UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Faculty of Science and Technology
Department of Computer Science

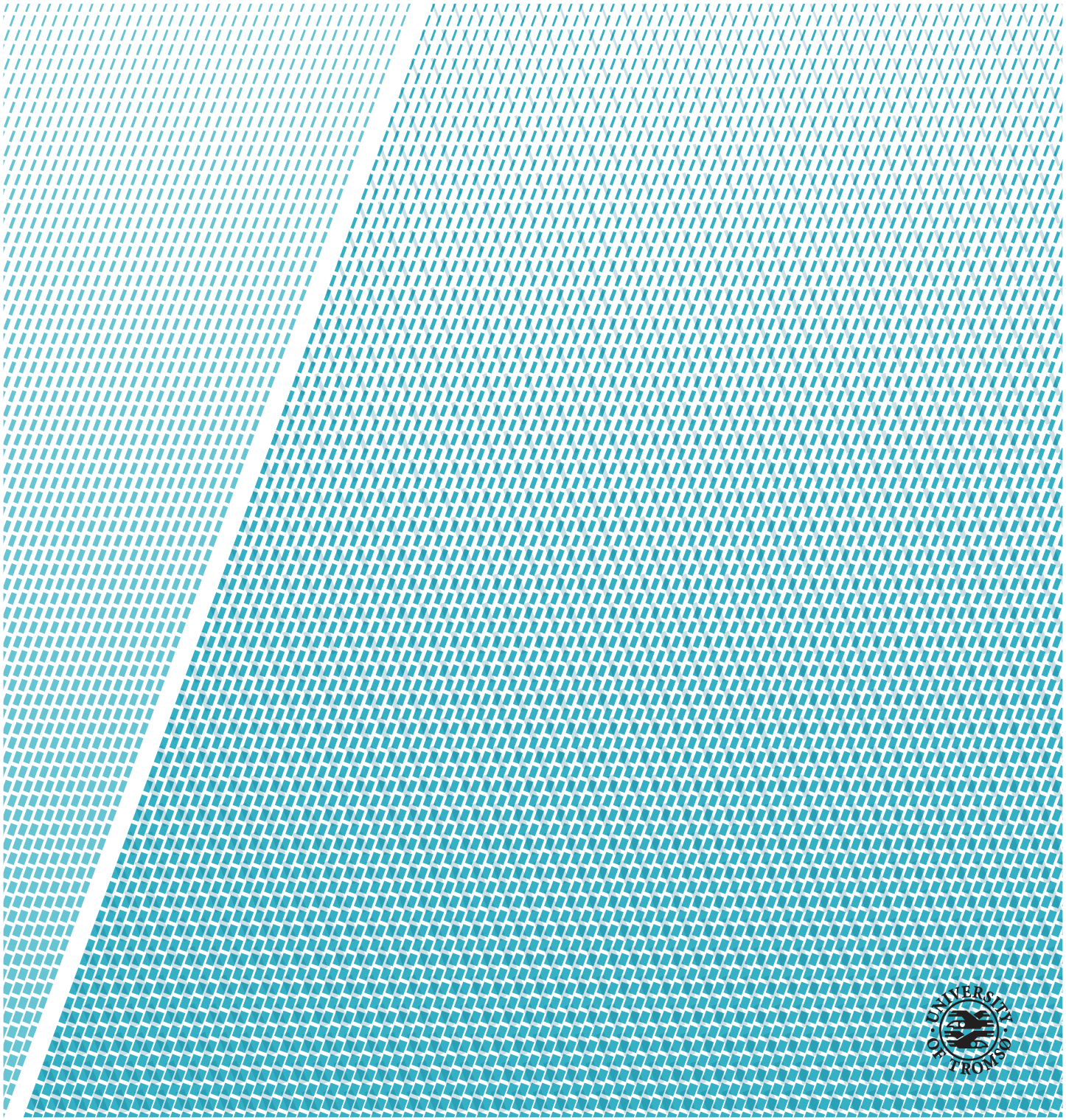
Trading Network Performance for Cash in the Bitcoin Blockchain

A Longitudinal Study on Bitcoin's Scalability, Performance, and Costs

—

Enrico Tedeschi

INF-3990 Master Thesis in Computer Science - November 2017



Acknowledgements

I would like to express my very great appreciation to Håvard D. Johansen for supervising my thesis and for his dedication and help he gave me over this year. His devotion and love for his job made everything much easier, and every time we had a talk, he came up with new brilliant ideas and shrewd solutions to our problems that totally changed my point of view and made me ponder and contemplate. I also thank him for introducing me in the Academia world and into the iAD research group at UiT.

I want to also thank Professor Dag Johansen for being such a good motivator and the advice given by him has been vital in building this thesis and made me more eager to learn about the topic after every discussion. His capacity to adapt and find solutions for any kind of unexpected changes is admirable, I've been so inspired by this.

My special thanks are extended to the iDA group which included me from the beginning in their community and helped me to socialize in a foreign country.

And finally, I would love to thank my parents that always supported my idea of studying abroad and motivated me mentally and economically, my girlfriend, my brother and my best friends that were always by my side, and at the end the city of Tromsø that gave me a pleasant and warm welcome, by allowing me to study at UiT and work as a northern lights photographer.

All of you have a share in this work, thank you!

Abstract

This thesis describes a longitudinal study of Bitcoin, the perhaps most popular blockchain based system today. Public blockchains have emerged as a plausible messaging substrate for applications that require highly reliable communication. However, sending messages over existing blockchains can be cumbersome and costly as miners require payment to establish consensus on the sequence of messages, since the electricity consumption needed to run miners is not negligible. The blockchain protocol requires an always growing size of the information stored in it so its *scalability* is the biggest problem. For that reason we decided to collect and store data locally in our own data structure, necessary for the analysis, allowing us to save up to 10 times the amount of disk space. Today, systems using the blockchain protocol are emerging, and cryptocurrencies are a glaring example of its implementation. Bitcoin represents the largest cryptocurrency on market, and it has to face a massive scale due to its popularity, having in 2012 about fifty thousands transaction per day and reaching now, in 2017, more than three hundred fifty thousands of transactions approved every day.

This massive scale in the system leads to a saturation of the messaging substrate, hence performance issues. In this thesis we will focus also on the Bitcoin network performance, in particular, transaction throughput and latency. From 2009 to 2017 a lot of analyses on the blockchain have been performed, enhancing the considerable change in the block size limit, from 256 bytes to 1 MB, as an attempt to overcome scalability problems. Different papers were published, discussing whether changing or not the block size limit. In addition, the Bitcoin price increased from $\sim 0.7\,\$$ to more than $7.000\,\$$, making the system even more desirable for miners, but causing several complications in the fee and reward mechanism. We evaluate and discuss possible ways to improve this fee mechanism in order to guarantee more revenue for miners along with an user fee optimization.

We finally present our own system for longitudinal analysis on the Bitcoin blockchain, BAS. It generates a dataset which contains a significant portion of the whole blockchain, updated on September 2017. We discuss our results and compare them with other evaluations from past years, considering three main

key points: *scalability*, *performance* and *fees/costs*. We discuss how scalability affects performance, and how the costs and fees are dependent from them both. We want also to take into consideration the environmental impact of Bitcoin and how it affects the coming of new cryptocurrencies. We evaluate and propose, using machine learning techniques, two different cost prediction models that aim to predict bandwidth for upcoming transactions according the fee they are willing to pay, and the expected revenue for miners according to the time spent mining. These models can be used by application to throttle network traffic to optimize message delivery. We also discuss whether the block size limit should be increased for a higher *throughput* or not.

Contents

List of Figures

List of Tables

My list of definitions

1

Introduction

In 1964, Paul Baran [8] described the differences between a centralized, decentralized and distributed network. Since then, the attention in developing systems moved from a centralized scheme to a distributed one, leaving most of the computation to every single processor in the network rather than a central coordinator. Such a change might be easy for systems that require little security, for instance systems that not do require authentication and authorization. However, the more secure a system needs to be, the more the decentralization process might be tricky as it becomes very important to rely on some trusted central coordinator. Systems requiring stronger security are the ones that are related to e-commerce, banking, and trade; essentially all systems that have to deal with money.

In 1983, David Chaum introduced the idea of digital cash [13]. In 1990 he founded *DigiCash*, an electronic cash company that closed due to bankruptcy in 1998. After DigiCash, other systems such as *e-gold* (1996) and *PayPal* (1998) emerged. However, systems for digital currency allowed digital money transfer while they were still relying on a central authority, hence, they were centralized or decentralized but not distributed. In 2008 Satoshi Nakatomo presented Bitcoin [35], the first decentralized digital currency. Until 2008 e-commerce relied exclusively on financial institutions serving as trusted third parties. Those are involved in the electronic payments process and they have to guarantee consistency of the transactions and security of data.

Decentralized digital currencies are not dependent on any trusted third parties

and they are built over a Peer to Peer (P2P) network in which every component has the same privileges. These systems allow money exchange without a central authority, which means lower fees, no geographical separation and global trust among users. After Bitcoin, more decentralized digital currencies emerged, in 2011 *Litecoin*, originally based on the bitcoin protocol, then in 2013 Gavin Wood presented *Ethereum* [53] and in 2014 *Monero* currency was released.

The transaction ordering is essential in any currency systems, however, establishing correct order can be problematic in decentralized cryptocurrency systems as they allow arbitrary nodes to join, including nodes that might be malicious. If arbitrary or Byzantine faults are allowed, the system might be left in an inconsistent or invalid state [29]. The ability to mask Byzantine faults has been implemented in various systems such as Byzantium [18], HRDB [49] and MITRA [30]. These protocols guarantee consistency of transactions having f faulty nodes, with a total of N nodes where $N = 2f + 1$ or $N = 3f + 1$. Fireflies [24] is perhaps the only Byzantine-fault tolerant membership protocol that can support overlay-network sizes of the current Bitcoin network. The protocol has also been demonstrated efficient in distributing large block of date. However, the suitability of Byzantine consensus to support cryptocash has yet to be demonstrated.

1.1 Blockchains

To guarantee the ordering of transaction all these cryptocurrencies rely on the *blockchain* protocol. The need to tolerate malicious members was the reason for introducing the blockchain into cryptocurrency systems. A blockchain is essentially a distributed database of records or public ledger of all transactions or digital events that have been executed and shared among participating parties [15]. The fundamental principle behind the blockchain is that consensus on transaction ordering is based on contributed computational power rather than number of participants. The blockchain works by appending transactions into blocks. Every block is generated after a relevant computation, called *proof-of-work*, and each new block is appended to the public ledger of data, the blockchain, having in that way an ever growing chain of information containing every transaction ever happened. Blockchains essentially implements a distributed consensus protocols that enable a set of untrusted processes to agree on the content of an append only data structure, often referred to as ledgers. These ledgers are divided into blocks and linked together in sequence by hashes. They facilitate transactions between consenting individuals who would otherwise have no means to trust each other and deal with geographical separation and interfacing difficulties. This technology promises a highly

resilient and communication substrate where messages are potentially kept for a long time.

Besides its use in cryptocurrency, blockchain technology opens up to several usages in different sectors as a basis for a tamper proof audit ledger in eHealth systems [19], or to execute decentralized data policies [23, 48], in addition, it is already used by NASDAQ in its private socket market. If used in a P2P file sharing network, blockchain removes the need of a centralized data base. Moreover it allows users to create tamper-proof digital identities for themselves and it opens up to usages in several important sectors such as trading, file storage, and identity management.

There are several limitations in current blockchain technologies. The most relevant is *scalability*, due to the steady growth of the ledger of data. It should be also considered that decentralized cryptocurrencies operate in open (or permissions less) networks in which the ledger of data could be manipulated from arbitrary adversaries. With the emerging of cryptocurrencies, smart contracts came out and they aim to facilitate, verify, or enforce the negotiation or performance of a contract [44]. According to Luu et al., 2016 [31], security of smart contracts has not received much attention yet. Since the only part not protected by cryptography is the *order of transactions* [11], an attacker could try to convince the network that a transaction occurred earlier than another one to gain money. The security bugs in smart contracts are classified as *Transaction-Ordering Dependence*, *Timestamp Dependence*, *Mishandled Exceptions* and *Reentrancy Vulnerability* [31]. In this thesis we will not focus on smart contracts but it is useful to know that in early 2017 RootStock (RSK) project was launched. *RootStock* is a smart contract P2P platform built on top of the Bitcoin blockchain and they are also working on Bitcoin scalability [28]. In this thesis we refrain from explaining Bitcoin and its terminology in detail, while we refer to previous works, studies or technical papers from Underwood and Böhme et al.[45, 12] or from Nakamoto and Wood [35, 11].

1.2 Problem Statement

The most urgent concerns in the blockchains are related to scalability, performance and a profit optimization for the participant in the system. In 2015, Möser and Böhme [34] said that Bitcoin may not be as cheap for consumers as it appears, and that Bitcoin users are encouraged to pay fees to miners up to 10 United States Dollar (USD)cents, per transaction, irrespectively to the amount paid. Already in 2015, Rizun [40] wrote that the block size limit was set at one MB, corresponding roughly to three transactions per second. Today the transaction rate is over three hundred times larger than when the block size

limit was introduced, hence increasing the block size limit is now being seriously considered. In 2016 Croman et al. [14] announced that the current trend of increasing the block sizes on Bitcoin portends a potential problem where the system will reach its maximum capacity to clear transactions, probably by 2017. These problems need to be taken into consideration.

In this thesis we propose a longitudinal study on Bitcoin blockchain, analyzing most recent data. In particular, we discuss the scalability of the blockchain, how it affects the throughput, and we present performance observations of the Bitcoin blockchain, analyzed with a blockchain analytic system developed for this purpose. We provide detailed insights and analysis of how the characteristics of Bitcoin, such as fee, block size, reward to different miners have changed over time, and provide an updated view from the one proposed by Möser and Böhme [34]. Furthermore, we analyze the correlation between the fee paid from a transaction and its *latency*, or the time it takes to become visible in the whole network. Three different models are proposed to describe how applications should spend money to improve network performance, this affects average bandwidth available to an application. By doing transaction-wise and block-wise experiments and analyses on the Bitcoin blockchain we state and focus on three major problems:

1. Scalability
2. Performance
3. Costs/Fees

1.2.1 Scalability

Scalability and network performances are urgent concerns in existing blockchain-based cryptocurrencies [14]. According to Buterin et al. [11], if Bitcoin would have the same amount of transactions as a VISA circuit, its blockchain would grow about 1 MB every 3 seconds, ~ 28 gigabyte (GB) per day, instead of the actual growth of ~ 0.12 GB per day. In this thesis we discuss how much scalability affects centralization in the Bitcoin network and how its impact will be in a couple of years. Furthermore, with our analysis we aim to study how this growth is affecting the system's performance and costs, and how much they change if the system capacity is filled with too many transactions to approve.

1.2.2 Performance

Centralized schemes, like VISA, are immediate while having a throughput of two thousand transactions per second up to fifty-six thousand transactions per

second [14]. It is true that Bitcoin has lower fees than centralized currency schemes, but these properties come at a performance and scalability costs. In the paper from Croman et al. [14], they claim that Bitcoin achieve a throughput of 7 transactions per second constrained by the block creation time and the block size limit. In this thesis we want to analyze the network performance with new data from 2017 with regards to throughput and transactional latency to see how much a possible change in the block size or block creation time might influence the efficiency of the system.

1.2.3 Costs/Fees

Bitcoin's strength has always been its lower fees if compared to centralized systems. We aim to find out whether this it still true today or not, if the fees are still low and the system still that cheap. The actual costs of the system have not been extensively studied yet and Bitcoin might not be as cheap for costumers as it appears. During our analysis we could observe a remarkable increase of the fee paid from users. In this thesis we aim to find and prove the reasons that leaded to this scenario and they involve also scalability and performance problems. Our first concern while studying fees and tolls is to analyze whether is possible or not for applications to control available bandwidth given from the system by paying higher fees to miners.

1.3 Method / Context

To analyze performance, make assumptions on scalability and fees in the Bitcoin network, data need to be retrieved from a source. We define three different approaches of data retrieval and all three have their advantages and disadvantages:

1. **Real time analysis:** It might be done by sniffing the traffic on the Bitcoin network in order to get real time data. For this analysis at least one full operating node needs to be implemented in the Bitcoin network. Advantages are that, if you set two or more nodes distributed around the world, you can get a lot of useful information regarding the inner-node communication (otherwise impossible to obtain) and the block propagation time plus the orphaning rate in the system. Disadvantages are that you need a relevant disk space to download the full ledger of data and the set up it extremely time consuming, plus, you have physical and geographical limitations in order to get up and running multiple nodes in different part of the world, without considering bandwidth restrictions and electricity costs related to it.

2. **Historical data retrieval through Bitcoin Core:** Another solution might be to use the client app of Bitcoin, Bitcoin Core [46]. You can set up your node containing the full ledger locally, storing all the necessary data and have a historical view of the entire blockchain. Advantages are that you have the whole view of the system. Disadvantages are that, this whole view, still does not include the propagation time nor any miner's information, having to retrieve these knowledge separately. Plus, to set up the full node might take up to 4 days and the disk space required is in the order of hundreds of GB.

3. **Historical data retrieval through Application Programming Interface (API):** Third choice is to build a system locally, and use Bitcoin APIs to gather data and generate information using those. Advantages are that you have more flexibility when retrieving data, allowing you to store only useful information, saving a considerable disk space, also, the blockchain might be stored in intervals, which gives a faster retrieval, more disk space saved and still provides enough information to have a whole view of the system. Disadvantages are that still you don't have information about block propagation time and orphaning rate, and it might be difficult to analyze the blockchain in real time.

In this thesis we use method 3, as this enables us to analyze a considerable part of the blockchain with little up-front investment in computational resources. A similar methodology was also adapted in the 2015 study by Möser and Böhme [34], analyzing tips and tolls in the Bitcoin blockchain, collecting data until 2014 and then analyzing more than nine million of transactions. Since then, the rate of transactions have risen considerably from one hundred thousand per day to three hundred fifty thousand per day, so the retrieving part turned out to be more time consuming than expected. Despite that, we aim to collect even a larger portion of the blockchain, storing data smartly in a *data frame*, which allows us to save up to 10 times the amount of disk space the blockchain actually requires. Then we analyze our collected data and with *machine learning* techniques we define models, discuss about the results and how much they can be reliable in a future-wise implementation. In our data structure we store more than one hundred twenty million of transactions occurred between April 2013 and September 2017. We used for the information retrieval, APIs from `blockchain.info`, combined with a HyperText Markup Language (HTML) parsing on the same website for the information missing while retrieving data with Bitcoin's APIs. Our assumption is that we can get enough information about the Bitcoin blockchain by retrieving and analyzing only a portion of the blockchain, but having in that way a finer granularity than data represented on the Bitcoin website. In that way we aim to gain more information out of it. Moreover, sampling data from a single node in the blockchain gives statistics representative of the whole system.

1.4 Outline

This chapter gives a briefly introduction about digital currencies, centralized and decentralized systems, and it talks about the emerging of cryptocurrencies. It also focuses to give a short introduction to the blockchain protocol and why it is so important in systems such as Bitcoin. Chapter 2 helps the reader to understand decentralized cryptocurrencies, focusing on mining and proof-of-work concepts. It also gives an overall view about previous works and analyses on the Bitcoin blockchain, and it shortly introduce the reader to machine learning techniques about big data management. Chapter 3 talks about the system we have developed for the blockchain analysis, it illustrates its architecture and structure regarding data retrieval, manipulation and visualization. Furthermore, assumptions that need to be proved in this thesis are defined. In Chapter 4 we show our results. We explain the generated plots and prove our assumptions, thereby we can discuss possible solutions of the problems raised. Finally, Chapter 5 presents the conclusions and discussions of the overall work, leading to the future implementations of our system, further analyses, and studies of Bitcoin.

2

Background

This chapter gives an introduction to blockchain systems, with a particular focus on Bitcoin. We will briefly explain the concept of mining and introduce the notion of payments and fees in the Bitcoin system. We will describe how miners profit from mining, how a user could choose whether including fee or not in its transactions, and how performance and scalability are changing the way miners select transactions. We will also describe some machine learning techniques and data structures, plus we will take into consideration the most relevant studies and works that has been done in the Bitcoin system regarding scalability, performance and fees. The list of symbols used is shown in Appendix B. For further detail we refer to already existing high-level [12] or technical [53, 32, 33, 40, 35, 11] descriptions.

Differences between *centralized* and *decentralized* digital currencies are summarized in Table 2.1. As mentioned in Chapter 1, more decentralized digital currencies are emerging nowadays, and they aim to facilitate transactions between consenting individuals who would otherwise have no means to trust each other and deal with geographical separation and interfacing difficulties. According to Nakamoto [35] an electronic coin is defined as follows:

The Definition 1. An *eletronic coin* is a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin.

Table 2.1: Centralized vs Decentralized digital currencies

Centralized	**Decentralized**
Under the control of a central authority	Requires consensus among users to make changes
Users depend on the control given from the Certificate Authority (CA)	Users have full control over their assets/data
No need to enforce cryptography or to create keys	User's data is uniquely identified by their private key
Only users allowed by CA can participate	Anyone can join and use the ledger
Historic transactions can be changed by the CA	Historic transactions are unalterable and permanent
Can be run very efficiently using relational databases that fit the need of the applications	Inefficient for the cost of mining

Transactions are registered in *blocks*, and all these systems use the *blockchain* technology to maintain a serial order of them. The blockchain is a consensus protocol that users run to maintain and secure a shared ledger of data and it is formed of all blocks, ordered by time and linked between each other with the *previous block* attribute. A transaction is approved only when it is included in a block, and the miner including this transaction is the one who has first solved the *proof-of-work*. Difficulty of proof-of-work is increased according to keep an average $\mathcal{T}$ of 10 minutes.

