Utrecht University
Faculty of Science
Department of Mathematics

Bitcoin mining as cooperative network

A pool of money

Bachelor Thesis

Supervisor:
Dr. M. Ruijgrok

Wicorel van der Pol
Student number 4174801

June 15, 2017

Abstract

Nowadays, cryptocurrencies are thriving. Due to the enormous growth of the Bitcoin Network, it was getting harder for miners to obtain Bitcoins and maintain a steady income. Therefore, mining pools were founded. Multiple miners combined their computational power and divided the rewards proportionally or by pay-per-share.

In this thesis we will focus on the proportional distribution according to the computational power of each miner and analyse if there exists a way to optimally divide the profits among the miners. We will mold the Bitcoin network to a model of cooperative games with coalitional structures.

The examination of the distribution within the pools led to a surprising conclusion: under certain conditions the $\mathbb{D}_{MS}$-CS-core will always be empty. When the partition function is constant-sum, monotonic and nonlinear with respect to the computational power, there exists no imputation which will satisfy each miner. In other words: there will always be an incentive for at least one miner to switch pools. It turns out that the delay of communication within a pool is crucial in this case. When the delay is below certain boundaries, the three beforementioned conditions will always apply, meaning the $\mathbb{D}_{MS}$-CS-core will always be empty. Thus, under certain conditions the network will never stabilize.

Contents

1 The Bitcoin Network **1**
1.1 Public ledger 2
1.2 Block Chain 2
1.3 Conflicting histories 2
1.4 Attacks 3
1.5 Pools 3
1.6 (Dis)advantages 5

2 Miner Coalitional Game **7**
2.1 Notation 7
2.2 The rate of block addition 8
2.3 Pools share 8
2.4 CS-core 11
2.5 $\mathbb{D}_{MS}$-CS-core 12

3 Delay: the key parameter **14**
3.1 Boundary d_1 14
3.2 Boundary d_2 15

A Proof-of-work **18**

B Notation overview **19**

References **I**

Credits for the illustration on the front page go to Manouk Locher.

Introduction

Cryptocurrencies are on the march: since the appearance of Blackcoin, Dogecoin, Litecoin and many others an interesting new digital market has arisen. In 2009, the first cryptocurrency was brought to life: the Bitcoin Network. This innovative network uses bitcoins, which only consists of one's and zero's. The virtual money is transferred in a similar way information across the internet is transferred: from one individual to the other (also referred to as peer-to-peer) and hence has no or small transaction fees. The second innovative part about Bitcoin is that the network is decentralized; no central authority controls the flow of bitcoins. The Bitcoin Network has miners who use their computational power to protect the system agains malicious users. They are rewarded with freshly produced bitcoins. This way, no central authority is needed to decide when to release more bitcoins into the market.

However, an unexpected growth of this network brought many miners to despair. It was getting harder, even impossible, to get a steady income of bitcoins by mining. This was fixed by working together in pools and splitting the profits. The rewards became lower, but more frequent. The manner in which the profits are divided, differ greatly between pools, which means that some miners might feel the need to switch strategy. Due to the establishment of pools, the structure of the network has become more complex. This raised questions as whether an ideal strategy to divide the profits within a pool exists and which factors do influence the decision of the miners to switch between pools or even work solo. Therefore, this thesis will focus on certain tools from cooperative game theory to study the behaviour of the miners.

Assuming that every miner wants a maximal payoff, we will examine when a miner is inclined to switch pools. During this process, multiple factors of the network will be considered. Eventually, it all comes down to the question whether there exists a cooperative miner game, with coalition structures, that doesn't have an empty core. As it will turn out, when the communication between miners is efficient enough, there exists no distribution of the shares that will keep every miner satisfied. Therefore, the network of pools will always be in motion.

1 The Bitcoin Network

Before we head on to the game theory of the Bitcoin Network, it will be quite useful to go through the basics first. Bitcoin is a digital cryptocurrency created by Satoshi Nakamoto. Creating a Bitcoin address (equivalent to a banking account) is free. Bitcoins can be bought with euros, dollars or other currency. Among other options can this be done at currency exchanges online, after which one can use it like any other currency; purchase anything you like in the form of transactions. The only difference is that bitcoins consist solely out of one's and zero's.

