryptography and Data Security*. Springer, 6–24.

Rosvall, M., Axelsson, D. and Bergstrom, C. T. (2009). The map equation. *The European Physical Journal Special Topics* 178.1, 13–23.

Siikanen, M., Baltakys, K., Kanniainen, J., Vatrapu, R., Mukkamala, R. and Hussain, A. (2018). Facebook drives behavior of passive households in stock markets. *Finance Research Letters* 27, 208–213.

Squartini, T., Van Lelyveld, I. and Garlaschelli, D. (2013). Early-warning signals of topological collapse in interbank networks. *Scientific reports* 3, 3357.

Takaffoli, M., Sangi, F., Fagnan, J. and Zaıane, O. (2010). A framework for analyzing dynamic social networks. *Applications of Social network Analysis (ASNA)*.

Tasca, P., Hayes, A. and Liu, S. (2018). The evolution of the bitcoin economy. *The Journal of Risk Finance*.

Tumminello, M., Micciche, S., Lillo, F., Piilo, J. and Mantegna, R. N. (2011). Statistically validated networks in bipartite complex systems. *PloS one* 6.3, e17994.

Vallarano, N., Tessone, C. and Squartini, T. (2020). Bitcoin Transaction Networks: an overview of recent results. *arXiv preprint arXiv:2005.00114*.

Vigna, P. and Casey, M. J. (2016). *The age of cryptocurrency: how bitcoin and the blockchain are challenging the global economic order*. Macmillan.