2.1 Mining

In a decentralized cryptocurrency network, transactions need to be approved by all the participant before being validated and confirmed. This validation requires high computational time and it is expensive in matter of electricity and resources used.

The Definition 2. *Mining* is how transactions are validated and confirmed by the network.

To validate and confirm transactions, the network needs *miners*. According to the Bitcoin miner document [4], Bitcoin miners create new blocks from solving a proof-of-work problem that is chained through cryptographic proof of the previous block. The work and the effort spent to create new blocks is often referred to as mining [4]. The mining process involves identifying a value that

when hashed twice with SHA-256, begins with a number of zero bits. If the amount of zeros needed for the creation is raised, the average work required increases exponentially, while a hash can always be verified by executing a single round of double SHA-256 [4, 42]. Each miner chooses whether to include a transaction into a block or not, and we believe that nowadays this inclusion strongly depends on the fee that this transaction has to offer. The first miner who solves the proof-of-work gains the sum of the fees (t_f) offered by each transaction included in the block, M, plus the block reward, R, which is now set to 12.5 ₿. When a new block is mined, that information is spread through the whole network and if the solution is propagated to $50\% + 1$ of the nodes in the system, the block is validated and all the transactions in it are accepted and confirmed. Finally, miners in the network express their acceptance by starting to work on the next block, incorporating the hash of the newly accepted one.

2.1.1 Proof-of-Work

Karpelès et al. define the proof-of-work as follows [25]:

The Definition 3. *Proof-of-work* is a piece of data which is difficult (costly and time-consuming) to produce but easy for others to verify and which satisfies certain requirements.

Mining a block is difficult because the SHA-256 [42] hash of a block's header must be lower or equal to a certain *target value* in order for the block to be accepted by the network. Producing proof-of-work is a random process with low probability of hitting the target value, so that a lot of trial and error is required on average before a valid proof-of-work is generated. For example, Hashcash is a proof-of-work system used to limit email spam and denial-of-service attack and recently has become known for its use in Bitcoin and other cryptocurrencies as part of the mining algorithm [7].

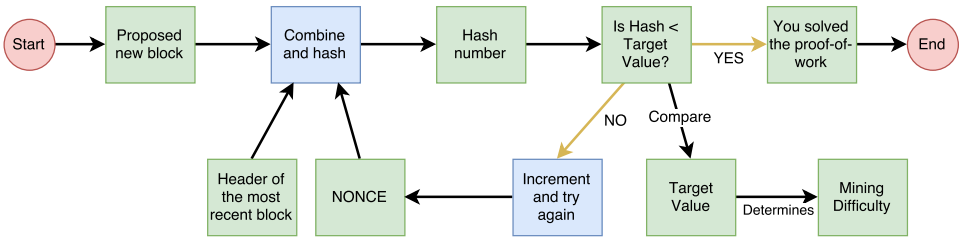


Figure 2.1: Diagram that shows how proof-of-work works.

Proof-of-work is a part of the mining process and a valid proof-of-work is determined by incrementing a nonce until a value is found that gives the block's

hash the required number of leading zero bits. Once the hashing has produced a valid result, the block is immutable and cannot be changed without rebuilding a chain bigger than the original. As later blocks are appended to the chain, the work to change the block would include redoing the work for each subsequent block [4]. The best chain is the longest one, so is the one that requires the greatest effort to produce, and majority consensus in Bitcoin is represented by this chain. In this way if the majority of the computing power is controlled by honest nodes, the honest chain will grow fastest and outpace any competing chains. Diagram in Figure 2.1 represents how the proof-of-work is performed, and it is inspired to the model presented in `bitcoinmining.com` [32].

2.1.2 Miners & Mining Pools

With paper money, a government decides when to print and distribute money. Bitcoin does not have a central government and it uses a special software to solve math problems to get a certain amount of coins in exchange. This provides a smart way to issue the currency.

The Definition 4. A *miner* is a computer specifically designed to solve problems according to the proof-of-work algorithm and they are required to approve transactions.

Since miners are required to approve transactions, more miners mean a more secure network. To become a Bitcoin miner nowadays, you need to buy highly specified chips called Application Specific Integrated Circuits (ASIC). Information about this hardware can be found at `bitcoinmining.com` [32]. In the early days of Bitcoin (2009-2013) it was possible to mine with your commodity hardware or high speed video processor card. Today that is no longer possible since custom Bitcoin ASIC chips offer performance up to 100 times the capability of older systems [32]. Another reason is that mining pools became more prevalent, yielding a constant growth of the *mining difficulty*.

The Definition 5. A *pooled mining* combines the work of many miners toward a common goal.

Pools of miners or mining pools find solutions faster than individual members and each miner is rewarded proportionally with the amount of work it provides. Mining is an important and integral part of Bitcoin that ensures *fairness* while keeping the Bitcoin network *stable*, *safe* and *secure* [32].

2.2 Reward & Fee

Miners generate new blocks by appending transactions to them. Every miner has a *mempool*, $\mathcal{N}$, containing the new, unapproved, n transactions, and they get compensated for mining with a revenue $\langle V \rangle = R + M$, where M is the sum of transaction fees in the block and R is the block reward, which is now set at 12.5 ₿. Miners are free to choose whether to accept or refuse a certain transaction $t \in \mathcal{N}$. Competitive miners will include transactions as long as the fee exceeds the marginal cost of inclusion. Production costs are fixed per block (but may vary between miners depending on access to technology and energy cooling) and the protocol defines a maximum block size Q. Because of that, the marginal cost of inclusion is zero if there are fewer unconfirmed transactions than the capacity left in the block. Competitive miners make positive expected profits only if transactions compete for space in the blockchain. Houy [21] argues that a maximum block size is necessary for the stability of Bitcoin. However in that way big mining pools might take advantage from less competition, having a centralization problem, since a transaction is willing to compete only if the capacity is reached. In that way if the system never reaches the capacity, raising the fee will not be helpful at all. Before 2015, transactions almost never required to compete for space in blocks, since the demand was less then the offer, but nowadays we have a totally different scenario. In 2012, the number of transactions confirmed per day reached peaks of fifty thousand, with an average of fifteen thousand transactions per day [33]. Today we have an average of two hundred fifty thousand transactions approved every day, with peaks of three hundred and fifty thousand. Because of throughput limitations in the system, caused by the block size limit, Q, set at 1 MB and the average block creation time, $\mathcal{T}$, set at 10 minutes, with such big scale of transactions that need to be approved every day, miners could start to choose transactions with a higher fee density ρ, or simply transactions that are willing to offer a higher fee rather than 0-fee transactions. The fact that the reward R has a 50% reduction every two hundred thousand blocks creates the need of a market mechanism to find the price of Bitcoin transactions.

2.3 Previous Works

The main problems analyzed in the past years concerning the Bitcoin blockchain were related to scalability [40, 14], performance [14, 17] and costs/fees [40, 34]. Already in 2014 researchers believed that these "fees" users were paying to miners were supposed to substitute miners' minting reward in the long run [34]. In 2015 discussions about how a rational Bitcoin miner should select transactions from his node mempool raised, and ideas about maximizing miners' profit when creating a new block emerged. Equations that aim to calculate the

cost of mining $\langle C \rangle$ and the miner's profit $\langle \Pi \rangle$ were presented by Rizun [40] and the concept of *orphaning* was introduced. as well Scalability is still a huge concern, causing relevant bottlenecks in Bitcoin which limits the ability of the current P2P overlay network to support substantially higher throughputs and lower latencies [14].

2.3.1 A Transaction Fee Market Exists Without a Block Size Limit

A pressing concern exists over the ramifications of changing (or not) a Bitcoin protocol rule called *block size limit*. This rule sets an upper bound on the network's transactional capacity, or *throughput* (γ). The limit is currently set to 1 MB, corresponding roughly to three transactions per second. This limit set by the blockchain protocol, allow miners to include up to 1 MB of transactions selected from their mempools. When this limit was set, it was over eight hundred times greater than what was required. However in 2015, blocks were filled near the capacity and users experienced delays. In 2015 the transaction rate was over three hundred times larger than when the block size limit was introduced. To improve performance the block size limit should be raised, but in that way transactions will not compete anymore for space in the block, creating an unhealthy transaction fee market. Miners should then include transactions in a manner that maximizes their expected profit [40]. For that reason *Miner's Profit Equation*, *Mempool Demand Curve*, *Block Space Supply Curve* and the concept of fee density were introduced. Every time a block is mined, the miner expects to generate a revenue $\langle V \rangle$ at hashing cost $\langle C \rangle$, yielding a profit $\langle \Pi \rangle$ as follows:

$$\langle \Pi \rangle = \langle V \rangle - \langle C \rangle. \tag{2.1}$$

where the hashing cost is represented by:

$$\langle C \rangle = \eta h \mathcal{T}. \tag{2.2}$$

So the hashing cost $\langle C \rangle$ is directly dependent on the miner's individual hash rate, h, the cost per hash, η, and the creation time, $\mathcal{T}$. The revenue $\langle V \rangle$ is calculated with the amount he would earn if he won the block (reward plus fees, $R + M$) multiplied by his probability of winning (ratio between miner's hash rate, h, and hash rate of the Bitcoin network, H), considering also the orphaning rate presented in Equation 2.4.

The Definition 6. A block B_1 generated at a timestamp t_1 is *orphan* if it does not get approved by the network because another block B_2 with a greater timestamp $t_2 > t_1$ which propagates faster, gets approved instead.

So the expected revenue is shown in Equation 2.3:

$$\langle V \rangle = (R + M) \frac{h}{H} \left(1 - \mathbb{P}_{orphan}\right). \tag{2.3}$$

Where $\mathbb{P}_{orphan}$ increases with the amount of time a block takes to propagate through network. If τ is the block propagation time, the probability of orphaning is defined as:

$$\mathbb{P}_{orphan} = 1 - e^{-\frac{\tau}{\mathcal{T}}}. \tag{2.4}$$

We can now define the miner profit equation as follows:

$$\langle \Pi \rangle = (R + M)\frac{h}{H} e^{-\frac{\tau}{\mathcal{T}}} - \eta h \mathcal{T}. \tag{2.5}$$

A *rational miner* selects which transactions to include in his block in a manner that maximizes the expectation value of his profit. This selection is explained with the Mempool Demand Curve and the Block Space Supply Curve. According to the size limit, a block can select a $b \leq n$ transactions from $\mathcal{N}$ to create a new block $\mathcal{B} \subset \mathcal{N}$.

Studies on inclusion came to the conclusion that a block first includes transactions with a higher *fee density*, ρ, which is the ratio between the *transaction fee*, t_f, and the *transaction size*, t_q. To construct the mempool demand curve, the mempool must be sorted from greatest fee density to least and its formula is shown in Equation 2.6.

$$M_{demand}(b) \equiv \sum_{i=1}^{b} t_{fi}, \tag{2.6}$$

and the sum of each transaction size in bytes, form the block size Q:

$$Q(b) \equiv \sum_{i=1}^{b} t_{qi}. \tag{2.7}$$

The mempool demand curve represents then the maximum fee, $M_{demand}(b)$ a miner can claim by producing a given quantity $Q(b)$ of block space. The size of the block a miner elects to produce controls the fees he attempts to claim, $M(Q)$, and the propagation time he chooses to risk, $\tau(Q)$. The block space supply curve represents the fees a miner requires to cover the additional cost of supplying block space Q. This cost grows exponentially with the propagation time. The equation which represents this curve is the one shown below:

$$M_{supply}(Q) = R\left(e^{\frac{\Delta\tau(Q)}{\mathcal{T}}} - 1\right), \tag{2.8}$$

where $\Delta\tau(Q) \equiv \tau(Q) - \tau(0)$. In order to have a transaction fee market without a block size limit, to maximize his profit, the miner constructs a mempool demand curve and a space supply curve. The block size Q^* where the miner's surplus, $M_{demand} - M_{supply}$, is largest represents the point of maximum profit.

2.3.2 Trends, Tips and Tolls

The Bitcoin protocol supports direct payments from transaction partners to miners, also called *fees* (t_f). Acknowledging their role in stabilizing of the system, the right level of transaction fees is a hot topic of normative debate. The actual costs of the system are not extensively studied and Bitcoin may not be as cheap for consumers as it appears. The definition of transaction fee is encoded as difference between the sum of all inputs and the sum of all outputs of a transaction (Equation 3.3). Previous analyses on the Bitcoin blockchain such as the one presented by Möser and Böhme in 2015 [34], show that transaction fees are lower than 0.1% of the transmitted value, which is significantly below the fees charged by conventional payment systems, and at the time of analysis the hard size limit did not (yet) significantly drive the level of transaction fees. However trends for the fees paid per transaction over time noticeably changed. The trend of 0-fee transactions had a drop after April 2012 leaving space to 0.0005 ฿ fee until May 2013. After that, until the beginning of 2015 the trend for fees was of 0.0001 ฿ and 0.0002 ฿, with peaks of 0.001 ฿. Generally, there seem to be two main reasons for the shift in trends: changes to the Bitcoin reference implementation and actions by large intermediaries in the ecosystem. The emergence of 0.0005 ฿ fees in June 2011 can be mapped to the release of version 0.3.23 of the Bitcoin Core client, which reduced the default transaction fee from 0.01 ฿ to 0.0005 ฿. The raise of these last transaction fees in the second quarter of 2012 was probably due to the launch of the gambling website *SatoshiDice* [51]. In May 2013, version 0.8.2 of Bitcoin Core was released. In the past years, during a period between 2011 and 2015 there was a small share of transactions that did not offer fee to miners, most of them offered default fee amount but some of them were even willing to pay a higher fee, a tip. A plausible reason is that paying more fee led to a faster confirmation. Analysis on the blockchain in 2015 [34] state that half of all 0-fee transactions had to wait more than 20 minuets for their first confirmation. In contrast to that, paying a 0.0005 ฿ fee lead to an inclusion into a block in half the time. 10% of all 0-fee transactions took almost 4 hours to confirm, in contrast to 40 minutes for transactions paying a 0.0005 ฿ fee. The difference between paying 0.0005 ฿ or 0.001 ฿ fee is not as pronounced, but the difference in medians are still statistically and economically significant. Analysis on pool behavior regarding a possible systematic exclusion of 0-fee transactions has been done. Shares have shifted between pools quite extensively. In 2013, BTC Guild had a market share of up to 40%, in 2014 both GHash.IO and Discus Fish ousted this pool. Also, the share of other pools has rose in 2014. Previous incumbents like Slush or 50BTC have lost popularity. Possible reasons include economic and technical factors, like pool fees, service availability, or robustness against attacks. Given the dominance of a few mining pools, evaluations whether some pools systematically enforce fees has been made. The results show that two pools, Discus Fish and Eligius,

have a considerably higher share of blocks without any 0-fee transaction, with 30.6% for Eligius and 62.5% for Discus Fish, in contrast to an average of 14.4%. Other than that though, there is no clear evidence for enforcement of strictly positive transactions fees.

2.3.3 Bitcoin Performance Limitation

Can decentralized blockchains be scaled up to match the performance of a mainstream payment processor? What does it take to get there? In 2016 the Bitcoin blockchain took 10 minutes or longer to confirm transactions, achieving 7 transactions per second maximum throughput. Visa credit card confirms a transaction within seconds and processes two thousand transactions per second on average with peaks of fifty-six thousand transactions per second. Bitcoin community has put forth various proposals to modify the key systems parameters of block size and block interval. Anyway, Croman et al. state that such scaling by reparametrization can achieve only limited benefits [14]. This is because because Bitcoin generates a lot of network traffic, due to its decentralization, this leads to have a lot of peers in the network and they all have to interact. To ensure that most of the nodes in the overlay network have sufficient throughput we follow two guidelines:

- **Throughput limit.** The block size should not exceed 4 MB given 10 minutes average block interval. Corresponding to maximum 27 transactions per second.
- **Latency limit.** The block interval should not be smaller than 12 seconds.

The maximum throughput of 3-7 transactions per second is a number constrained by the 1 MB block size Q and the 10 minutes creation time $\mathcal{T}$. About the propagation time τ, measured in 2016 [14] shows that 10%, 50% and 90% block propagation times are 0.8 seconds, 8.7 seconds and 79 seconds respectively, with an average block size of 540 kilobyte (KB). Projecting to a 1 MB block size, τ would be 2.4 min, 15.7 sec, and 1.5 sec respectively. In that way an effective throughput on the network could be calculated as follows:

$$\text{X\% effective throughput} = Q/(\text{X\% block propagation delay}).$$

Having with X = 50%, an effective throughput of 496 Kbps, equals to 248 tx/sec, and with X = 90%, an effective throughput of 55 Kbps, equals to 26 tx/sec. It follows that Q, and interval $\mathcal{T}$, must satisfy Equation 2.9:

$$\frac{Q}{\text{X\% effective throughput}} < \mathcal{T} \tag{2.9}$$

Consequently, for a 10 minute (or shorter) block interval, the block size should not exceed 4 MB for X = 90%; and 38 MB for X = 50%, corresponding to a

throughput of at most 27 transactions per second. To improve the system's latency it could be enough to reduce the block interval. To maintain effective throughput would also require a reduction in the block size. Propagating a block smaller than 80 KB would not make full use of the network's bandwidth, as latency would still be a significant factor in the block's propagation time. Propagating an 80 KB block to 90% of the nodes would take roughly 12 seconds. In conclusion, to retain at least 90% effective throughput and fully utilize the bandwidth of the network, the block interval should not be smaller than 12 seconds.

2.4 Machine Learning

When a new node joins the Bitcoin network, it has to download all the useful information to process transactions and verify them. This information consist of a relevant quantity of data, nowadays ~ 125 GB, and it grows continuously over time. A full node might require more than four days to download the entire ledger. Once all the data are collected, is necessary to implement a way to store them and to get the right information out of them. Many data structures allow the storing of big datasets, and several machine learning techniques grant to infer more information out of them. While applying machine learning techniques to big datasets might be recommended to use data structures such as *trees* or *data frames* which are more likely to be used in machine learning algorithms.

2.4.1 Training Dataset

Having a huge dataset implies having a big amount of data with different types of attributes. These attributes might have *numerical* or *categorical* values.

Numerical: Data that can be measured, quantified. For example the fee t_f paid to a miner in Bitcoin Currency (BTC).
Categorical: Data which represent characteristics. For example the name of the miners in the Bitcoin system.

These data might be saved in a data structure such as a data frame, having a sample for each row and different attributes of that particular sample on every column. Once data are collected, they might be trained to create predicting models or other useful material. Training and predicting models involve different types of variables:

Target variables: the variable that you are attempting to predict, represented

with Y;

Predictors: variables that you can use to make the prediction, represented with X.

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix} \qquad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \tag{2.10}$$

When the targets are real numbers, the problem with training those data is called *regression problem*. If the targets are two-valued the problem is called *binary classification problem*. We then need to predict every $y_i \in Y$ using each row $x_i \in X$ and evaluate the performance of our predictions. Good performance means using the attributes x_i to generate a prediction that is *close* to y_i. For a regression problem where y_i is a real number, performance is measured in terms like the mean squared error (MSE) (Equation 2.11) or the mean absolute error (MAE) (Equation 2.12) [9].

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^{m} (y_i - pred(x_i))^2 \tag{2.11}$$

$$\text{MAE} = \frac{1}{m} \sum_{i=1}^{m} |(y_i - pred(x_i))| \tag{2.12}$$

However, since MSE is in squared units, the Root Mean Square Error (RMSE) is usually a more usable number to calculate. If the problem is a classification problem, then other measures of performance must be used. One of the most used is the *misclassification error*. Classification problems generally revolve around misclassification error rates and, usually, algorithms for such kind of problems can present predictions in the form of a probability rate for the attributes rather than an attribute itself.

Before and during the analysis we made assumptions about how attributes and values might change if other attributes were increased or decreased. These assumptions are shown in Table 3.2. To be able to see whether an attribute is related to another we might measure it by using *Pearson's correlation*. Pearson's correlation coefficient is defined for two equal length vectors u and v:

$$u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \qquad v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \tag{2.13}$$

First subtract the mean value of u from all the elements of u, and the same with v, then we generate Δu and Δv as follows:

$$\overline{u} = avg(u)$$

$$\overline{v} = avg(v)$$

$$\Delta u = \begin{bmatrix} u_1 - \overline{u} \\ u_2 - \overline{u} \\ \vdots \\ u_n - \overline{u} \end{bmatrix} \qquad \Delta v = \begin{bmatrix} v_1 - \overline{v} \\ v_2 - \overline{v} \\ \vdots \\ v_n - \overline{v} \end{bmatrix} \tag{2.14}$$

The Pearson's correlation between u and v is defined as follows in Equation 2.15 [9]:

$$corr(u, v) = \frac{\Delta u^T \times \Delta v}{\sqrt{(\Delta u^T \times \Delta u) \times (\Delta v^T \times \Delta v)}} \tag{2.15}$$

2.4.2 Pandas Data Frame

In machine learning is important to pick one data structure and store your data in it for the analysis and testing. We focus on explain Pandas data frame [6] since is the data structure we chose for this thesis. Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. One of the primary data structure that Pandas provides is *DataFrame*. A data frame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). Arithmetic operations align on both row and column labels [6]. In our case each sample is a transaction T, and if we consider a data frame D like a set created from the union of all m transactions retrieved, $D = \{T_1 \cup T_2 \cup \cdots \cup T_m\}$, then we have the following representation of D:

$$D = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_m \end{bmatrix}. \tag{2.16}$$

Now, if every transaction T has n attribute, $T = \{a_1, a_2, \ldots, a_n\}$, we can finally represent our data frame as follows:

$$D = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix} \tag{2.17}$$

Furthermore, operations such as groupby, mean, median, sum are well managed in Pandas data frame, making it easy to manage your data. In Appendix C code for these operations is shown. This is just a theoretical explanation of data frame and we focus on a practical explanation on our data structure in Chapter 3.2.2.