Another unique aspect of the currency is the absence of a global authority that regulates the distribution of the money; the Bitcoin Network is completely decentralized. There is no central authority that subordinates a monetary policy. The total number of bitcoins that eventually will circulate in the market has already been set by Nakamoto: never will there be more than 21 million bitcoins. [[7] p.105]

Considering the fact that the virtual, decentralized currency was a first and that is was not supported by the government, one might not be surprised that many people in Wall Street were sceptical about its endurance. In the beginning there were not many places where bitcoins were accepted as legal payment and many bitcoin exchanges failed. [[7] p.105] All in all, it had a rough start. However, what seemed to start as a quirky experiment suddenly flourished into something big:

"In the late spring of 2011, the price of [a] bitcoin had reached parity with the U.S. dollar, and by July, one bitcoin was worth $31." [[3] p.2]

This process has not deteriorated: on april the fifth 2017 one bitcoin was approximately worth $1118,- and on july the eleventh 2017 about $2902,- dollar. More and more people became aware of the profitable currency and started investing in it. Meanwhile, well-known companies like Subway, Steam, Wikipedia and Tesla started accepting bitcoins, which aided the network's growth even more. [1]

1.1 Public ledger

Digital money is faster to transmit than regular money. Still, every digital currency has one huge disadvantage: the double spending problem. Due to the absence of physical tokens (e.g. watermarks on papermoney), virtual money is quite easy to replicate. One can keep a copy of the string of bits, that represents your payment, and use this copy to pay for something else. This is referred to as the double spending problem.

A solution to this problem is to list all holdings of each Bitcoin address in a public ledger. A transaction simply changes the records on this ledger. However, this inevitably leads to the neccessity of a record keeper. This third party has much influence: it can demand high fees and block or run transactions without consent. As stated before, the Bitcoin Network strives to keep it decentralized. Therefore, the third party has been replaced by miners. The network thus consists of two kinds of agents: the clients (bitcoin owners who make bitcoin transactions) and the miners. Together, all the miners authorize the bitcoin transfers. They check for double spending and other inconsistencies in the transactions using the information of the ledger. Meanwhile they compete to authorize a share of the transactions, so they are rewarded with a small amount of 'new' bitcoins. Usually, a central authority decides when to print more money, but since the Bitcoin Network is decentralized, more bitcoins are added to the network by rewarding miners. Because all miners work together without a central authority, no unreasonably high fees can be demanded by one miner, nor can one of them decide to block transactions. Thus, all miners form a peer-to-peer network that keeps the Bitcoin Network secure.

1.2 Block Chain

Each miner keeps a copy of the ledger and is continuously updated with all transactions that take place. Consequently, these transactions are public and therefore the ledger is transparent. The workload is not distributed. Every miner has to do the same verification. Now, what if two miners obtain a different ledger, because of double spending? Their information somehow has to be synchronized. In order to show how this is done in the Bitcoin Network, we first have to obtain more information about the details of Bitcoin's main data structure: the block chain.

All verified transactions of the Bitcoin Network are stored in the block chain. As the name suggests it consists of blocks linked together. It starts with the first block that was created at the beginning of the Bitcoin network and it is therefore called the genesis block. Every other block has a parent block, which is a predecessor of either one or two other blocks. A block in the chain contains the following:

- batches of approved transactions,
- a (unique) cryptographic hash to indicate the preceding block (the cryptographic hash can be seen as an identifier for the Bitcoin addresses which requested the transfer),
- and a coinbase transaction (the reward for the lucky miner).

Each miner verifies the signatures of the Bitcoin addresses which requested the transfer of bitcoins. The miner then forwards it to peers in the network. A block is called valid when the included transactions are not in conflict with the current balances which are given by the preceding blocks. Together, all these blocks keep a record of all the transactions since the creation of Bitcoin. Therefore, we can see the chain as the virtual representation of the transparent ledger. More information about how miners exactly verify a block, is available in Appendix A.

1.3 Conflicting histories

Since blocks are created quite randomly, it often occurs that two miners extend the same parent block. When different miners extend this block at approximately the same time a fork is created in the block chain. The result is different ledgers spread throughout the network. However, the Bitcoin Network has two main rules that solve the problem of conflicting histories.

1. **The use of proof-of-work**
   To mine a block miners need to solve a computationally hard problem. The solution of this problem is

called the proof-of-work and is easy verifiable by every miner. Still, finding this solution is far from easy. A miner will simply have to let his computer try out many guesses until he finds the right one. When a proof-of-work is found, the miner adds it to the block and sends it to the other miners (by adding it to the block chain), so they can verify his solution. The miner who mined this block is rewarded with bitcoins.

Because there is no simpler way of finding the proof-of-work, it is hard to create a block. The difficulty of the computational problem is set such that every ten minutes a block will be created by the entire network. This means that the time interval to update the other miners is hopefully long enough to notify each miner, before one of them creates a second block for the same parent. The proof-of-work therefore prevents forks in the block chain.

2. **The longest chain concept**
When miners notice that different versions of the ledger exist, it means the chain has split and according to Bitcoin policy every miner must adopt the longest chain. This chain has the longest transaction history. The other version of the chain will be abandoned. Transactions that were included within this branch become invalid and are taken out of the ledger, by which the fork dissappears. The concept of the longest chain therefore resolves fork-situations.