2.4.3 Visualizing Data

When analyzing big datasets, finding a smooth and nice way of representing information of them might be tricky. For example, if you have millions or billions of samples it will result almost impossible to get any information by simply plotting these data. When analyzing longitudinal data for example, information might be shown daily, monthly or even yearly-wise, then the *mean* for every portion is applied. However, bigger is the dataset and most likely it will contain *outliers*. In this case, calculating the mean on those data might be misleading, that is why in a longitudinal study should also be considered whether to apply a *median* value for every portion rather than the mean, since the mean is particularly susceptible to the influence of outliers.

3

Blockchain Analytics System - BAS

To understand digital cryptocurrencies, in particular Bitcoin, and to study existing systems, a complete data analytics system, BAS, was designed and implemented. The system retrieves information from different sources, writes it into a local data structure and then analyzes data by generating and plotting new information. Using this system, we were able to run a longitudinal study on the Bitcoin blockchain, collecting useful information from 2014 to 2017, analyzing more than one hundred twenty million of transactions, which corresponds to about half of the blockchain for that period. Previous studies [34, 17] did not analyze such large portion of the blockchain. Using BAS, such studies can be conducted more easily and be scaled up. We believe this is essential for understanding cryptocurrencies in the wild.

This chapter details our experimental setup and explains how our studies on the Bitcoin blockchain took place, clarifying how we analyzed the APIs from `blockchain.info` to get the best information out of them. We illustrate also how we collected and then manipulated our data, describing how we plotted our data and why. In Chapter 4 again, we point out our solutions compared to our assumptions and our conclusions.

As mentioned in Chapter 1.3 we opted for building a small data analytics system locally rather than using Bitcoin Core client or running full nodes in the Bitcoin

network. By doing that we had the freedom and the flexibility to retrieve and store only the information we needed for our purpose, and considering how much the blockchain scales it turned out to be one of the most valid points. Furthermore, we thought that the effort for setting up one or more nodes in the system was far above than the benefits we would have gained. It would have only allowed us to have a better estimation of the block propagation time, hence we are using the one evaluated first in 2013 by Decker et al. and then in 2015 from Croman et al. [17, 14], and we can consider them reliable enough for the purpose of our work. Plus, from the beginning we excluded the idea of using Bitcoin Core, since it forces you to download, keep, and update the ledger of data, without providing information about propagation time nor miners.

3.1 System Architecture

BAS architecture is shown in Figure 3.1. Information coming from the Bitcoin network is collected in databases and web services such as `blockchain.info` [33] or `coindesk.com` [2]. We use their web services, REpresentational State Transfer (REST) APIs and HTTP parsing to get this information. Our system then collects and gathers data locally and saves it in our data frame D. After that data are ready to be trained and plotted.

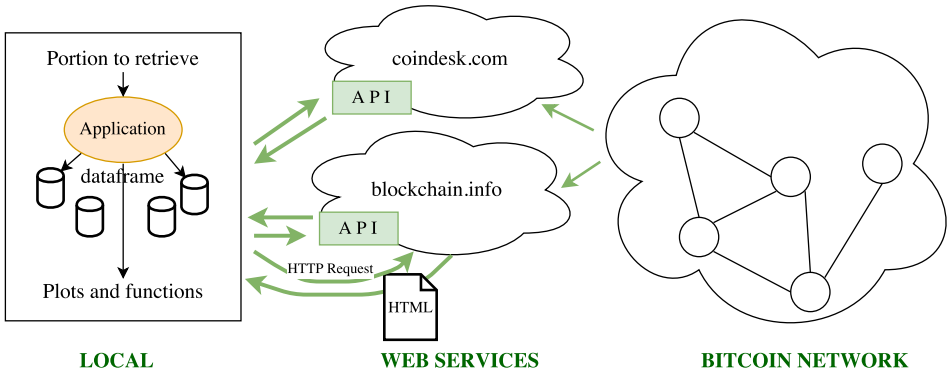


Figure 3.1: Architecture of BAS.

As Figure 3.2 shows, the information retrieval works in a combined way, *block-wise* and *transaction-wise*, information gathered from both sources are written and stored in D. BAS collects information from:

1. **JavaScript Object Notation (JSON)** file sources: files processed block by block and for each block transaction by transaction, then information

are added to D.

2. **HTML** file sources: files obtained via a HTTP request. The parsed information are directly added to D.

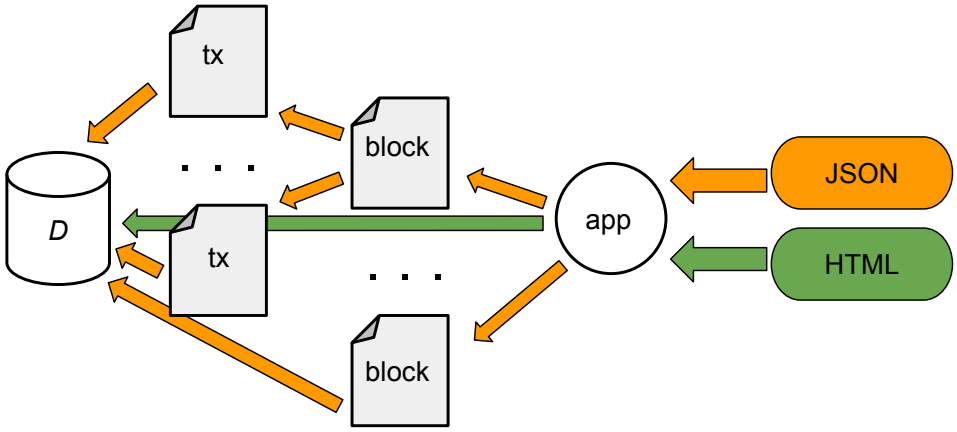


Figure 3.2: How information is retrieved and gathered in our data frame D.

3.2 Data Sources

Many websites like `tradeblock.com` [41] or `blockchain.info` [33] are observing the Bitcoin blockchain every day. These websites also provide a remote API that allows you to retrieve information about blocks and transactions directly from the website through REST APIs. Furthermore, websites like `coinbase.com` provide useful information about the money exchange price, and they arrange API along with libraries like the one we used, called *forex-python* [27]. As already mentioned, one of the most popular system for analyzing Bitcoin data is Bitcoin Core [46]. However, the client requires users to download the full ledger of data before any analytics can take place, which at the time of writing was 125 GB, with a total disk space required of 250 GB. Since we wanted to optimize and be able to select data we wished to analyze and did not have the need to run a full node, we developed a system for blockchain retrieval and analysis. This system allows to fetch a portion of the blockchain by storing data in Pandas data frame [6]. This allows saving one order of magnitude of disk space while still storing all the information needed for our purpose. Finally, the system displays all the knowledge acquired, analyzing data and applying machine learning techniques on those, using Python libraries such as *matplotlib* [22] and *seaborn* [52]. Even though we know Python is not known for its computational speed, it provides nice access to the Bitcoin blockchain by using APIs arranged by `blockchain.info`, allowing the retrieval of all the information stored in a block and transaction, plus, its enormous amount of

Table 3.1: Data sources and information gathered

Source	Entity	Information
Blockchain.info APIs	Block, Transaction	B_{ha}, t_{ha}, B_t, B_h, $\mathcal{T}$, Q, $t_{in}, t_{ou}, t_q, B_{epoch}, t_{epoch}$
Blockchain.info HTML parsing	Miner	B_{mi}
Coindesk.com	Price	USD and ₿ value
Data frame D	Block, Transaction, Miner, Price	τ, t_f, t_l, γ, ρ, $t_{\%}$

libraries grants a vast flexibility on which machine learning algorithms can be applied or which plotting system to use.

3.2.1 Data Retrieval

The architecture displayed in Figure 3.1 gives information about the different data gathered in our system. The system imports Bitcoin APIs and uses them for data retrieval. The website `blockchain.info` also provides REST APIs, in that way requests made to a resource's Universal Resource Identifier (URI) will elicit a response that may be in eXtensible Markup Language (XML), HTML, JSON or some other defined format. In our case, REST APIs from Bitcoin return JSON data. The website `blockchain.info` is monitoring the blockchain 24-7, producing graphs and statistical analysis on the actual Bitcoin network. The local application is monitoring these data as well, producing graphs that are not taken into consideration from this website, using a finer granularity to represents the data, allowing us to do an in-depth analysis on those. For our analysis we select a time range from April 2013 to September 2017, considering more than one hundred twenty million of transactions and one hundred thousand of blocks. Before 2013, analyses on the system were already performed and evaluated [14, 21, 34, 40] plus the popularity of the system before 2011 was low and interpreting this early data would not be useful and might even be self-defeating in order to understand system's behavior. To have an overall view of the entire Bitcoin network we combine data from different sources. As Table 3.1 shows, we use both `blockchain.info` APIs and HTTP parsing to gather information about blocks, transactions and miners, plus we used `coindesk.com` APIs [2, 27] to gain information about the pricing of USD and ₿. Unfortunately APIs from `blockchain.info` do not allow us to retrieve all the information we need, knowledge regarding miners is not included in the blockchain, so we need to get this information by parsing web pages on `blockchain.info`. Finally, additional useful information and derived data such as throughput γ or transaction fee t_f are obtained using our data structure

D (Chapter 3.2.2), containing both, this new information and the previously collected ones. An example of data retrieval is shown in Appendix C.6 and the retrieved block and transaction structures by using RESTful APIs is shown in Appendix C.7, C.8.

First idea, Blockchain Splitted in Portions

During our analysis we evaluate different possible ways to retrieve data. Because of its relevant size and its fast growth, we considered the blockchain as an object and we thought that analyzing different portions over time following a certain pattern would give us an accurate summary of the whole system. We started retrieving data by dividing the blockchain in k portions. Suppose now that m is the last block's height, then we divide all in k portions, having $p = m/k$. While retrieving b blocks, they are retrieved and stored in the following way:

$$\langle 1 \dots b \rangle, \langle p \dots p + b \rangle, \langle 2p \dots 2p + b \rangle, \dots, \langle kp \dots kp + b \rangle.$$

Retrieving blocks in such portions gives statistical representation of the whole system. Like Figure 3.3 shows, every new b blocks added to the blockchain will be added to the indexes:

$$ip + b \ \Big\{ \text{for } i = 0 \text{ to } k. \tag{3.1}$$

Retrieving data in this way reduce drastically the space requirements and yet

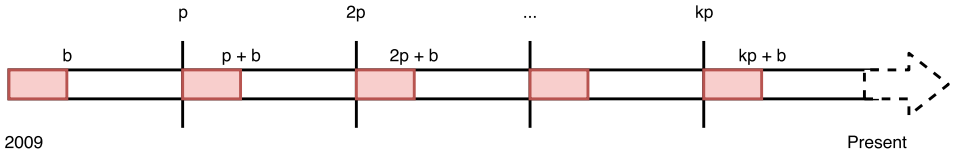


Figure 3.3: Science of Bitcoin blockchain, the entire blockchain considered as an object and divided in k portions. One of the first attempts of data retrieval on Bitcoin blockchain.

retaining information about the whole system. However, data collected using these methods were not enough to achieve our target of longitudinal study and analysis, but this technique gave us a spark for the next data retrieval attempt.

Fragmented Blockchain

In the next approach, we use much smaller interval between each portion retrieved, called jump or J. Plus we do not retrieve the earliest part of the

blockchain, the one from 2009 to 2013. By retrieving 10 blocks each time, according to the granularity of J we can have a more or less precise analysis. To balance disk space and precision, we choose a $J = 10$ and a $b = 10$ blocks retrieved each time we performed BAS. This means that every 10 blocks retrieved, there was a jump of other 10 blocks, corresponding to the ~50% of the entire blockchain, daily analyzed. Figure 3.4 shows how our last data fetching has been implemented, so D is generated accordingly, collecting transactions happened in one hundred minutes and then having one hundred minutes of gap. This method allowed us to collect relevant information from 2013 to 2017 by only using ~25 GB of disk space. Then, the pattern we have chosen to store

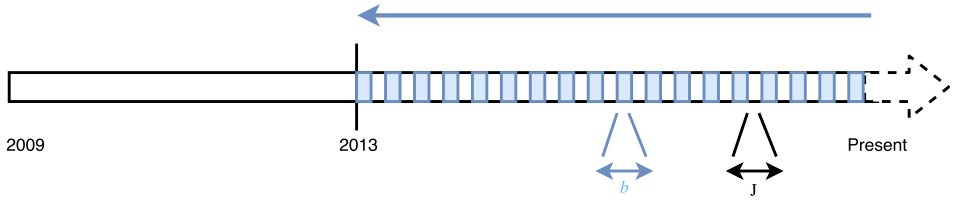


Figure 3.4: Fragmented blockchain divided in smaller portions with a jump of J blocks.

transactions is the following:

$$\langle m \ldots m{-}b\rangle, \langle m{-}b{-}J \ldots m{-}2b{-}J\rangle, \ldots, \langle m{-}(k{-}1)b{-}(k{-}1)J \ldots m{-}kb{-}(k{-}1)J\rangle.$$

And we can write the indexes for each portion that has to be retrieved according to Equation 3.2, following the format $\langle \text{start} \ldots \text{end}\rangle$:

$$\langle m - ib + b - iJ + J \ldots m - ib - iJ + J\rangle \Big\{ \text{for } i = 1 \text{ to } k. \tag{3.2}$$

Having m as the height of the most recent retrieved block and k number of portions. An example of this retrieval method is shown in Appendix C.9.

3.2.2 Data Organization

Figure 3.2 shows how data, once retrieved, are collected and gathered in BAS. The Pandas package makes it possible to read data into a specialized data structure called a *data frame* [6]. As argued by Michael Bowles [9], you can think of a data frame as a table or matrix-like structure with a row representing a single case or observation and columns representing particular attributes. We decided to use a data frame structure and not a matrix because even though the data frame structure is matrix-like, its elements in various columns may be of different types. For statistical problems, the matrix is too confining because statistical samples typically have a mix of different types, in addition, Pandas package makes it possible to automate the steps to calculate *mean*, *variance*,

median, *group-by* elements, and it offers a nice and smooth control over the data.

Finally, once data are retrieved and collected, they are saved in a data frame structure that we call D. Derived data such as γ or ρ, useful for the blockchain analysis, are obtained using D. This data structure is divided among multiple files with an overall size of ~25 GB and it contains both numerical and categorical variables, saving an average of 5 times the amount of disk space compared to the same information you get by downloading the full ledger, reaching peaks of 10 times for some blocks. In D, information is stored in a bi-dimensional table, with transaction samples represented in rows and their attributes in columns. If we consider a set of attributes as a transaction, T, and a set of transactions D, then we have $T = \{a_1, a_2, \ldots, a_m\}$ and $Y = \{T_1 \cup T_2 \cup \cdots \cup T_n\}$. Our data frame D will have then a cardinality of $|D| = (m \times n)$ and it will be structured as shown in Equation 2.17.

Once defining the structure for our dataset D, we consider a set T having the following attributes (Appendix B):

$$T = \{t_{ha}, t_{in}, t_{ou}, t_f, t_q, t_{\%}, t_l, Q, \mathcal{T}, B_h, B_{epoch}, B_t, B_{ha}, B_{mi}\}.$$

We structure the way our data are stored in order to do subsequential sampling. This is the reason why we stored transactions ordered by block and each block ordered by height/epoch, in that way we have the dataset ordered by block creation with all its transactions, making it easier to do analyses or sampling data in a chronological order. Summarizing, our data frame allows us to collect all the useful information without storing the whole raw blockchain like current systems for blockchain analysis [46] do. Raw data are collected in JSON format, analyzed, printed to our data frame and then deleted. This yielded up to save 10 times the amount of disk space, storing in a 1 GB data frame what the raw files from `blockchain.info` store in 10 GB space. Plus, we believe that analyzing a small percentage of the blockchain everyday will give us relevant information about the whole system. Due to time limitations we could only collect 50% of the data, every day.

3.2.3 Data Visualization

Once data are retrieved and saved in our data structure D, is finally possible to get the information we need out of them. Even though Pandas offers a nice way to plot your data, visualizing big data might be not always immediate or intuitive, and a study on how to plot your data is necessary if you want to gain the right information out of it. Once data are ready to be analyzed, the first step is to determine the outliers. A spontaneous question that came up in our minds is: *how to deal with outliers?* We could segregate them out and train on them

as a separate class, or easier, when data are grouped, a median is calculated instead of the mean, since mean is extremely sensitive to outliers. Dealing with categorical attributes instead, as the number of attributes grows, the complexity of handling them mounts, also, most of binary tree algorithms, which are the basis for ensamble methods, have a cutoff on how many categories they can manage [9]. Random Forest package written by Breiman and Cutler has a cutoff of 32 categories [10]. Our only categorical variable is B_{mi} so we did not deepen the analysis of categorical attributes.

To visualize our data we use Pandas libraries as well as *matplotlib* and *seaborn* [6, 22, 52]. We mostly use seaborn when we need to plot data according to a certain category, so we need multiple samples for each (x, y) point, e.g. Figures 4.2, 4.8. We use it to calculate how major mining pools change their number of transactions approved over time, or to calculated the ρ for every mining pool. We also use it to calculate the linear regression and the Pearson's coefficient for every attribute $a \in D$. Pandas was used mainly for area plots and pie plots, e.g. Figure 4.11, 4.12.

One example of data manipulation and visualization on our dataset D regards to the calculation and classification over time of the fee density, ρ. From D, we collected the data we needed and we generated another data frame D'. In that way we have a new data frame with a new set of attributes X', generated according to to the transformation we need, calling it for brevity $\xrightarrow{\lambda}$, having then

$$D \xrightarrow{\lambda} D',$$

$$T \xrightarrow{\lambda} X'.$$

So the transformation $\xrightarrow{\lambda}$ brings a new set of attributes with new data in it. This new X' contains a set of numerical values categorized, regarding ρ, plus the related date and the total number of transactions approved in the following way:

$$X' = \{0, <50, <100, <200, <300, >300, \text{date}, \text{total}\}.$$

Another use of data manipulation for visualization is the one we apply to the epoch. We want to make our data more readable and categorize the time, by transforming our epoch in date time and then get only the information we need regarding the day, the month or the year. By doing that we have to apply a function to our data frame D that hits one attribute (in this case the epoch, B_{epoch}), convert it to date and then parse it to get information about the day, month or year. The listing is shown in Appendix C.5.

3.2.4 Derived Data

As shown In Table 3.1, our data frame D gives us information about derived data, from other data collected. This section aims to explain how they are calculated and why they are so important. We first focus on calculating the exact amount a transaction has to pay in order to get the payment processed and approved: the transaction fee, t_f. This fee is a difference between the sum of all inputs t_{in} and the sum of every output t_{ou} of a transaction t. In that way, if n is the number of input and m the number of output, t_f is calculated according to Equation 3.3 and measured in BTC (฿).

$$t_f = \sum_{i=1}^{n} t_{in_i} - \sum_{i=1}^{m} t_{ou_i}. \tag{3.3}$$

Next, we calculate the time it takes for every transaction t, to be approved in the network: transaction latency, t_l. This value is calculated following Equation 3.4 and is the difference between two epochs, the first is the block creation time in which a transaction t is included, the second is the transaction timestamp, so when it was created. t_l it could be measured in seconds, minutes or hours.

$$t_l = B_{epoch} - t_{epoch}. \tag{3.4}$$

Another relevant derived information using D, is the throughput of the Bitcoin network, or how many transactions it can approve per second. We measure it according to Equation 3.5, we call it γ and the unit used is transactions per second (tx/sec).

$$\gamma = \frac{t_B}{\mathcal{T}}. \tag{3.5}$$

As stated by Rizun [40], information regarding fee density might be of a relevant interest. This value, that we represent with ρ, could be an important factor for miners to chose whether to include a transaction or not in their next block, and indicates how many BTC(฿) per bytes a transaction t has to offer. Equation 3.6 shows how ρ is calculated.

$$\rho = \frac{t_f}{t_q}. \tag{3.6}$$

We also calculate and insert in D the fee paid in percentage, compared it with the total input of a transaction t. This value might be useful when the Bitcoin price increases or decreases drastically, in a way that we are able to monitor if the fee is stable, or changes over time. Then we calculated $t_{\%}$ as it is shown in Equation 3.7:

$$t_{\%} = 100 - \frac{f_{ou} \times 100}{f_{in}}. \tag{3.7}$$

Table 3.2: Possible Relations Between Major Attributes

	Q	$\mathcal{T}$	τ	γ	t_f	$\langle C \rangle$	$\mathbb{P}_{orphan}$	t_l
$Q \uparrow$			$\uparrow\uparrow$	$\uparrow$	$(\downarrow)$	$\uparrow$	$\uparrow\uparrow$	$(\downarrow)$
$\mathcal{T} \uparrow$	$(\uparrow)$			$\downarrow$				$\uparrow$
$\tau \uparrow$	$\uparrow\uparrow$			$\downarrow$	$(\downarrow)$	$(\uparrow)$	$\uparrow\uparrow$	$\uparrow$
$\gamma \uparrow$	$\uparrow$	$\downarrow$			$(\downarrow)$	$\uparrow$	$(\uparrow)$	$\downarrow$
$t_f \downarrow$			$(\uparrow)$			$\uparrow$		$\uparrow$
$\langle C \rangle \downarrow$	$\downarrow$				$\uparrow$			$(\downarrow)$
$\mathbb{P}_{orphan} \downarrow$	$\downarrow$					$\downarrow$		

[1] $\uparrow\uparrow/\downarrow\downarrow$: more than linear or even exponential increase/decrease
[2] $\uparrow/\downarrow$: increase/decrease.
[3] $(\uparrow)/(\downarrow)$: might increase/decrease.

The last derived data is the block propagation time, τ. The calculation of this value follows the papers from Decker and Croman [17, 14]. From 2012 to 2015 block propagation time had an increment. When it was tested from Decker and Wattenhofer in 2012 the median and 90-percentile time for Bitcoin nodes to receive a block was 6.5 seconds and 26 seconds respectively. Considering that at the time of their measurements, the average block size was 87 KB, a full 1 MB block would have taken 5 minutes to be visible for 90% of the nodes in the network. In 2015, the last measurements from Decker and Croman showed that the 10%, median, and 90% block propagation times were 0.8 seconds, 8.7 seconds, and 79 seconds respectively. Further, the average block size was at the time roughly 540 KB. Projecting to a 1 MB block size, the 90%, median, and 10% block propagation times would be 2.4 min, 15.7 sec, and 1.5 sec respectively. We consider this last measurement our reference for the block propagation time τ.

3.3 Assumptions

Our focus and studies are related to blockchain scalability, performance and to investigate whether the constant increase of transactions per day affects the way miners include transactions or the way users offer fees to miners. We assume that, analyzing the last 5 years of transactions might produce interesting results about how this system is evolving and changing its properties, and how this properties are related between each others. We studied and analyzed previous works on the Bitcoin blockchain [14, 40, 34, 17] and we made some assumptions of how the following properties of the Bitcoin systems may vary if other attributes are changed as well. We tried to classify the main properties of the system and make assumptions on how they may vary. These attributes

are Q, $\mathcal{T}$, τ, γ, t_f, $\langle C \rangle$, $\mathbb{P}_{orphan}$ and t_l and our assumptions are summarized in Table 3.2. For each property, the rows describe what happens to other attributes if the attribute considered, is increased/decreased. White spaces means that there might be no relation or we do not have any assumption yet. For example, if we are going to increase the block space Q, then we have a relevant increase of the propagation time, τ, the chances of orphaning are way bigger, so it will be the cost of mining a block, but we have a better throughput γ, and we might have less fee from transactions, t_f, since they will have less competition while trying to be included in a block.

4

Observations

In this chapter we present and discuss observations of the Bitcoin blockchain, captured by the analytic system outlined in Chapter 3. To evaluate and discuss our problem definition in Chapter 1.2, we focus on considerations related to performance, scalability, fees and costs. To answer our questions and focus on the problems and possible solutions, a large number of test has been evaluated, more than one hundred twenty million transactions over one hundred thousands blocks were collected, stored and analyzed, between April 2013 and September 2017. Graphs and relations between attributes are represented and discussed, and they mine to verify our assumptions in Table 3.2. We can finally state that scalability brought to a centralization problem concerning miners, since nowadays there are less individual miners and more mining pools, an unlikely and ill-judged increment of $\mathcal{T}$ that might lead to an excessive high throughput γ, and an urgent problem whether to increase the block size Q or not, due to the massive scale of transactions to approve; performance studies brought to consider carefully an increment of γ since it may leads either to a raise in costs $\langle C \rangle$ or a growth of $\mathbb{P}_{orphan}$, while a user-side performance study concerning the transaction latency t_l, enhanced after 2015, t_l became strictly dependent of t_f; and finally a study on tips and tolls brought us to analyze miner's profits and users' benefits, and thanks to our data collected we were able to infer the miner profit function $f^2_{\langle \Pi \rangle}$, which establishes a relation between a block creation time and a miner's profit $\langle \Pi \rangle$, and to analyze how the fee paid to the miner changed over time, studying also the fee density ρ.

4.1 Scalability

Scalability may concern the number of nodes in the network but also the amount of transactions requested per second that the system faces. The constant growth of cryptocurrencies popularity rise many concerns regarding the scalability of the system. The reward for miners resulted in an increasing interest in mining, and the number of nodes for this purpose highly increased during years. From that, *mining pools* emerged to solve the puzzles faster and share the earned reward afterwards. This creates a centralization problem though, only bigger mining pools nowadays, such as *AntPool* [55], *F2Pool* [3], *BTCC Pool* [1] or *BitFury* [50], will find a solution to mine new blocks, discouraging a lot of small miners to join the network.

More scalability problems are related to the growing amount of transactions that need to be approved every day. With an average t_q of 500 bytes, a fixed estimated $\mathcal{T} = 10$ minutes and a maximum block size Q fixed at 1 MB, the outside number of transactions the system can approve per second is roughly 3-4 transactions per second. Compared to systems like VISA that can easily approve two thousand transactions per second, Bitcoin or any other digital cryptocurrency using the blockchain protocol are still far in taking over digital systems such as VISA or Master Card. However we want to analyze these boundaries and understand what it possible to do with regards to improving scalability.

4.1.1 Miners & Mining Pools

In our data frame D, all the information regarding every single transaction are stored, including which miner approved it. We analyzed and tracked down every active miner in the system from 2013 to 2017. With 50% of the total information we evaluate a total number of 6137 miners, of which 6066 are just occasional nodes not belonging to any mining pool. Note that this number is approximate to the number of active miners, and it counts every mining pool as one, since mining pools withheld information about miners in their systems. In Figure 4.1 the number of monthly active miners from 2013 until now is displayed. Even if the number is relative to 50% of the blockchain, we analyze every block mined over time and we state that after December 2014 until August 2015 there was a relevant drop, connected to the growth showed in Figure 4.2, mainly caused by the come of mining pools, KnCMiner, AntPool and BitFury in late 2014, and the massive mining power growth of SlushPool and F2Pool in mid 2015. This is the main cause of the occasional miners' drop in the system, and more mining pools will join the network, more it will be disheartening for smaller nodes to start mining, since the costs will be probably higher than the revenue. This creates a sort of centralization problem, since

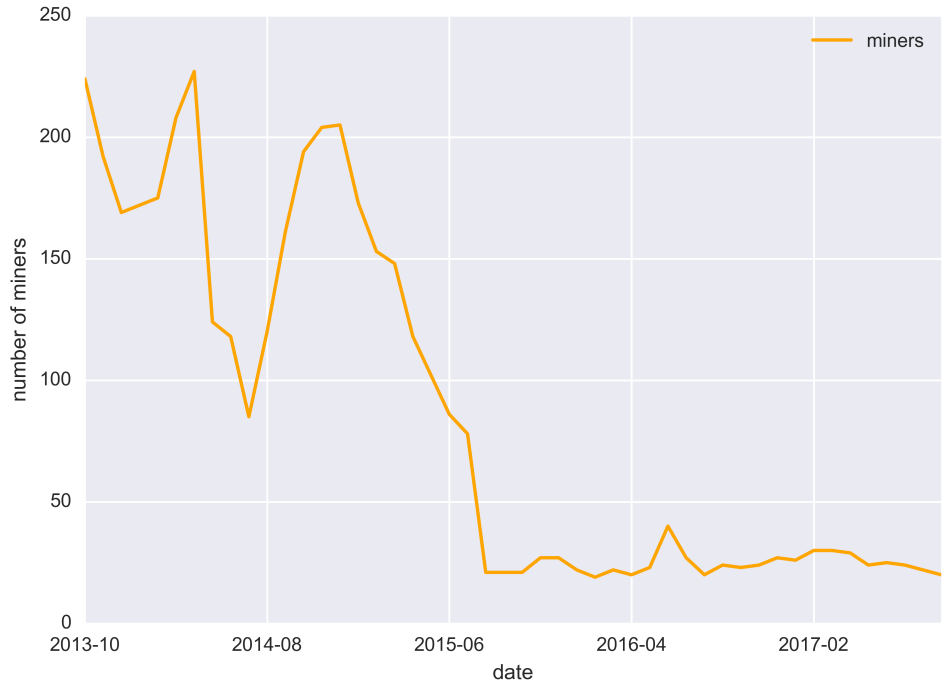


Figure 4.1: Monthly number of active miners from 2013 to 2017.

only the biggest mining pools could take part in the whole process of approving transactions, forcing small and occasional miners to join big mining pools. From our analysis, we state that there are different mining pools joining and leaving the Bitcoin network during these years. As Figure 4.2 shows, the most relevant example is the *GHash.IO* pool, which was one of the major mining pool until mid 2015, then its mining power started to decrease slowly until its end in October 2016. We refer to an article from `coindesk.com` [20] which says that Bitcoin miners ditched GHash.IO pool over fears of 51% attack (Appendix A). Indeed in 2014, according to `blockchain.info`, GHash.IO accounted for more than 42% of Bitcoin mining power. Nowadays as the Table 4.1 shows, AntPool has 19.53% of the mining power, and there is no centralization risk. After that, we analyzed the major mining pools from 2013 until 2017, dividing the time period in two portions, 2013-2015 and 2016-2017, and selecting the first 10 miners regarding their mining power, as represented in Tables 4.2-4.1. Table 4.2, shows that a big percentage of the mining power comes from occasional miners, and for the whole period between 2013 and 2015 GHash.IO only had 13.3% of share. We can see in Figure 4.2 that this mining pool only had a short time peak between 2013-2015, but anyway in an overall scheme it managed to collect 13.3% of the whole share. Table 4.1 shows that AntPool, F2Pool and BTCC Pool are dominating the mining power in the Bitcoin network from 2016 until now with values respectively of 19.53%, 16.43% and 10.5% of share. If compared

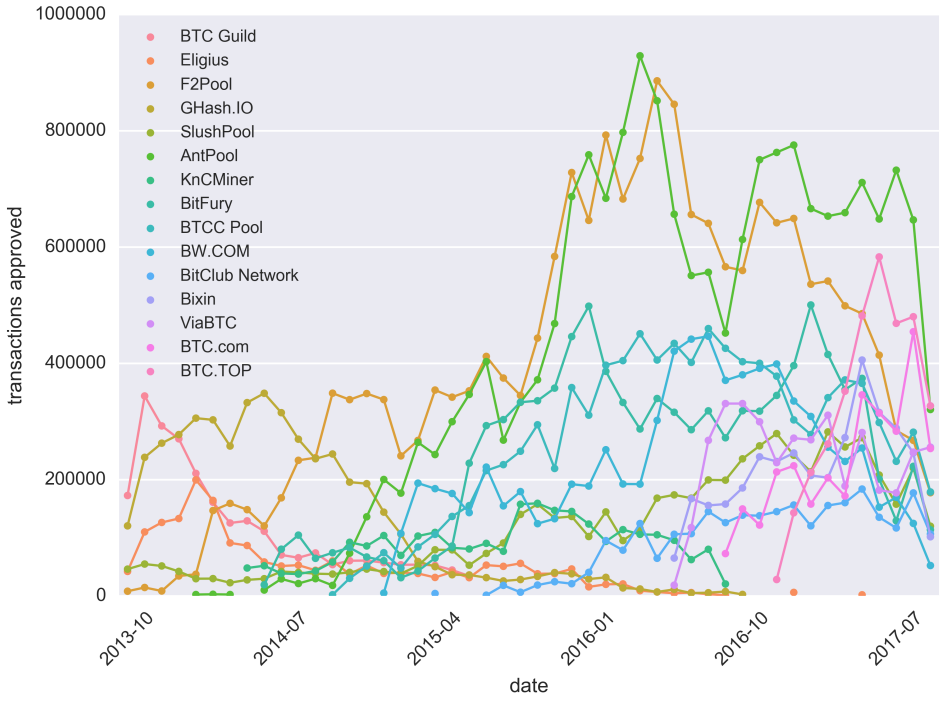


Figure 4.2: Top 15 mining pools from 2013 until 2017 and their trend according to the number of transaction they approve every month.

with Table 4.2, AntPool is the one which is growing the fastest, and mining pools such as GHash.IO and BTC Guild stopped their activity. We state that today, no miner seems to get even close to the GHash.IO share in 2014 and it seems like we will not have a centralization problem in the very near future. However, the mining power over the years changed from a distributed architecture to a decentralized one. We can see in Figure 4.3 that occasional miners almost disappeared in the last two years, leaving space to mining pools in their absence, while they were contributing for almost the 100% at the end of 2013. This brought a downside for individual users that want to buy their own hardware and start mining for themselves. Since Bitcoin always claimed its distribution as a system strength, this scalability issue might result a centralization problem in the long run, excluding small miners from the system and letting only the biggest mining pools to have information about the whole blockchain. Bitcoin is useful only if is decentralized because centralization requires trust and Bitcoin value proposition is trustlessness.

Table 4.1: Mining power distribution in a scenario from 2016 until 2017, considering the 10 major mining pools.

Mining Pool	Mining Power (%)	Blocks Mined
AntPool	19.53	9,078
F2Pool	16.43	7,635
BTCC Pool	10.5	4,874
BitFury	8.71	4,050
BW.COM	8	3,717
SlushPool	5.26	2,447
ViaBTC	4.8	2,222
Bixin	4.12	1,947
BTC.TOP	4	1,857
BTC.com	3.55	1,648
Total Analyzed	84.9	39,475
Other Miners	15.1	6,996

Table 4.2: Mining power distribution in a scenario from 2013 until 2015, considering the 10 major mining pools.

Mining Pool	Mining Power (%)	Blocks Mined
F2Pool	13.53	10,548
GHash.IO	13.3	10,370
BTC Guild	7.37	5,749
AntPool	6.82	5,323
Eligius	4.99	3,889
BitFury	4.73	3,693
BTCC Pool	3.9	3,042
SlushPool	3.37	2,627
KNCMiner	3.12	2,432
BW.COM	2.67	2,064
Total Analyzed	63.8	49,737
Other Miners	36.2	28,225

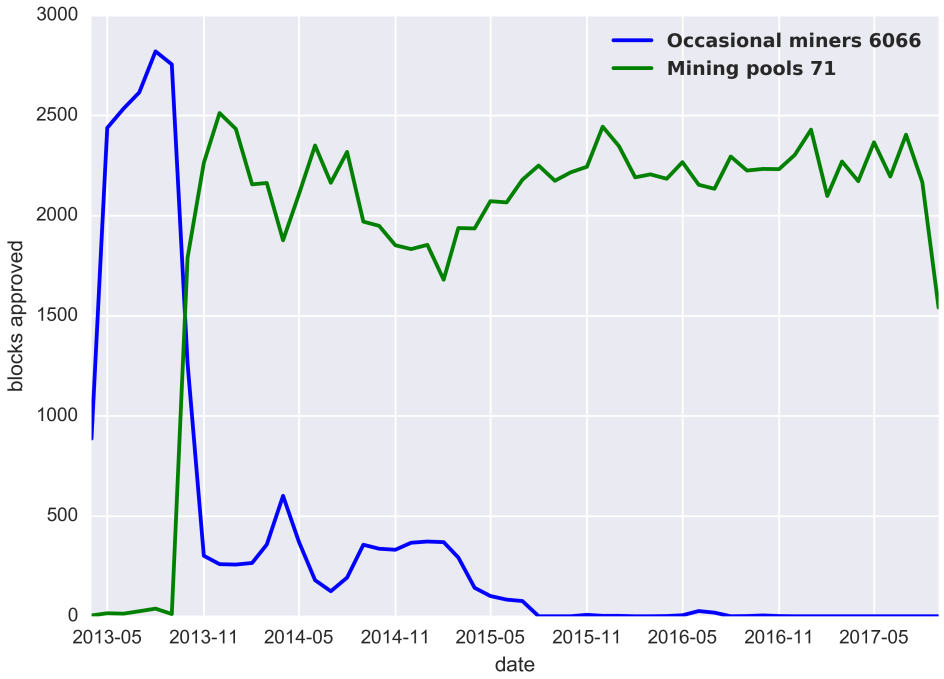


Figure 4.3: Number of transactions approved from occasional miners and mining pools from 2013 until 2017.

4.1.2 Unconfirmed Transactions

Another relevant concern about scalability in blockchains, is the number of transaction requests the system can accept. As we state in Chapter 4.2.1, throughput (γ) may vary according to Q, $\mathcal{T}$ and t_q. The difficulty of the Bitcoin proof-of-work is adjusted according to the hashing power of the entire Bitcoin network so that the average time to mine a block, $\mathcal{T}$, will always be around a certain value. In our analysis, we prove that $\mathcal{T}$ stays stable for the whole time and this value is set at 10 minutes. In this interval of time, since the block size Q is set to a limit of 1 MB, the system can approximately approve, if we consider an average t_q of 500 bytes, 3.3 transactions per second. This value is calculated considering the maximum possible block size of 1 MB. If we consider the time between 2013 and 2017, the number of daily transactions that need to be approved scaled from fifty thousand in 2013 to three hundred thousand in 2017, and all blocks are filled over their maximum capacity since April 2016, as Figure 4.4 shows. Because of this scaling limitation, transactions might be willing to pay more fee in order to get accepted in miner's mempools. To improve the number of transactions that can be accepted per second by the system, we consider an eventual change of $\mathcal{T}$ and Q, since we analyzed that the average t_q is constant and equal to 500 bytes for the whole period of

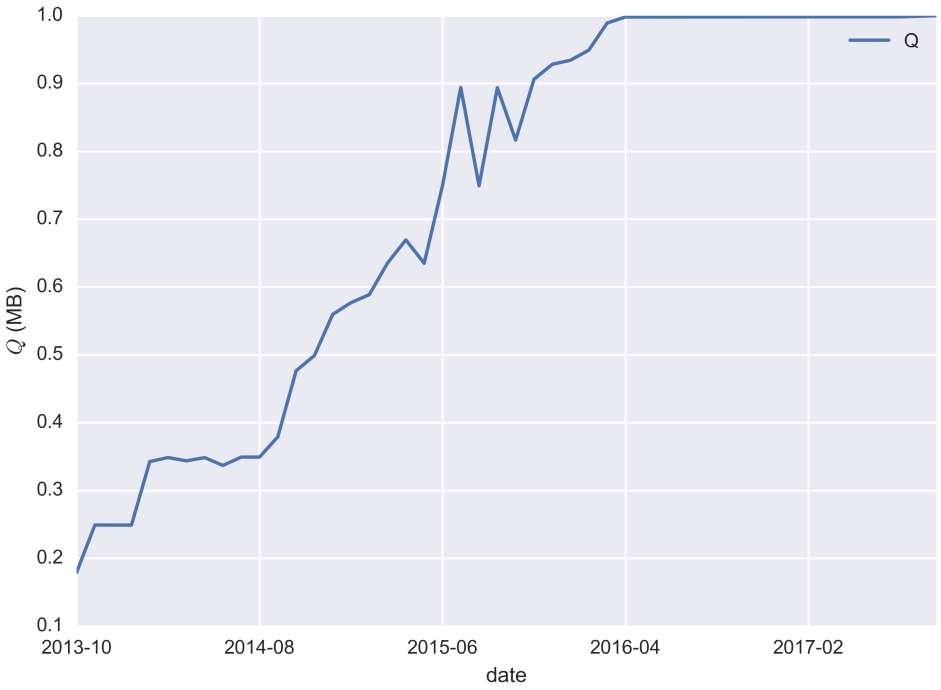


Figure 4.4: Average block size every month, from 2013 until 2017.

testing.

Block Creation Time $\mathcal{T}$

In our assumptions, listed in Table 3.2, we argue that an increment of $\mathcal{T}$ may lead to a drop in γ and an increment on t_l, for that reason we believe that it is a better choice to adjust the difficulty of the proof-of-work in order to have a shorter block creation time rather than increasing it. Furthermore, according to Equation 2.2, the cost of mining a block is directly proportional to the block creation time, which means that if the $\mathcal{T}$ is decreased, the cost of mining a block is less. Thanks to our data frame D, we managed to prove that there is an inverse proportionality between γ and $\mathcal{T}$. Only in blocks with a creation time lower than 10 minutes, the throughput reaches peaks of 10-14 transactions per second, while in cases from 10 to 20 minutes of creation time, the throughput is between 2-4 transactions per second and in blocks where the $\mathcal{T}$ is higher than 20 minutes, throughput never reaches 2 transactions per second.

Our assumptions about transaction latencies were partially right. It is true that t_l tends to grow along with $\mathcal{T}$, but this is true only if $\mathcal{T} \geq 10$ minutes. If the block is mined before that time then we do not have any relation between

transaction latency and block creation time. We suspected that this uncertain relation might be somehow connected with who mined that particular block. While mining pools may have different criteria (higher transaction fee) while appending transactions to be mined, occasional miners might have none, or less. This is the reason why the t_l is so random if t is mined by an occasional miner, since transactions are not selected according to any criteria, so a transaction t that did not offer any fee and has been waiting to be approved for long time, might be included by some occasional miner if it luckily wins the proof-of-work before the 10 minutes time. After some tests, we show the results in Figure 4.5. As suspected, for $0 \leq \mathcal{T} \leq 7$, the occasional miners take the 21.3% of mining share, while in the other portions they never reach the 18% of the total blocks mined.

In conclusion, an increment of $\mathcal{T}$ is ill-judged, while decrementing it should be considered, acknowledging, according to Equation 2.4, that an increased bandwidth for a better inner node communication is needed, otherwise we might have an exponential growth on $\mathbb{P}_{orphan}$. Furthermore, in Chapter 4.3.1, we state that with a higher $\mathcal{T}$ the profit $\langle \Pi \rangle$ of a miner is significantly lower.

Block Size Q

If the creation time $\mathcal{T}$ tends to have a fixed value, then Q plays a big role in throughput limitations. Discussions on whether to increase the block size limit or not are still ongoing and this issue has been brought up in several papers [40, 21]. Increasing the block size limit leads to a bigger amount of transactions that could be accepted at every block creation time, so we may think that this could be the right solution to adopt in order to increase the performance of the system. Anyway there is a big controversy on this topic, and even the biggest entities in the Bitcoin system do not agree to a solution. Some such as CoinBase, Blockchain.info or BTC Guild, deem that is beneficial to increase the block size, some others like F2Pool or Ethereum are neutral and other entities, for example GreenAddress, Bitrated or Bitcoinpaygate, think that this is just pushing the problem a little further and is not a good solution. In our assumptions we also state that transaction fee t_f may decrease with an increment of the block size limit, since transactions do not compete for space in the block anymore. So a small block size will require higher fees for fast confirmation, which will bring the following pros (+) and cons (−) [16]:

+ It will no longer be cheap to spam transactions such as Satoshi Dice bets [51].

+ Fees will not be zero. This is eventually a necessity in order to encourage

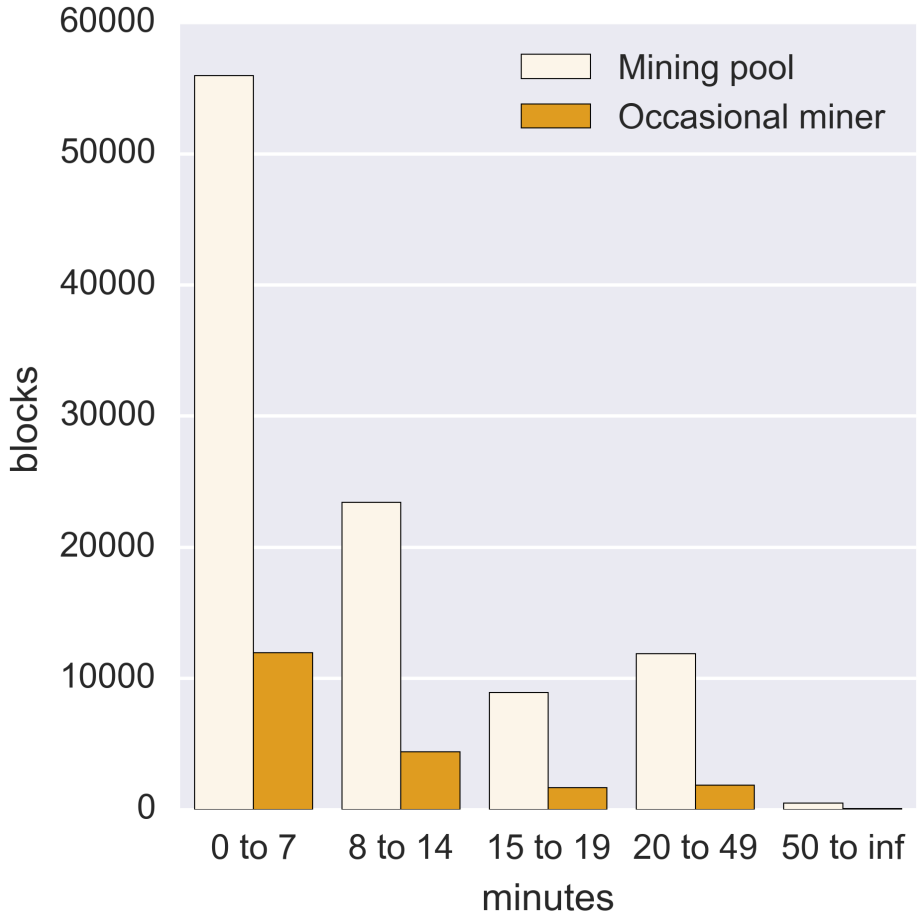


Figure 4.5: Number of blocks mined by mining pools and occasional miners according their creation time.

miners and secure the mining ecosystem.

– Bitcoin may look unattractive to new users, having high fees.

– Users pay higher fees.

We tested not only that the t_f is related to the increment of the block size limit, but thanks to our evaluations presented in Figures 4.11-4.12, we can state that from when the blockchain started to become saturated in mid 2016 (Figure 4.4), t_f but also the fee density ρ had a big increment, which is a sign that the blockchain is needing an urgent change according to avoid scalability problems. Increasing the block size limit is one of the possible solutions, but we have to consider that it might not only be good for users,

but also bad for miners. The cost of mining increase along with the block size, so receiving no more fees from users would be costly effective and lead to a centralization problem, since larger blocks make full nodes more expensive to operate, having then less hashers running full nodes, discouraging in that way miners to join the network. If we consider now the statement that block size limit should be increased, these are the arguments against it:

- Orphan rate amplification.
- Damage to centralization.
- "Congestion" concerns can be solved with mempool improvements including transaction eviction.
- No amount of max block size would support all the world's future transactions on the main blockchain (various types of off-chain transactions are the only long-term solution).

In conclusion, even if we think that an immediate increment of the block size is not the final solution to the scalability problem, we believe that an immediate and temporary solution should be considered to avoid network congestion, too high t_l and drastic increment of t_f. This temporary solution might be the block size limit increment, especially if big mining pools such as F2Pool claim that they could support a maximum block size of 20 MB [16].

4.2 Performance

Together with scalability, performance is one of our main subject to study on Bitcoin and we can say that it is highly related with scalability. In Chapter 4.1.2 we already talked about throughput γ and how it is related to the block size Q and block creation time $\mathcal{T}$. We are now going to focus on the Bitcoin network performance from both, user side (t_l) and system side (γ). Moreover, our target by studying transaction latency t_l is to verify our assumptions whether there is any relation between t_f and t_l during the whole period analyzed. These tests are strictly related to the cost and fee problems (Chapter 4.3), and are important to make users aware whether is possible to pay for bandwidth in Bitcoin system, and if yes, how to optimize their fee, t_f, according to get faster t_l.

4.2.1 Throughput γ

As we state in Chapter 4.1.2, γ is highly dependent on Q and t_q, but also related to $\mathcal{T}$. Block size limit can set the number of transactions the network can confirm per block, thereby the throughput of the whole Bitcoin network. In Figure 4.6 we show how throughput has changed during the years. We want to clarify that in 2013 the system was not performing less, there were just less transactions to process, roughly fifty thousands per day, and only when the blocks started to get saturated in mid 2016, the system began to use its maximum throughput capacity of ~3-4 transactions per second. However, if compared to centralized trusted-oriented system, throughput is extremely low. Assumptions in Table 3.2 about throughput were correct, and in order to have a higher throughput, Q needs to be bigger, $\mathcal{T}$ lower, or both. However, an increment on γ will bring a raise on the cost $\langle C \rangle$ if Q is changed, and a growth of $\mathbb{P}_{orphan}$ with also repercussions on $\langle C \rangle$ if $\mathcal{T}$ is lowered.

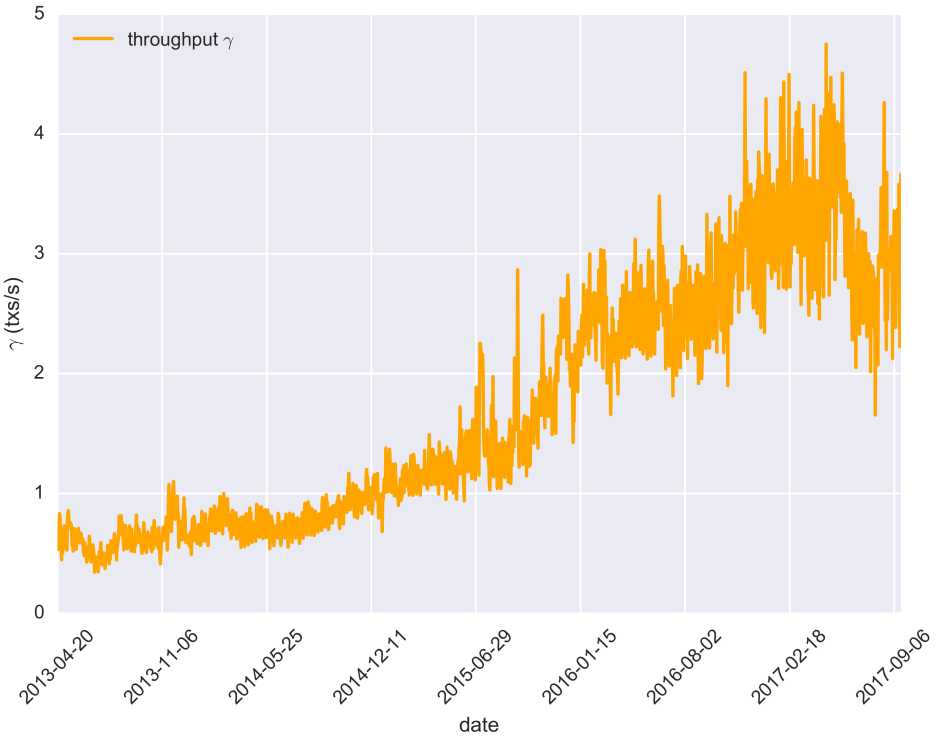


Figure 4.6: Throughput (γ) of the Bitcoin blockchain calculated in a period from 2013 until 2017.

4.2.2 Transaction Latency t_l

Another important aspect of performance regards the bandwidth that Bitcoin could offer to users. We considered the latency t_l of every transaction, which is the time necessary for t to be approved by the network. We want to find out how and why t_l and t_f changed during these years pointing out, if any, the relation between these two attributes, so that users are aware while paying a certain t_f, in how much time approximately their transactions will be approved and confirmed by the network. We observed that from 2013 to 2017, while the average transaction latency did not vary much and was stable at ~15 minutes, the fee paid from users to the miners has undergone a relevant raise, from an average of 0.0001 ฿ to 0.0004฿, with peaks of 0.0012 ฿ in 2017, where the Bitcoin price is also 2000 times higher than in 2013. In our assumptions (Table 3.2) we show how t_l could be affected by other attributes in the system. Assumptions on τ and γ are immediate, so the transaction latency will increase if the propagation time grows and will decrease if the throughput gets higher. Other assumptions required a deeper study, for example, we analyzed the relation between block size Q and transaction latency t_l. We divided the block size in categories, then plotted these categories in relation with t_l and grouped the whole dataset by year, Figure 4.7. As we suspected, during the years, whenever there was an increment on the block size, transaction latency had a drop. In 2016 when we had the final increment to 1 MB, the latency started to become incredibly low, while in 2017 when the blocks started to get saturated we have a latency up to 9 times higher, plus, smaller blocks in 2017 verify a much higher transaction latency since they include less transactions than the amount that need to be approved.

Not much intuitive and more difficult to study was the relation between t_l fee and t_f. According to the analysis made on t_f we categorized the fee paid from users to miners and then we split the data frame per year. Table 4.3 enhance this categorization and Figure 4.8 shows the plot generated while analyzing more than one hundred twenty million of transactions from 2013 to 2017. As Table 4.3 and Figure 4.8 enhance, the fee paid from users has a bigger impact year by year on the available bandwidth the system offers. Our assumptions about this relation were right, and especially after the blocks started to get saturated in mid 2016 there was big change in the way miners include transactions. If during 2013 and 2014, miners and mining pools tended to include in blocks a compromise between fee and waiting time, in 2017 miners tend to completely ignore 0-fee transactions. This could be good to avoid transaction spamming and it is good for miners that are gaining more from mining, since the block reward is halved every two hundred ten thousand blocks. On the other hand, impatient users need to pay a higher fee to see their transactions approved faster. Furthermore, we can state that in 2017, 0-fee transactions have a latency of ~33 hours, 4 times bigger than 2016, 8 times than

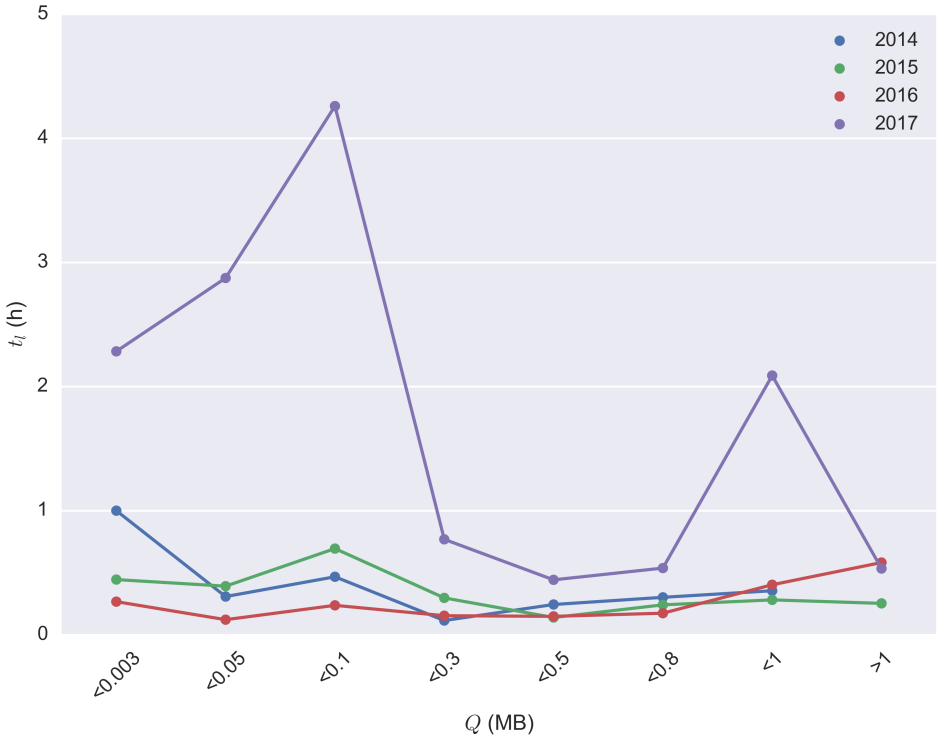


Figure 4.7: Relation between t_l and Q from 2014 until 2017.

2015, and more than 16 times bigger than 2013 and 2014, which means that the problem of block saturation has been partially considered by some miners with 0-fee transactions eviction. Today miners are using fork mechanism such as *BitcoinABC* [47] and they tend to exclude 0-fee transactions, so users should be aware that, a 0-fee transaction might not be mined at all.

4.3 Costs and Fees

This Section aims to show the changes in tips and tolls in Bitcoin system from 2013 to 2017. Möser and Böhme [34], claim that the hard size limit still does not affect the level of the fee paid. We want to enhance instead, that nowadays the saturation problem significantly drives the fee paid from users in order to get more bandwidth or to be accepted at all. Furthermore, at the time of analysis (2015), Bitcoin claimed that the fees were lower than 0.1% while today we analyzed (Figure 4.15) to have an average percentage of 0.25%, which is more than the double. Plus we want to consider the mining cost for miners, from what it is driven and how it will change in the past years. When

Table 4.3: Table representing the results in Figure 4.8, showing how t_l might vary for each category of fee.

Fee category	t_f **value (฿)**	t_l **variation (h)**
0	$t_f = 0$	1 to 33
<0.0002	$0 \leq t_f < 0.0002$	0+ to 5+
<0.0004	$0.0002 \leq t_f < 0.0004$	0+ to 2.5
<0.0006	$0.0004 \leq t_f < 0.0006$	0+ to 1.5
<0.0008	$0.0006 \leq t_f < 0.0008$	0+ to 1
<0.001	$0.0008 \leq t_f < 0.001$	0+ to 1
<0.01	$0.001 \leq t_f < 0.01$	0+ to 1
<0.1	$0.01 \leq t_f < 0.1$	0+ to 1.5
>0.1	$t_f \geq 0.1$	0+ to 2

we talk about fees and tolls we might refer both to miner's profit and users' benefits.

4.3.1 Miners' Profit

The study of how profitable is mining for miners, is complex and a lot of data analysis and manipulation is required. Knowledge regarding electricity consumption, mining hardware, and cost of hashing is needed, and a deep study and data analysis on how the Bitcoin hashing rate and the Bitcoin price in USD changed during the years has to be taken into consideration. Making assumptions about mining costs/profits might be difficult due to the information withheld from mining pools. We think though, that this is an important concern and it has to be addressed and studied, since the block reward is even halved every two hundred ten thousand blocks, and miners should find another way to profit from mining. Solutions for miners might be to additionally charge users with higher fees or to optimize their mining profit $\langle \Pi \rangle$ by analyzing the costs $\langle C \rangle$ and the revenue $\langle V \rangle$ over time. We refer to Equations 2.2, 2.3, 2.5 in order to calculate $\langle C \rangle$, $\langle V \rangle$ and $\langle \Pi \rangle$. For the reckoning we had to adapt our data and formulas to the Bitcoin price and Bitcoin total hash rate H at the time when data were generated. To do so, we extracted our time information from D, collected JSON data regarding H and Bitcoin price from `blockchain.info` and finally fitted them in order to get $\langle \Pi \rangle$, $\langle C \rangle$ and $\langle V \rangle$. Equation 2.5 gives us information in order to calculate $\langle C \rangle$ and $\langle V \rangle$, and we use a propagation time of $\tau = 15.7$ seconds, considering in that way a scenario with an average block size of 1 MB and a 50% propagation; in addition, we contemplate an electricity cost of 0.04 \$/kWh (Chinese price, since AntPool is a Chinese mining pool), and examining the most powerful mining hardware on the market for Bitcoin, AntMiner S9 [54], having a hashing power of $h = 14,000,000$ MH/s and a

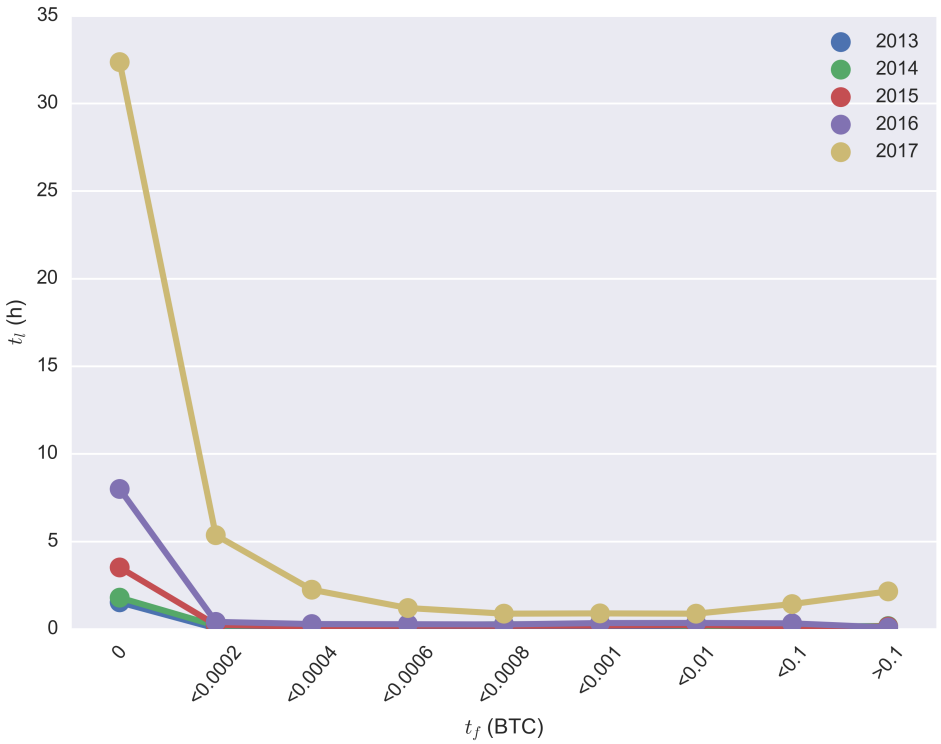


Figure 4.8: Relation between t_l and t_f grouped by year.

consumption of 1375 W, with an overall cost per hash of $\eta = 1.091 \times 10^{-18}$ \$/H. A similar analysis on electricity consumption and cost of mining was made by Croman et al. [14] in 2016. For our calculations, we only considered transactions from late 2016 and 2017, when Bitcoin's price was already over 1000 \$, before that, data might contain too many outliers due to huge differences in Bitcoin's hashing rate and Bitcoin price, and they could drive the analysis to an incorrect status. Figure 4.9, shows how the miner's profit is related with the block creation time and how, after 15-20 minutes, the miner starts to decrease its profit. This is caused by the constant growth of the cost necessary to mine a block, $\langle C \rangle$, which is directly proportional to the cost per hash η and the block creation time $\mathcal{T}$. Is important to enhance that Figure 4.9 shows the profit $\langle \Pi \rangle$ for a single miner, using AntMiner S9, and that in a real case scenario we might have hundreds of computers working together, so the possibilities to find a new block are much higher and the revenue $\langle V \rangle$ is accordingly bigger, having then much greater profit that needs to be shared among all the miners.

Furthermore, as we will see in Chapter 4.3.2, miners most likely include transactions with a higher fee density (ρ), so if users are not willing to pay more fee, once all the best ρ-wise transactions are already included in the new

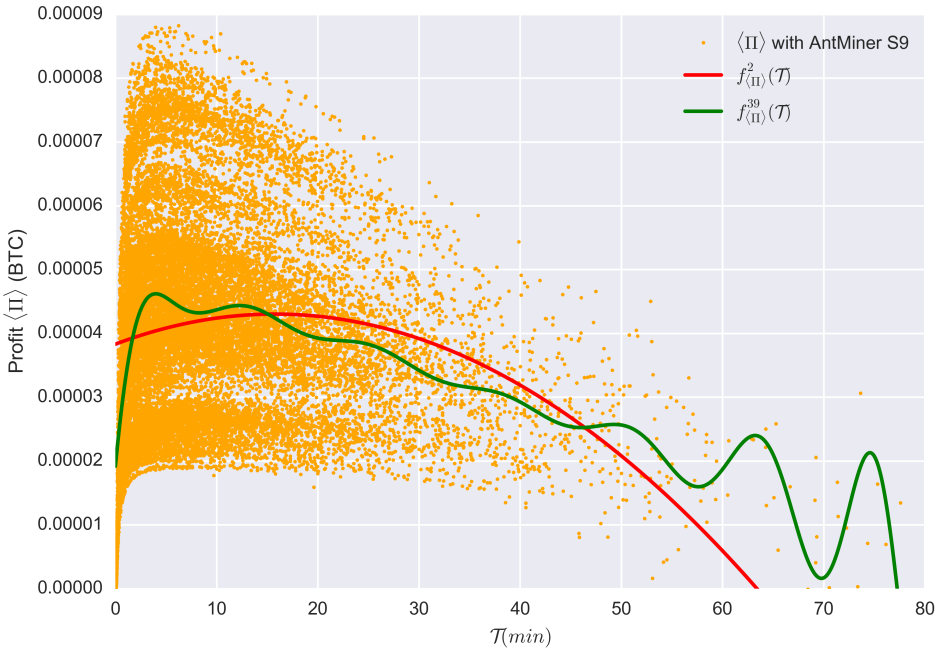


Figure 4.9: Profit $\langle\Pi\rangle$ in relation with block creation time $\mathcal{T}$ while mining with AntMiner S9 from 2015 until 2017.

upcoming block but Q is not reached yet, miners have to include lower fee density transactions, having in that way a loss in their profit. So if we look at it from a miner's prospective, the increment of Q might not be the best choice in the long run, unless the fees are standardized in a way that every miner has a guaranteed profit. If back in 2009-2013 the block revenue was poorly influenced by user's fees and it relied more on the reward R, we can see instead in Figure 4.10, that R is decreasing while the fees are getting higher, and we expect M to overcome R by 2020 when the reward is halved again at 6.25 ₿. We have seen that the block profit $\langle\Pi\rangle$ is highly influenced by users' fees t_f, unfortunately, more miners will request for fee, more users are discouraged to pay with Bitcoin. Furthermore we noticed a strong link between the halving of R in mid 2016 and the increment of t_f. Because of that we analyzed transaction fees offered by users to see how they are influenced in paying more or less ₿ to the miners during they years.

Miner Profit Function

To have a mathematical idea of what is happening between miner's profit and creation time, we define $f^{2}_{\langle\Pi\rangle}$ as the *Miner Profit Function*, and represents it in Equation 4.1, which is inferred after our analysis on almost twenty-five

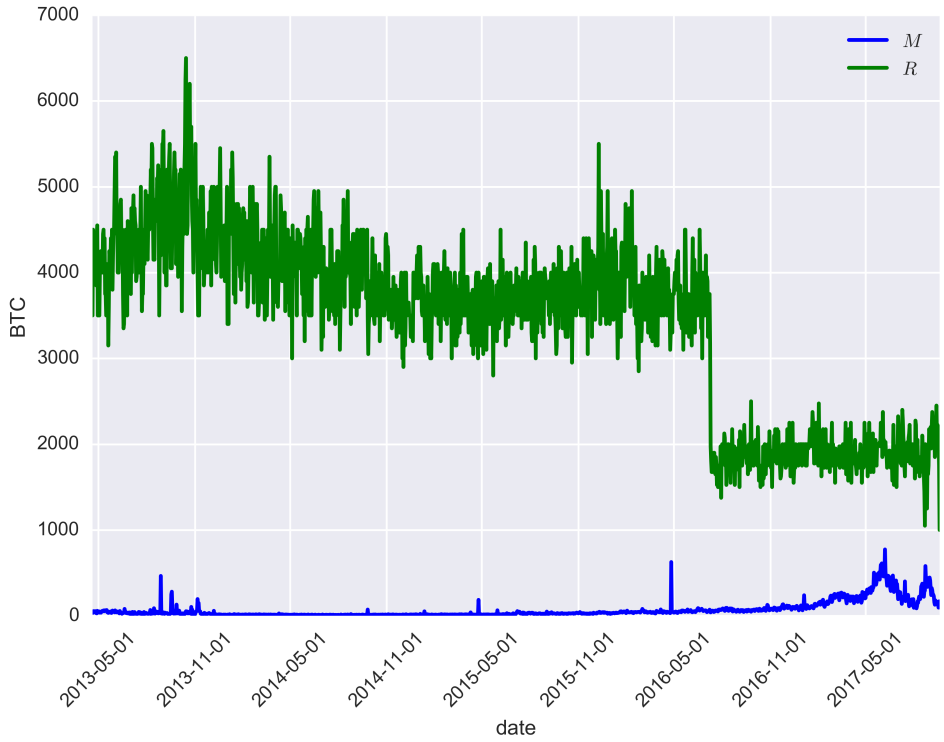


Figure 4.10: Miners revenue $\langle V \rangle$ divided in block reward R and sum of all t_fs in a block, M, analyzed between 2013 and 2015. Data are represented daily, and the R and M values are sums of every specific day.

thousand blocks, considering a single miner using AntMiner S9, and having $\mathcal{T}$ in minutes as input.

$$f^2_{\langle \Pi \rangle}(\mathcal{T}) = -\frac{1.896}{10^8}x^2 + \frac{5.977}{10^7}x + \frac{3.831}{10^5} \quad \Big\{ \text{with } 0 \leq \mathcal{T} < 80 \tag{4.1}$$

To optimize the profit $\langle \Pi \rangle$, according to $f^2_{\langle \Pi \rangle}$, the difficulty should never be increased in a way to have a creation time $\mathcal{T} >> 20$ minutes otherwise miners have a loss in their profit, and for miner's sake, should be even better to slightly decrease $\mathcal{T}$ according to avoid useless computation that leads to a higher cost and waste of electricity. We also generate, to have a more accurate trend, $f^{39}_{\langle \Pi \rangle}$, which is also shown in Figure 4.9 but not mathematically represented, being a 39 degrees polynomial. We could use this trend though to assume that the real profit occurs between 3 and 8 minutes, while having less or more creation time the $\langle \Pi \rangle$ will not be optimized.

4.3.2 Users' Benefits

In order to make assumptions and conclusions on how users should use their fee to optimize their bandwidth in the system, we first analyze how the t_f changed during time. Figure 4.11, shows the numerical attribute t_f divided into categories and represented in percentage for each category. We can notice that, after the first half of 2016, fees between 0 ₿ and 0.0002 ₿ almost disappeared from the system, and considering that the Bitcoin price raised from less than 1000 $ to more than 5000 $ between mid 2016 and second half of 2017, this results to be as a huge increment on the fees paid to miners, especially if we consider that, if we compare the Bitcoin price and the fee paid in USD, we see a substantial co-movement which indicates that BTC is the dominant unit of account when deciding about the fee offers. Is possible also to see this increment in Figure 4.10, indeed the total t_f paid to miners almost reaches 1000 ₿ per day 2017. We can also see that we had higher fees early in 2013, but at the time the USD price for Bitcoin was less than 120 $ so respectively the fees were much lower. We believe that, events happening on the Bitcoin blockchain might influence fee, tolls and the way blocks choose transactions, and to confirm our assumptions we see that the biggest increment in t_f was in late 2016, which coincides with the halving of R and the blocks saturation. To see how miners change their behavior according to events happening on

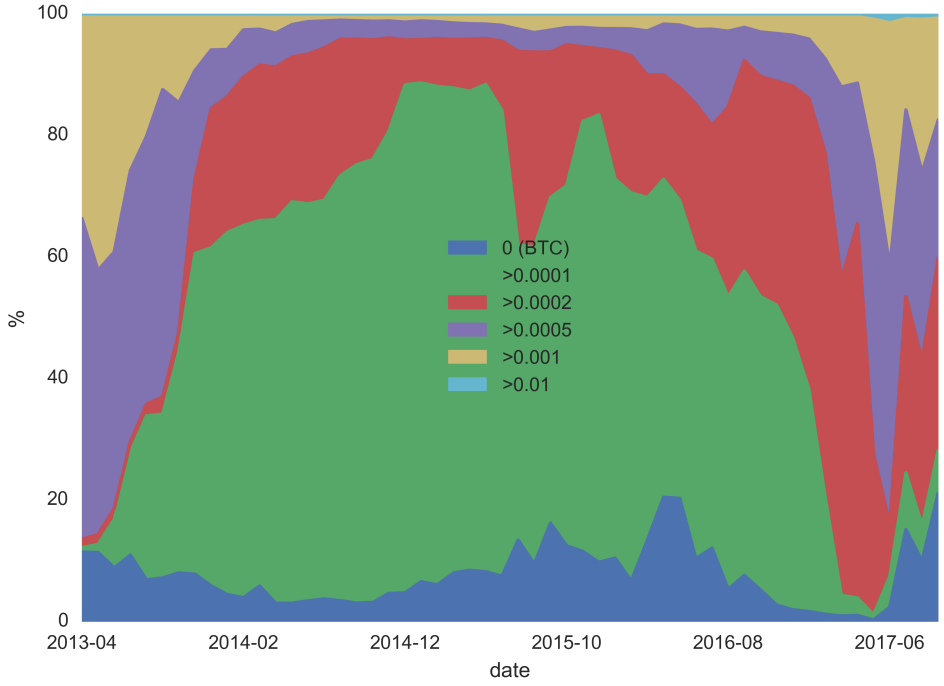


Figure 4.11: Transaction fee (t_f) distribution during the years from 2013 until 2017.

the Bitcoin system we study the fee density ρ, defined in Equation 3.6 and displayed in Figure 4.12. Fee density and fee are directly connected, since the transaction size t_q has an average of 500 bytes, but at the end of 2017, even if we have some fees with < 0.0001 ₿, we almost never have transactions with a $\rho = 0$, which means that recently, miners might have changed from having constraints on fees to having constraints on fee density. This drastic change in

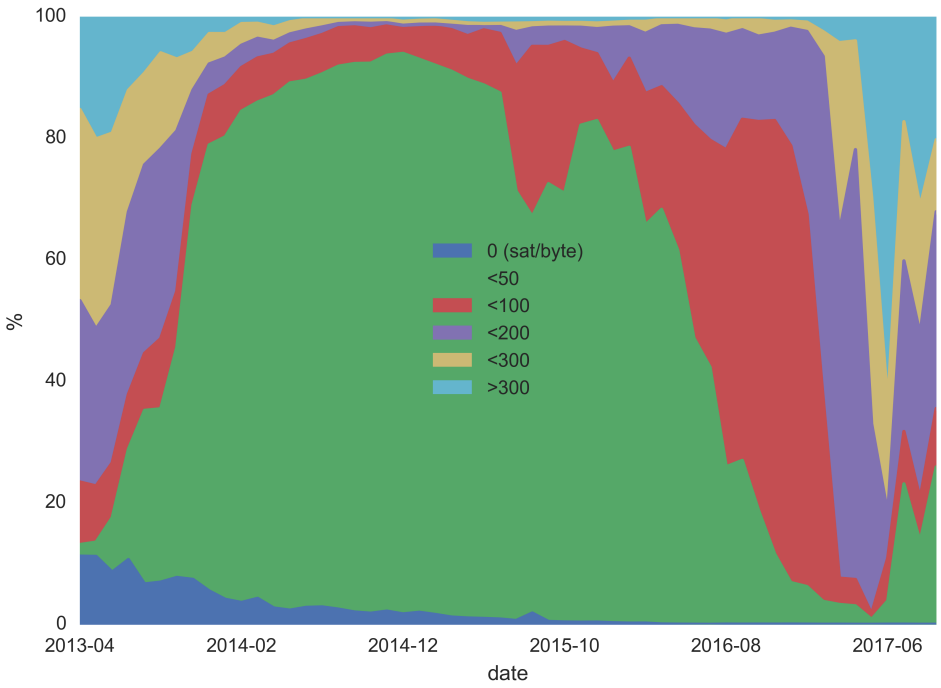


Figure 4.12: Fee density (ρ) distribution during the years from 2013 until 2017.

t_f and ρ lead us to think that there is another factor which is influenced by it, and in our assumptions we thought it was the transaction latency t_l. This is the reason why, we analyzed t_l in function of t_f from 2013 to 2017 (Figure 4.8) for an overall trend, and then t_l in function of t_f and ρ for transactions occurred in 2017 to generate our prediction models. As also Figure 4.8 shows, is not good to invest in so much high fees, since after a certain threshold, the latency would be even higher. With our models we want to define these thresholds and give users some ideas on how they might invest wisely their money, according to optimize their bandwidth. With data regarding transactions evaluated in 2017 we generate two models, *Fee Density - Latency Function*, $f_{t_l}(\rho)$, represented in Equation 4.3, and *Latency Function*, $f_{t_l}(t_f)$ presented in Equation 4.2. In Figures 4.13, 4.14, due to our interpolation function's boundaries, we show two different degrees of polynomial interpolation, $f^2_{t_l}$ and $f^{39}_{t_l}$. The purpose of $f^2_{t_l}$ is to have a general equation to refer on while talking about fees and latency, while the $f^{39}_{t_l}$ gives us more strict thresholds about the trend our data tend to

follow, for example in Figure 4.8 we see that in 2017, if you pay more than 0.001 ₿, you most likely get an increment in your transaction approval time, and in Figure 4.13, $f_{t_l}^{39}$ shows this threshold while $f_{t_l}^{2}$ does not.

$$f_{t_l}^{2}(t_f) = 6248x^2 - 555.8x + 1.42 \tag{4.2}$$

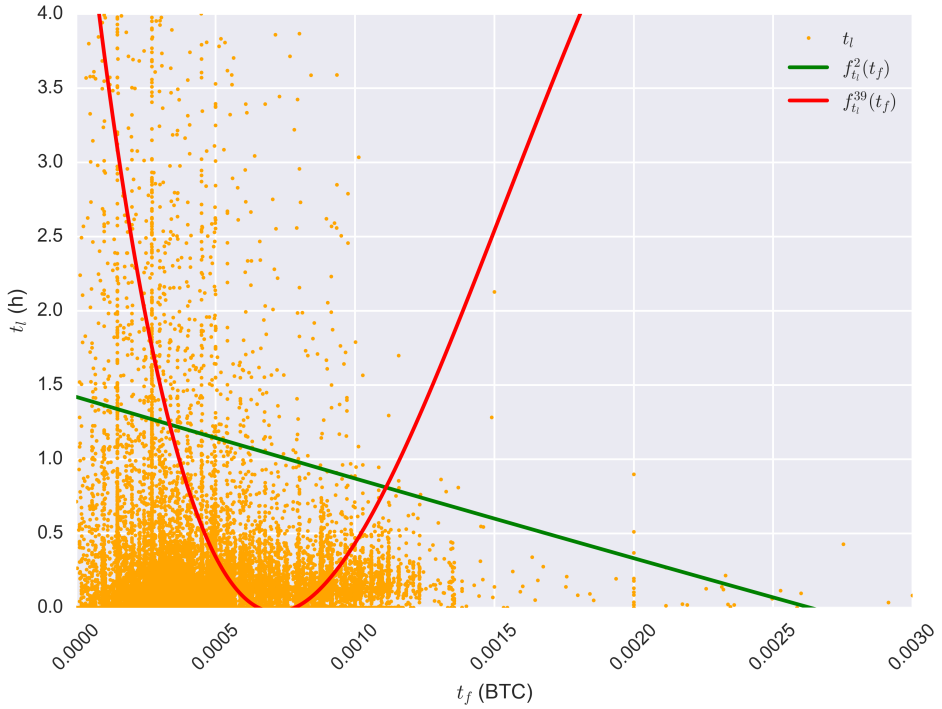


Figure 4.13: Interpolation with a 2 and 39 degrees polynomial of the relation between t_f and t_l, for transactions analyzed in 2017.

$$f_{t_l}^{2}(\rho) = \frac{5.416}{10^8}x^2 - \frac{2.215}{10^3}x + 1.598 \tag{4.3}$$

We can define in that way the thresholds in which our functions might work and they might be the following:

$$0 \leq t_f \leq 0.0011,$$

and

$$0 \leq \rho \leq 460.$$

If the $f_{t_l}^{2}$ function is used to make predictions and your t_f and ρ are outside this thresholds, then predictions might not be accurate and plus, you might be losing more money while getting even an higher latency. In conclusion, with data collected for the whole 2017, we can state that a user can definitely pay

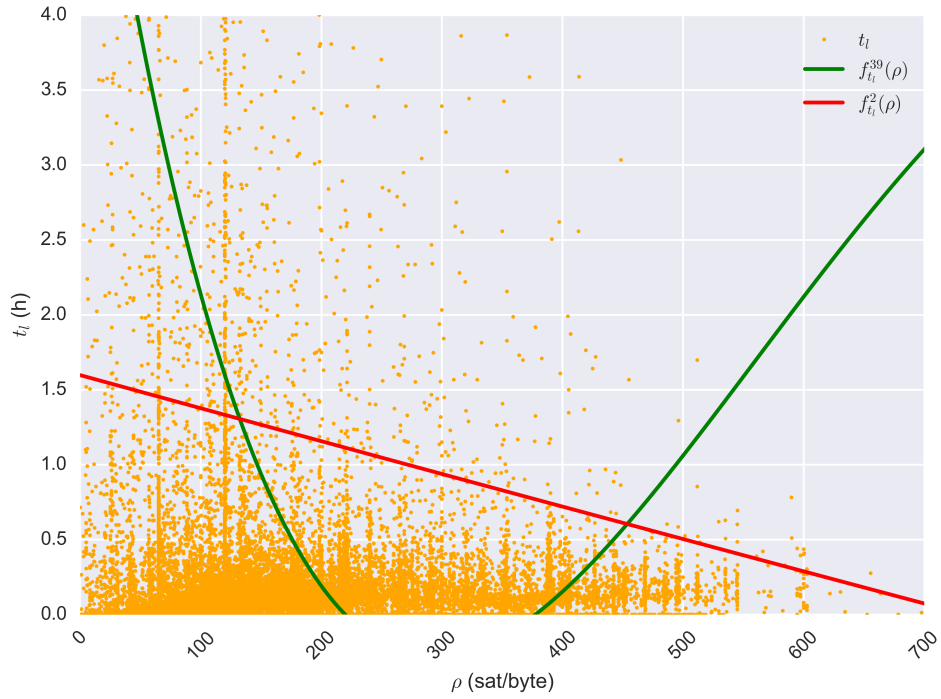


Figure 4.14: Interpolation with a 2 and 39 degrees polynomial of the relation between ρ and t_l, for transactions analyzed in 2017.

for bandwidth in Bitcoin's applications but it shouldn't be willing to offer more than 0.0011 ₿ or it might waste those money for no reasons. In addition, if nowadays miners tend to consider ρ rather than t_f, is good for users to know that an optimal ρ according to get higher latency would be between 200 and 300 sat/byte.

Now if we consider our t_f as a percentage of the total output t_{ou}, where $t_f\% = (t_f \times 100)/t_{ou}$, also the overall percentage $t_f\%$ had a consistent increment during the years and from all mining pools, like Figure 4.15 shows, going from less than 0.05% between 2013 and 2015, to more than 0.2%, reaching also 0.3% in 2017. Considering now that the Bitcoin price raised from 1000$ to more than 7000$ in the last two years, this increment in the $t_f\%$ results as a huge addition to the costs for users, and Bitcoin might not be as cheap as it claims to be.

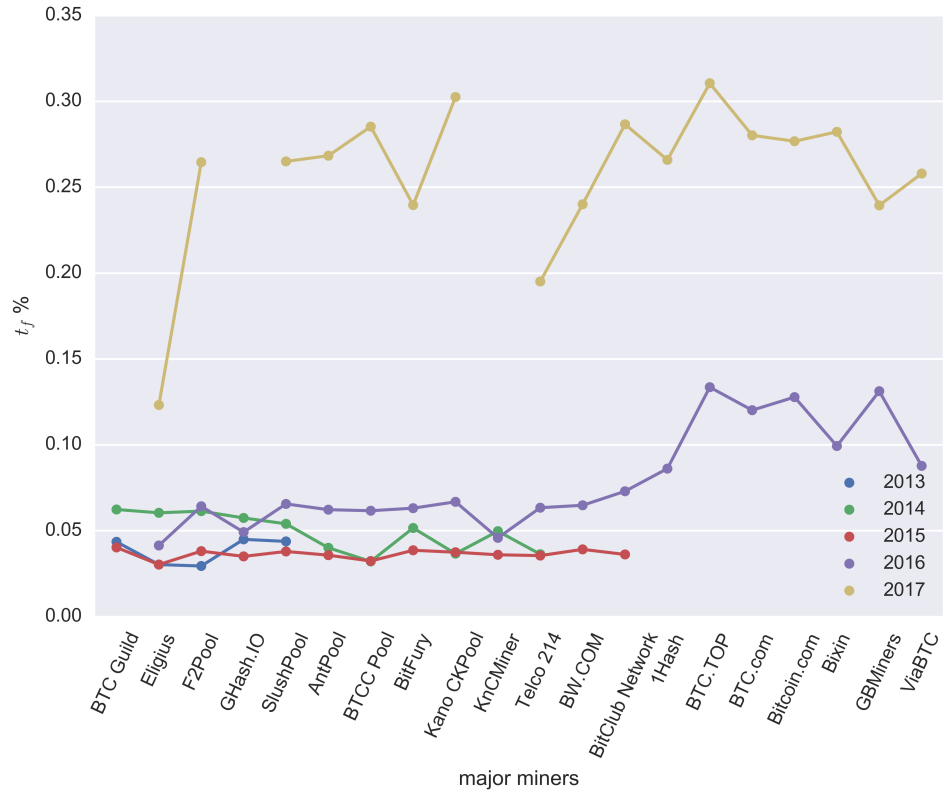


Figure 4.15: Average % of t_f paid from users to the top 20 miners of all times, grouped by year.

4.4 Is Bitcoin a Green-Wise Choice?

After calculating the miner's consumption related to the revenue we are concerned to know which is the impact of Bitcoin network on the environment. To know the electricity cost of mining in the Bitcoin network is necessary to know exactly how many miners are running and how much is their consumption. Unfortunately no one really knows how many miners are active and running in the network and this number changes every day, plus we don't have information about which miner's hardware is used but we could estimate they are in the order of hundred thousands, they group in mining pools and we could consider they are using the most powerful mining hardware on market at the time of analysis which is AntMiner S9. The problem of consumption is that, even if there are hundreds, thousands or ten thousands of miners in the network, they will approve anyway 3-4 transactions per second but they will use, accordingly, more consumption of electricity. Now, if we make assumptions on transaction consumption, considering a relatively low number of miners, with peaks of one hundred fifty thousand and all of them using AntMinerS9, we would have

a consumption between 400 and 500 Kwh per transaction, and if we grant that an average consumption of a house in the United States, which is one of the highest in the world, is of 11000 Kwh per year, consuming then about 30 Kwh per day, we can say that Bitcoin is anything but a green choice. The problem of Bitcoin consumption is that it only depends on how many miners are active in the network and if the Bitcoin price is rising then there could be more eager individuals willing to join, causing an increment of the difficulty to solve the proof-of-work so the system is anyways able to produce blocks every 10 minutes with a defined maximum size of 1 MB, but the overall consumption of the network will be higher. So every time the difficulty is raised also the consumption will increase accordingly. It might be good then to mine in places where the electricity is produced using renewable energy and also trying to avoid the inclusion of stand alone miners, otherwise they would work without ever mine a block, consuming a lot of energy. Even if our evaluation shows a downward estimation, the Bitcoin economy has a consumption only of a thousandth if compared to the whole oil industry per hour. We estimate for Bitcoin a consumption of 4.32 Gwh per hour while in the Statistical Review of World Energy from BP [5] they show that the oil industry has a consumption of 5600 Gwh per hour. Until Bitcoin would be that competitive for eager miners, then the difficulty is intended to grow leading the cryptocurrency into a really costly and consuming scenario, plus always less miners will be able to enjoy reward benefits and they will end up mining for nothing with more electricity consumption and consequence of that would be the increment of the fees, but that would attract even more miners and here we are at the earlier scenario. A solution to that might be to regulate the miners traffic by controlling and auto-adjusting the number of miners which can join the network, in that way the fees are contained and regulated, having also a content consumption.

5

Conclusion

This Chapter aims to highlight the most relevant observations while running BAS and to proof whether our assumptions were accurate or not, providing an explanation and also an estimation of how the system might change in the next years. We compare previous results with ours and we give opinions on the hot topics in Bitcoin, regarding the block size, the fees and miner's profit and then we discuss and test the accuracy of our prediction models. We evaluate that scalability issues brought to a centralization problem. When it comes about mining, the scenario changes from distributed to decentralized because of mining pools. Then we state that in order to increase the system performance and scalability an eventual change on $\mathcal{T}$ and Q should be considered with all the advantages and disadvantages that these changes might lead to. Finally we generate prediction models about the miner's profit and the transaction latency according to the fee paid and the fee density for each transaction, and this is profitable either for miners and users so the firsts need to have a guaranteed gain in mining while the seconds need to get a faster confirmation time by optimizing their fees.

5.1 Results

When Bitcoin was first implemented, one of its strength was the decentralization. Miners could join the network all over the world and more miners meant a more secure network. During the years though, the reward enticed

always more people around the world and new miners kept joining the network increasing in that way the difficulty to solve the puzzle, since the creation time should be kept fixed at 10 minutes. Consequently, the Bitcoin hashing power kept growing, so it became more difficult for single individuals to mine new blocks, decreasing in that way the chances for miners to gain their reward and leaving space in the network to mining pools, since they can gather up miners as well as their hashing power. This led to a scenario where singular individuals need to join bigger mining pools and the whole network changed from distributed to decentralized since mining pools are controlled by third parties society so they control a big portion of the hashing power needed to mine new blocks. Furthermore, large mining pools nowadays withhold information about number of miners, the hardware they use or they profit, making the centralization in the system even more enhanced. Being one of Bitcoin value proposition trustlessness, Bitcoin is only useful if is decentralized, since centralization requires trust. The other problem in scalability regards the amount of transactions to be approved every day by the Bitcoin system, and we have seen that this number is constrained by Q, $\mathcal{T}$ and t_q and since the transaction size will always be around 400-500 bytes, we consider a change in the block size Q and in the block creation time $\mathcal{T}$. Table 5.1 shows the results and our assumptions after a longitudinal study on the Bitcoin blockchain. Decreasing Q might appear a good solution for the number of advantages it has, but the only two disadvantages are critical for both performance and scalability matters, while it could be much easier to deal with the orphan rate amplification. For that reason a decrease on the block size is ill-judged. We have instead an opposite scenario with the block creation time. An increment would be ill-judged since the system will not be scalable, will be less performing and miners will have less profit while mining with the only advantage to have a lower orphaning rate.

As we can observe from Table 5.1, throughput γ increase when either the block size Q is raised or the creation time $\mathcal{T}$ is lowered, a good compromise of both might be the solution to part of the scalability and performance problems in the Bitcoin network. According to Croman et al. [14], the block size should not exceed 4 MB given a creation time of 10 minutes. We assume that a good compromise could be to increase the block size limit at 1.5 MB and lower the creation time at 8 minutes. In that way the system will perform an average of 5-6 transactions per second with a creation time of 10 minutes, reaching 10 transactions per second if the 8 minutes interval is respected.

While we can not define a relation between Q and t_f, since they are related only when a drastic change in the block size is made in the network, we can state that, from 2013 to 2017 the relation between t_f and t_l day by day more noticeable, having almost an inverse proportionality in the latest data from 2017 as displayed in Figure 4.8. We represent this relation with Equation 4.2.

Table 5.1: Scalability and performance scenario and consequences if block size Q and block creation time $\mathcal{T}$ are increased or decreased.

	Higher ↑	**Lower ↓**
Q	+ more scalability and transactions accepted per day + less latency t_l −/+ lower fees, good for users bad for miners − orphan rate amplification − damage to centralization − congestion concern solved with transaction eviction by miners − temporary solution	+ no transaction spam + no 0-fee transactions + less mining cost + less propagation time + less chance of orphaning −/+ higher fees, good for miners bad for users − less throughput − more latency t_l
$\mathcal{T}$	+ orphaning rate much lower + no physical changed needed to support faster inner node communication − lower throughput γ unless Q is increased − system not very scalable unless Q is increased − profit $\langle \Pi \rangle$ is confined	+ higher throughput γ + system is more scalable + profit $\langle \Pi \rangle$ much higher for miners +/− not much information about t_l but for sure it will not drastically increase − faster inner node communication is required with possible changes in the physical structure − exponential increment of orphaning rate

In 2017 0-fee transactions almost disappeared from the system and that is due to the incredibly high latency they were facing, since it took an average of 33 hours to get confirmed and included in a block by some miner. If only this fee is raised less than 0.0002 ₿, the transaction latency drops to an average of 5 hours, and with a fee between 0.0008 ₿ and 0.001 ₿ the average expected latency is less than 1 hour.

Regarding fees and tolls in the network we estimate the profit of a miner using AntMiner S9 [54] as mining hardware. We put in a relation the profit of a single block mined with its creation time. Once we collected enough data and calculated the profit, we inferred a possible trend for $\langle \Pi \rangle$ and called it *Miner Profit Function*, $f_{\langle \Pi \rangle}$. We used Numpy libraries [36] for the interpolation (Appendix C.10) and we perform a regression of 2 and 39 degrees function to see how the trend changes if the polynomial level is increased. Figure 4.9 shows both functions interpolated plus the samples for each block in 2017 and we define $f^2_{\langle \Pi \rangle}$ in Equation 4.1 while we use $f^{39}_{\langle \Pi \rangle}$ more like a reference to set our boundaries and see where the profit tend to increase or maintain its value after a certain amount of time. We state that for the first 2 minutes and after 20 minutes is not profitable to produce new blocks, while is good to have a creation time between 3 and 8 minutes. We finally tested the accuracy of our Miner Profit Function, $f^2_{\langle \Pi \rangle}$, using scikit-learn libraries [39] and calculating the MAE as shown in Appendix C.11. We considered then as predictors, our dataframe D, and we used as target values for accuracy tests, data retrieved on a newer portion of the blockchain for every blocks occurred between the 4^{th} and the 11^{th} of November 2017, and they turned out to have a median absolute error of MAE = 0.00002455 ₿.

The other two functions inferred are shown in Equations 4.2, 4.3 and they try to optimize users' bandwidth according to the fee these users are willing to pay for their transactions. Like in the Miner Profit Function, we evaluate a polynomial level of 2 and 39 degrees, and while the first one is the reference for our prediction models $f^2_{t_f}$ and f^2_{ρ}, the second one is useful to give us an idea about the boundaries of these models. Unfortunately, since rules behind the inclusion in a block are not yet well defined, and mining pools could change these rules, for example they possibly have changed the transaction eviction for 0-fee transactions to an eviction for the 0-ρ transactions, and the block is generated after a random number of minutes, the predictions on the available bandwidth for users are not as accurate as the one for the miners' profit, having an MAE = 1.2 hours in $f^2_{t_f}$ and an MAE = 1.03 hours in f^2_{ρ}. We ran these tests, as in the Miner Profit Function, in a newer portion of the blockchain, between the 4^{th} and the 11^{th} of November 2017. However, thanks to our f^{39} interpolation we can define the boundaries in which this functions are reliable, plus, Figure 4.8 gives a good overview of the transaction approval time

trend in relation to the fee paid and according to both measurements, paying more than 0.001 ฿ would be non profitable for users. Furthermore, we believe that nowadays miners changed their inclusion rules from a priority on t_f to a preference on ρ, this is the reason for us to infer also f_ρ^2. However, fee density is not dependent form users, since it is calculated considering also the transaction size t_q. For that reason, users should tend to consider a $t_f = 0.001$ ฿ as the limit for their fees, no matter how much is the amount transferred.

5.2 Discussion

Despite the withhold of information form mining pools and the difficulty in retrieving the blockchain, containing sometimes not convincing data, e.g. a negative creation time for some blocks mostly before 2013, we are satisfied about our studies and the results obtained. We managed to perform a longitudinal study on the Bitcoin blockchain, analyzing more transactions than any previous work, collecting recent data of 2017 and combining different data sources to get even more information, all stored in our data structure that aims to give all the information regarding blocks and transactions, saving up 10 times the amount of disk space. We could also, using machine learning techniques, infer useful functions effective for both users and miners, in order to gain more profit out of the mining process. Creating our data analytics system might not have been the easiest solution, but it was the one which allowed us to retrieve the exact information we wanted without storing the whole blockchain. We had trouble using Bitcoin's API since the website was not always available plus it prevents from having too many requests at the same time; because of that we spent more than a month to be able to retrieve all the information we wanted and to find a way to store them nicely in a way to be easily retrieved. With all the data collected and our data structure D we were able to verify our assumption on scalability, performance and finally to generate prediction models that even if are constrained and reliable only for a small interval of data input, they give an idea of how entities should behave in Bitcoin system.

Despite we inferred our models, giving information on how miners and users could use their time and money, the creation time $\mathcal{T}$ is still a random process, determined by the proof of work, and how lucky miners are to find the solution. Because of that, paying so much fee, t_f, will not guarantee an extremely fast execution, but at least your transaction will be considered as "high priority" in more mining pools according to the amount of its t_f.

It is also important to note that the problem of scalability has been taken into serious consideration and nowadays smart contracts like RSK [28] are

emerging, claiming to scale almost as much as PayPal even though they are not still in common use and maybe more testing on those will be necessary.

And finally, the problem of consumption is also urgent, moving also prominent characters such as Bram Cohen (inventor of BitTorrent) to launch new green cryptocurrencies such as *Chia* [38], since he claims that a Bitcoin transaction wastes as much as electricity as it takes to power an American home for a week, and we tested that it might be true, comparing the Bitcoin system to the whole oil industry.

Another issue while performing the longitudinal study was the pricing and money value of Bitcoin. It's hard to make prediction when the currency is affected by such high volatility. Usually the Bitcoin price tends to raise if there is an unstable situation in the global market and it frequently goes down when Bitcoin currency is involved in scandals such as the bankrupt of Mt Gox [37], once the biggest Bitcoin exchange on the market, money laundering and black marked related issue, since people are willing to spend less money in Bitcoin if something like that happens in the system.

5.3 Future Implementation

By profiling BAS we were able to understand what we could improve. For performance evaluation we used a line-by-line profiling for Python [26], displayed in Appendix D. We divided the computation in plot and retrieval, then retrieval in fetch and write. The fetching of data takes the 85.5% of the total time, while the writing of data only the 13.8%. The 99.9% of the computation in data fetching is due to the API calls on `blockchain.info`. For that reason, the system still suffers in performance but we could get faster latency in the future if we alter the primitives of data retrieval in a way to have bigger portions retrieved and less connections to establish.

Further analyses should take into consideration a more accurate estimation about the impact Bitcoin has on the environment. This could be done by studying the number of miners in the network and which kind of mining software is used, even if these data are kept secret by mining pools. The fact is, that Bitcoin has an high electricity consumption and it is intended to grow along with the Bitcoin hashing rate, so further studies and concerns about this topic should be considered and we believe that in the near future, the focus on cryptocurrencies will address in that direction.

A future useful feature of BAS might be a *short-time model prediction*. We thought about it as a retrieval of the last weekly transactions in the system,

then, a real time model is generated, giving to users information about the fee they should pay in order to optimize their latency at a reasonable costs.

References

[1] BTCC mining pool in Bitcoin network. https://www.btcc.com/.

[2] Coindesk, Bitcoin prices calculated every minute. https://coindesk.com.

[3] F2pool, mining pool in Bitcoin, Ethereum and Litecoin. https://www.f2pool.com/.

[4] Bitcoin mining process. http://bitcoinminer.com/, 2015.

[5] Bp: Statistical review of world energy. https://www.bp.com/en/global/corporate/energy-economics/statistical-review-of-world-energy.html, 2016.

[6] Tom Augspurger, Chris Bartak, Phillip Cloud, Andy Hayden, Stephan Hoyer, Wes McKinney, Jeff Reback, Chang She, Masaaki Horikoshi, Joris Van den Bossche, et al. Pandas: Python Data Analysis Library. software 0.21.0, Pandas community, 2012.

[7] Adam Back. Hashcash, a Denial of Service Counter-Measure. Technical report, 2002.

[8] Paul Baran. *On Distributed Communications: Introduction to Distributed Communication Networks*. The Rand Corporation, 1964.

[9] M. Bowles. *Machine Learning in Python: Essential Techniques for Predictive Analysis*. Wiley, 2015.

[10] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[11] Vitalik Buterin, Fabian Vogelsteller, et al. Ethereum's white Paper. https://github.com/ethereum/wiki/wiki/White-Paper.

[12] Rainer Böhme, Nicolas Christin, Benjamin Edelman, and Tyler Moore. Bitcoin: Economics, technology, and governance. *Journal of Economic*

Perspectives, 29(2):213–38, May 2015.

[13] David Chaum. Blind signatures for untraceable payments. In D. Chaum, R.L. Rivest, and A.T. Sherman, editors, *Advances in Cryptology Proceedings of Crypto 82*, pages 199–203, 1983.

[14] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. *On Scaling Decentralized Blockchains*, pages 106–125. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[15] Michael Crosby, Nachiappan, Pradan Pattanayak, Sanjeev Verma, and Vignesh Kalyanaraman. Blockchain technology: Beyond bitcoin, 2016.

[16] Luke Dashjr et al. Block size controversy. `https://en.bitcoin.it/wiki/Block_size_limit_controversy`, 2015.

[17] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10, Sept 2013.

[18] Rui Garcia, Rodrigo Rodrigues, and Nuno Preguiça. Efficient middleware for byzantine fault tolerant database replication. In *Proceedings of the Sixth Conference on Computer Systems*, EuroSys '11, pages 107–122, New York, NY, USA, 2011. ACM.

[19] Anders T. Gjerdrum, Håvard D. Johansen, and Dag Johansen. Implementing informed consent as information-flow policies for secure analytics on eHealth data: Principles and practices. In *Proceedings of the IEEE Conference on Connected Health: Applications, Systems and Engineering Technologies: The 1st International Workshop on Security, Privacy, and Trustworthiness in Medical Cyber-Physical System*, CHASE '16. IEEE, June 2016.

[20] Nermin Hajdarbegovic. Bitcoin Miners Ditch Ghash.io Pool Over Fears of 51% Attack. *Coindesk*, 2017.

[21] Nicolas Houy. The economics of bitcoin transaction fees. Working Papers 1407, Groupe d'Analyse et de Théorie Economique (GATE), Centre national de la recherche scientifique (CNRS), Université Lyon 2, Ecole Normale Supérieure, 2014.

[22] John Hunter. Matplotlib, Python 2D plotting. `https://matplotlib.org`, 2002.

[23] Håvard Johansen, Eleanor Birrell, Robbert Van Renesse, Fred B. Schneider, Magnus Stenhaug, and Dag Johansen. Enforcing privacy policies with meta-code. In *Proceedings of the 6th ACM SIGOPS Asia-Pacific Workshop on Systems*, APSys 2015. ACM, July 2015.

[24] Håvard D Johansen, Robbert van Renesse, Ymir Vigfusson, and Dag Johansen. Fireflies: A secure and scalable membership and gossip service. *ACM Transactions on Computer Systems (TOCS)*, 33(2):5:1–5:32, May 2015.

[25] Mark Karpelès et al. Proof-of-work. `https://en.bitcoin.it/wiki/Proof_of_work`, 2010.

[26] Robert Kern et al. line-by-line profiling for Python. `https://github.com/rkern/line_profiler`, 2016.

[27] Ravi Kumar, Ashwin Kumar, et al. Foreign exchange rate and currency conversion, based on `coindesk.com`. `https://pypi.python.org/pypi/forex-python`, 1999.

[28] Gabriel Kurman. Rsk: Rootstock smart contracts. `http://www.rsk.co/`, 2017.

[29] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[30] Aldelir Fernando Luiz, Lau Cheuk Lung, and Miguel Correia. Mitra: Byzantine fault-tolerant middleware for transaction processing on replicated databases. *SIGMOD Rec.*, 43(1):32–38, May 2014.

[31] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 254–269, New York, NY, USA, 2016. ACM.

[32] Trace Mayer et al. Bitcoin website mining. `https://www.bitcoinmining.com`, 2016.

[33] Clark Moody, Assaf Yossifoff, et al. Bitocoin blockchain analytic website. `https://blockchain.info`.

[34] Malte Möser and Rainer Böhme. *Trends, Tips, Tolls: A Longitudinal Study of Bitcoin Transaction Fees*, pages 19–33. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[35] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf, 2008.

[36] Travis E. Oliphant. NumPy, the fundamental package for numerical computation. https://docs.scipy.org/doc/numpy/index.html, 2008.

[37] Yessi Bello Perez on Coindesk.com. Mt gox: The history of a failed bitcoin exchange. https://www.coindesk.com/mt-gox-the-history-of-a-failed-bitcoin-exchange/, 2015.

[38] Josh Constine on techcrunch.com. Bittorrent inventor announces eco-friendly bitcoin competitor chia. https://techcrunch.com/2017/11/08/chia-network-cryptocurrency/, 2017.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[40] Peter R. Rizun. A transaction fee market exists without a block size limit. Technical report, 2015.

[41] Jeff Schvey and Greg Schvey. TradeBlock, the world's leading provider of institutional trading tools for digital currencies. https://tradeblock.com, 2013.

[42] Chaitya B. Shah and Drashti R. Panchal. Secured hash algorithm-1: Review paper. Technical report, Indus Institute of Technology and Engineering, Gujarat Technological University, 2014.

[43] David Siegel, Mark Ly, Greg Fraser, Scott Miller, Caleb Silver, et al. Definition of 51 attack. https://www.investopedia.com/terms/1/51-attack.asp, 2017.

[44] Nick Szabo. Smart contracts: Building blocks for digital markets. http://www.fon.hum.uva.nl/, 1996.

[45] Sarah Underwood. Blockchain beyond bitcoin. *Commun. ACM*, 59(11):15–17, October 2016.

[46] Wladimir J. van der Laan, Pieter Wuille, Gavin Andresen, et al. Bitcoin client application. Bitcoin Community, https://bitcoin.org/en/bitcoin-core/.

[47] Wladimir J. van der Laan, Pieter Wuille, Cory Fields, Gavin Andresen, et al. BitcoinABC. MIT - Massachusett Institute of Technology, https://github.com/Bitcoin-ABC/bitcoin-abc, 2017.

[48] Robbert van Renesse, Håvard Johansen, Nihar Naigaonkar, and Dag Johansen. Secure abstraction with code capabilities. In *Proceedings of the 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, March 2013.

[49] Ben Vandiver, Hari Balakrishnan, Barbara Liskov, and Samuel Madden. Tolerating Byzantine Faults in Transaction Processing Systems Using Commit Barrier Scheduling. In *ACM SOSP*, Stevenson, WA, October 2007.

[50] Valery Vavilov, Valery Nebesny, Jamie Elizabeth Smith, et al. The Bitfury Group, Leading full service in Blockchain technology. http://bitfury.com/.

[51] Eric Voorhees. Satoshi dice - Blockchain based betting game. https://satoshidice.com/rules, 2012.

[52] Michael Waskom, Olga Botvinnik, drewokane, Paul Hobson, David, Yaroslav Halchenko, Saulius Lukauskas, John B. Cole, Jordi Warmenhoven, Julian de Ruiter, Stephan Hoyer, Jake Vanderplas, Santi Villalba, Gero Kunter, Eric Quintero, Marcel Martin, Alistair Miles, Kyle Meyer, Tom Augspurger, Tal Yarkoni, Pete Bachant, Mike Williams, Constantine Evans, Clark Fitzgerald, Brian, Daniel Wehner, Gregory Hitz, Erik Ziegler, Adel Qalieh, and Antony Lee. seaborn: v0.7.1 (june 2016), June 2016.

[53] Dr. Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. Technical report, 2014.

[54] Jihan Wu. Antminer S9 - Bitcoin mining hardware. BITMAN, https://shop.bitmain.com/productDetail.htm?pid=00020171024170217293t9Sp5rm406A9.

[55] Jihan Wu. Antpool, mining pool in bitcoin network. BITMAN, https://www.antpool.com/.

A

Other Definitions

full node: in a decentralized digital currency peer-2-peer network, is a node that stores and processes the entirety of every block, storing locally the entire size of the blockchain.

light node: in a decentralized digital currency peer-2-peer network, is a node that only stores the part of the blockchain it needs.

satoshi: Unit of the Bitcoin currency. 100,000,000 satoshi are 1 ₿.

51% **Attack:** 51% refers to an attack on a blockchain, by a group of miners controlling more than 50% of the network's mining hash rate, or computing power. The attackers would be able to prevent new transactions from gaining confirmations, allowing them to halt payments between some or all users. They would also be able to reverse transactions that were completed while they were in control of the network, meaning they could double-spend coins. They would almost certainly not be able to create new coins or alter old blocks, so a 51% attack would probably not destroy Bitcoin or another blockchain-based currency outright, even if it proved highly damaging [43].

B

List of Symbols

t_B	number of transaction approved in a block B.
t_{in}	transaction input in bitcoin (฿). All the money sent.
t_{ou}	transaction output (฿). All the money received.
t_f	transaction fee (฿) (Equation 3.3).
t_q	transaction size, in bytes.
t_l	commit latency of a single transaction (seconds, minutes, hours) (Equation 3.4).
t_h	transaction height. Height of the block in which the transaction is included.
t_{ha}	transaction hash.

$t_{\%}$	percentage of t_{in} paid in fee, t_f.
$\mathcal{T}$	expected block interval time (~ 10 min)
$\mathbb{P}_{orphan}$	probability that given a block is orphaned.
τ	block solution propagation time, we consider a $\tau = 15.7$ seconds according to Croman [14].
t_{epoch}	timestamp of a transaction t. Epoch of when t was first seen in the network
η	cost per hash.
$\langle \Pi \rangle$	expectation value of a miner's profit per block.
$\langle V \rangle$	expectation value of a miner's revenue per block.
$\langle C \rangle$	expectation value of a miner's hashing cost per block.
R	block reward, currently at 12.5 ₿, halved every 210,000 blocks.
h	miner's individual hash rate.
H	total hash rate of Bitcoin network.
Q	block size or block space in bytes/MB.
Q^*	the block size that maximizes the miner's expected profit.
ρ	fee density, or the price per byte for block space. (sat/byte).

M	money in ₿. Sum of all t_f in a block.
$M_{demand}(b)$	partial sum of the b transaction fees in mempool in order of descending fee density.
$M_{supply}(Q)$	miner's cost due to orphaning to produce a certain block size Q.
$\mathcal{N}$	the set of transactions in a miner's mempool.
n	number of transactions in a miner's mempool.
B	single block.
B_t	transaction root that links to every transaction in a block B.
B_{epoch}	timestamp of a block B. Epoch of when the block was included in the blockchain
B_h	block height.
B_{ha}	block hash.
B_{mi}	miner which mined the block B.
γ	throughput of Bitcoin network, measured in transactions per second.
D	data frame generated by BAS.

C

BAS Source Code

The complete source code can be found at: `https://github.com/ted92/Bitcoin-Analytics-System`.

Listing C.1: Creation of Pandas data frame

```
1 import pandas as pd
2 # a1, a2, a3 = lists of attributes
3 df = pd.DataFrame.from_items([('label1', a1), ('
    label2', a2), ('label3', a3)])
```

Listing C.2: Union of df1 and df2 in a new data frame new_df

```
1 # df1, df2 = DataFrame
2 new_df = pd.concat([df1, df2])
```

Listing C.3: Group by attributes 'a1' and 'a2'. After that the mean, the sum and the median is calculated on the other attributes. The method reset_index() returns a data frame with the original attributes before the groupby was applied.

```
1 grouped_df = df .groupby(['a1', 'a2']).mean().
    reset_index()
2 grouped_df2 = df.groupby(['a1', 'a2']).sum().
    reset_index()
3 grouped_df2 = df.groupby(['a1', 'a2']).median().
    reset_index()
```

Listing C.4: Count how many occurrences for the attribute 'a1' and save this number in a new attribute called 'size'.

```
df = df.groupby('a1').size().to_frame('size').
    reset_index()
```

Listing C.5: Function for data manipulation. It creates a new column ('date'), from another ('B_ep') containing the respective B_{epoch} transformed in date time value with days as granularity.

```
def epoch_date_dd(df):
    # get a data frame with a column of epoch 'B_ep',
        returns
    # another column with the date yyyy-mm-dd so it
    # orders the date by day
    # :param df:  dataframe in input
    # :return:    new dataframe containing the 'date'
        attribute
    df['date'] = df['B_ep'].apply(epoch_datetime)
    df['date'] = df['date'].apply(revert_date_time)
    df['date'] = df['date'].str.slice(start=0, stop
        =10)
    return df
```

Listing C.6: Data retrieval using RESTful APIs provided from `blockchain.info`.

```
import urllib2
import json
global block_hash_url
block_hash_url = "https://blockchain.info/rawblock/"

def get_json_request(url):
    # Read the url and get json data.
    # :param url: str, site where to fetch information
    # :return: str, data requested in json format
    json_req = urllib2.urlopen(url).read()
    request = json.loads(json_req)
    return request

# get a block given a hash
block = get_json_request(block_hash_url + hash)

# get the previous block through block attribute '
    prev_block'
hash = block['prev_block']
```

Listing C.7: Block object structure obtained using Bitcoin's APIs

```
class Block:
  def __init__(self, b):
    self.hash = b['hash']
    self.version = b['ver']
    self.previous_block = b['prev_block']
    self.merkle_root = b['mrkl_root']
    self.time = b['time']
    self.bits = b['bits']
    self.fee = b['fee']
    self.nonce = b['nonce']
    self.n_tx = b['n_tx']
    self.size = b['size']
    self.block_index = b['block_index']
    self.main_chain = b['main_chain']
    self.height = b['height']
    self.received_time = b.get('received_time', b['
       time'])
    self.relayed_by = b.get('relayed_by')
    self.transactions = [Transaction(t) for t in b['
       tx']]
    for tx in self.transactions:
        tx.block_height = self.height
```

Listing C.8: Transaction object structure obtained using Bitcoin's APIs.

```
class Transaction:
  def __init__(self, t):
    self.double_spend = t.get('double_spend', False)
    self.block_height = t.get('block_height')
    self.time = t['time']
    self.relayed_by = t['relayed_by']
    self.hash = t['hash']
    self.tx_index = t['tx_index']
    self.version = t['ver']
    self.size = t['size']
    self.inputs = [Input(i) for i in t['inputs']]
    self.outputs = [Output(o) for o in t['out']]

    if self.block_height is None:
       self.block_height = -1
```

Listing C.9: Portion retrieval having a jump $J = 10$ and a number of blocks retrieved per time $b = 10$.

```
global latest_block_url
latest_block_url = "https://blockchain.info/
   latestblock"

jump = 10
b = 10
if(os.path.isfile(name)):
  # retrieve data frame from name.csv file
  df = pd.DataFrame.from_csv(name, sep='\t')
  hash_list = df['B_h'].values
  # get the last height
  height_list = df['B_he'].values
  last_block = height_list[-1]
  # subtract the jump
  last_block = int(last_block) - jump
  # get the block where to start the new fetching
  b_array = get_json_request("https://blockchain.
     info/block-height/" + str(last_block) + "?
     format=json")
  blocks = b_array['blocks']
  b = blocks[0]
  #  lget the block hash that has to be fetched
     first
  block_hash = b['hash']
  # call the method for fetching the blockchain
  get_blockchain(b, block_hash)
else:
  # file does not exist
  # retrieve the last block hash
  latest_block = get_json_request(latest_block_url)
  block_hash = latest_block['hash']
  get_blockchain(b, block_hash)
```

Listing C.10: Polynomial interpolation on miner's profit, $\langle \Pi \rangle$, and creation time, $\mathcal{T}$, using Numpy libraries.

```
x = df['B_T'].values
y = df['profit'].values
new_x, new_y, f = polynomial_interpolation(x, y,
    degree=2)

def polynomial_interpolation(x, y, degree=2):
  # given two lists of data it generates two new
     lists containing the y values interpolated
  # :param x : x values of the data to interpolate
  # :param y : y values of the data to interpolate
  # :param degree : polynomial degree
  # :return : x and y interpolated values. f is the
     polynomial.
  # order lists
  together = zip(x, y)
  sorted_together = sorted(together)
  x_vals = [el[0] for el in sorted_together]
  y_vals = [el[1] for el in sorted_together]
  # calculate polynomial
  z = np.polyfit(x_vals, y_vals, degree)
  f = np.poly1d(z)
  x_new = np.linspace(x_vals[0], x_vals[-1], len(
     x_vals))
  y_new = f(x_new)
  return x_new, y_new, f
```

Listing C.11: MAE accuracy calculation on miner's profit $\langle \Pi \rangle$ calculated using scikit-learn libraries in Python.

```
from sklearn.metrics import mean_absolute_error
new_x, new_y, f = polynomial_interpolation(x, y,
    degree=2)
predicted = []
samples = df['B_T'].values
real = df['profit'].values
for float(s) in samples:
  predicted.append = f(s)
mae = mean_absolute_error(real, predicted)
```

D

Profiling

Performance of BAS using a line profiler for Python [26]. The profiling was evaluated running BAS on a small dataset *D* with a size of 46 MB. We ran the profile on the *main.py* class, to have an overall of the whole BAS execution, then on the transaction retrieval and writing, and finally, on the plotting of data. Note that, for convenience, not all the lines are listed.

Timer unit: 1e-06 s
Total time: 103.534 s
File: main.py
Function: main at line 99

Line #	% Time	Line Contents
99		@profile
100		def main(argv):
111	0.0	opts, args = getopt.getopt(argv, "hipt:d:")
112	0.0	valid_args = False
114	0.0	for opt, arg in opts:
124	0.0	if opt == "-d":
126	85.5	jump = basretrieve.fetch_txs(int(arg))
128	13.8	basretrieve.read_txs_file()
129	0.0	if os.path.isfile(info_file):
130	0.0	os.remove(info_file)
131	0.7	print basmanipulation.get_dataframe_info(jump)

Timer unit: 1e-06 s
Total time: 85.4585 s
File: retrieval.py
Function: fetch_txs at line 87

```
Line #              % Time  Line Contents
==============================================================
87                          @profile
88                          def fetch_txs(jump):
93                  0.0       blocks_to_retrieve = 10
95                  0.0       if os.path.isfile(temporary_blocks):
96                  0.0           os.remove(temporary_blocks)
99                  0.0       if os.path.isfile(temporary_transactions):
100                 0.0           os.remove(temporary_transactions)
117                 0.8           df = pd.DataFrame.from_csv(dataframe, sep='\t')
118                 0.0           height_list = df['B_he'].values
122                11.4            b_array = get_json_request("https://blockchain.info/block-height/" +
str(last_block) + "?format=json")
123                 0.0           blocks = b_array['blocks']
125                 0.0           block_hash = b['hash']
126                87.7            get_blockchain(blocks_to_retrieve, block_hash)
```

```
Timer unit: 1e-06 s
Total time: 13.5002 s
File: retrieval.py
Function: read_txs_file at line 286

Line #              % Time  Line Contents
==============================================================
286                                           @profile
287                                           def read_txs_file():
301                 0.0       if os.path.isfile(temporary_transactions):
302                 0.0           with io.FileIO(temporary_transactions, "r") as file:
303                 0.0               file.seek(0)
304                 0.1               txs = file.read() # operations on txs are the most expensive ones
306                 0.1           list_txs = txs.split("\n")
307                 0.0           list_txs.pop()
317                 0.0           for el in list_txs:
318                 0.0               epoch_list.append(list_txs[i + 1])
319                 0.0               list_txs.remove(list_txs[i + 1])
327                 0.0           for t in list_txs:
328                 75.5                list_txs[i] = ast.literal_eval(t) # parse json transactions
335                 0.0           for i in range(len(epoch_list)):
342                                                   temp_input, temp_output, temp_fee_list, temp_size_list,
temp_approval_time_list, temp_hash_tx = \
343                 4.2                   calculate_transactions_fee(list_txs[i], int(epoch_list[i]))
352                 0.0           f_percentile = []
353                 0.1           for f_in, f_ou in zip(input, output):
354                 0.1               if float(f_in)!= 0:
355                 0.2                   percentile = 100 - (float(f_ou * 100) / float(f_in))
356                 0.1                 else:
358                 0.1               f_percentile.append(percentile)
389                 0.1           for tx in input:
390                 0.1               if i < indexes_list[counter]:
391                 0.1                   b_s.append(block_size[counter])
392                 0.1                   b_ct.append(block_creation_time[counter])
393                 0.1                   b_h.append(block_height[counter])
394                 0.1                   b_ep.append(block_epoch[counter])
395                 0.1                   b_t.append(block_txs[counter])
396                 0.1                   b_hash.append(block_hash[counter])
397                 0.1                   b_rel.append(block_relayedby[counter])
398                 0.1                   i += 1
410                 0.0           if os.path.isfile(dataframe):
413                 6.3               old_df = pd.DataFrame.from_csv(dataframe, sep='\t') # get data frame
416                 0.0               new_df = pd.DataFrame.from_items(
417                 0.0                   [('t_ha', hash_tx), ('t_in', input), ('t_ou', output), ('t_f', fee_list), ('t_q',
size_list),
418                 0.0                    ('t_%', f_percentile), ('t_l', approval_time_list),
419                 0.0                    ('Q', b_s), ('B_T', b_ct), ('B_he', b_h), ('B_ep', b_ep), ('B_t', b_t),
420                 0.1                    ('B_h', b_hash), ('B_mi', b_rel)])   423         1        46862  46862.0
0.3               new_df = pd.concat([old_df, new_df])
433                 8.9           new_df.to_csv(dataframe, sep='\t')
```

Timer unit: 1e-06 s
Total time: 328.466 s
File: plotting.py
Function: plot at line 96

Line #	% Time	Line Contents
96		@profile
97		def plot():
98	0.2	df = basmanipulation.get_dataframe()
101	1.3	plot_reward_fee(df, axes)
105	1.3	plot_profit_multiple_miners(df, axes)
109	3.3	plot_profit_creation_time(df, axes)
113	6.2	plot_total_btc(df, axes)
117	7.4	plot_fee_input_miners(df, axes)
121	1.5	plot_fee_latency(df, axes)
133	8.3	plot_fee_latency_years(df, axes)
137	12.8	plot_blocksize_latency(df, axes)
141	24.6	plot_throughput(df, axes)
145	1.0	plot_creation_time_miners(df, axes)
149	11.5	plot_block_size(df, axes)
157	10.7	plot_trendy_miners(df, axes)
161	9.3	plot_number_of_miners(df, axes)

/E

Usage

Trading Network Performance for Cash in the Bitcoin Blockchain - University of Tromsø

Longitudinal study on the Bitcoin blockchain from 2013 to 2017. In our thesis we mainly focus on three major problems on the Bitcoin blockchain: 1-Scalability. 2-Performance. 3-Fees and Tolls. We make our assumptions and test the results by analyzing the real blockchain with a data analytics system created for that purpose, BAS (Blockchain Analytics System), then we propose approaches that might be followed in order to get more performance in the Bitcoin network by optimizing the amount of fee paid from users. We also consider miner's revenue and give advide about the right creation time in order to optimize miner's profit, according to the mining hardware he is usign. We finally discuss whether is good to increase or not the block size to increase system performance.

Getting Started

```
\thesis: contains the thesis in .pdf format
\BAS: Blockchain Analytics System folder
\BAS\dataframe: data frame D generated for the analysis
\BAS\info: contains the info.txt file with information about data retrieved
\BAS\plot: plots generated with data retrieved
\BAS\src: source code containing main.py, data_manipulation.py, plotting.py and
retrieval.py
```

Prerequisites

Some libraries used for the computations might have to be installed, such as numpy, pandas or matplotlib.

```
pip install numpy
pip install matplotlib
pip install pandas
```

Usage

Usage of the blockchain alaytics system:

```
observ.py -d number
    -h | --help : usage
    -i          : gives info of the blockchain stored in /info
    -p          : plot data
    -t number   : get the amount of unconfirmed transactions for <number> minutes
    -d j        : retrieve information about transactions and save them in a Panda
DataSet, having a
                jump of j blocks with a default number of blocks retrieved b = 10
```

Example of use: use -d command for retrieval and set an initial jump J = 10

```
python main.py -d 10
```

Note: this jump will remain of 10 even if in the later analysis the variable is changed

once D is created data can be plotted

```
python main.py -p
```

Note: to have a nice plotting is suggested to have downloaded at least few months of activity in the blockchain

```
reward_fee.png                  : (date, BTC) plot the revenue from the block
reward R compared to the fee from users M
profit_multiple_miners.png      : (creation time, profit) plot the profit using
AntMinerS9 having 1, 50, 100, 500 miners in the mining pool.
profit_creation_time.png        : (creation time, profit) plot the revenue, costs
and profit for miners according the creation time
total_btc.png                   : (date, BTC) total bitcoin in circulation
fee_input_miners.png            : (miners, fee%) comparison between the percentage
of fee paid by the 20 biggest mining pools
fee_latency.png                 : (fee, latency) plot the transaction fee in
relation with the fee latency
txs_fee_distribution.png        : (date, %) plot the transaction fee distribution,
divided in category
txs_feedensity_distribution.png : (date, %) plot the transaction fee density
distribution, divided in category
fee_latency_years.png           : (fee, latency) plot the relation between the
transaction fee and the latency, distributed during years
blocksize_latency.png           : (block size, latency) plot the block size Q in
relation with the transaction latency
throughput.png                  : (date, throughput) plot throughput during time
creation_time_miners.png        : (creation time, blocks mined) bar plot of
occasional miners and mining pools about the creation time
block_size.png                  : (date, block size) plot the block size during
time
top_miners_monthly.png          : (date, blocks) plot the occasional miners and the
mining pools every months according to how many blocks they mine
trendy_miners.png               : (date, transactions) plot the transactions
approved by the 15 major miners during the years
number_of_miners.png            : (date, miners) plot the number of active miners
in the network
```

Built With

- [Python] :v2.7.12
- [PyCharm] :v2017.1.4

Enrico Tedeschi @ UiT - University of Tromsø (ete011)