1.4 Attacks

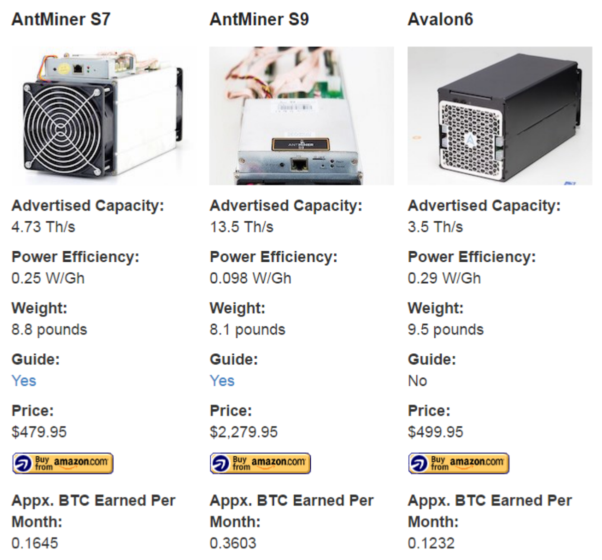
Unfortunately, like any other system, there are people who try to find it's weaknesses and exploit them. The fork-problem has been used by some to their advantage. Let's consider such a person (an attacker) who just bought something, but suddenly wants to reverse his transaction, which has already been approved. He creates a fork in the block chain by adding a block to some part in the chain before the block that contains his approved transaction. (Remember that the ledger is transparent, so an attacker can easily figure out where this block is.) If this branch becomes long enough, the longest chain concept ensures that the blocks of the other branch will be removed from the block chain, thereby erasing all the transactions in these blocks from the public ledger. This includes the transaction the attacker wanted to reverse and so he had his way. This is called a double spending attack.

However, we already learned that creating a block is very difficult. Being a succesful miner requires a lot of computational power. This means that if an attacker wants to create a fork, he needs to mine at a higher rate than the rest of the network. In other words, he must be incredibly well technologically equiped. Since the peer-to-peer network is far from a small community an attacker must posses at least 50% of the computational power of the entire network. Otherwise, his chances are nil. [[7] p.108] When a miner does obtain at least 50% of the network's computational power, he can launch a 50% attack. Because he creates blocks at a faster pace than the rest of the network, he can adjust the block chain anyway he likes by double spending attacks. The attacker can even stop all transactions by generating a chain full of empty blocks. The only thing he can't do is move funds he doesn't control, because transactions always have a cryptographical signature of the sender. Still, it is a worrying concept.

These attacks stress the need for a well-connected peer-to-peer network. If enough honest miners join the network, it will get less likely that an attacker has enough computational power to force a fork in the chain. In other words: the more honest miners, the more resilient the network is to double spending attacks. Miners keep the network safe.

1.5 Pools

Due to the vast growth of the network it was getting harder and harder for miners to mine bitcoins. In the beginning people could mine using the processors of their computers. Then more and more people discovered that graphics cards (which are used for gaming) were more efficient. The downside is that they use a lot of electricity and therefore generate a lot of heat. Eventually, special chips were developed for mining bitcoins: ASIC (Application-Specific Integrated Circuit chips). Figure 1 shows the best Bitcoin mining hardware options based on the current (02/05/17) price per hash and electrical efficiency.



AntMiner S7	AntMiner S9	Avalon6
Advertised Capacity: 4.73 Th/s	**Advertised Capacity:** 13.5 Th/s	**Advertised Capacity:** 3.5 Th/s
Power Efficiency: 0.25 W/Gh	**Power Efficiency:** 0.098 W/Gh	**Power Efficiency:** 0.29 W/Gh
Weight: 8.8 pounds	**Weight:** 8.1 pounds	**Weight:** 9.5 pounds
Guide: Yes	**Guide:** Yes	**Guide:** No
Price: $479.95	**Price:** $2,279.95	**Price:** $499.95
Buy from amazon.com	Buy from amazon.com	Buy from amazon.com
Appx. BTC Earned Per Month: 0.1645	**Appx. BTC Earned Per Month:** 0.3603	**Appx. BTC Earned Per Month:** 0.1232

Figure 1: The best Bitcoin mining devices. Source: www.bitcoinmining.com

Still, it could take years for slower miners to mine a block. Those who wanted a steady income, needed a new approach. Therefore, the idea arose to form groups of miners who can combine their computational power and share the profit, hence mining pools were set up. When a member mines a block the profit is divided among the entire pool, such that all the miners have a lower but more steady income. The reward for mining a block is included in the block as the coinbase transaction and consists of a small fee that is paid for adding transactions to the block and new bitcoins which are added to the market. Remember that there will never be more than 21 million bitcoins, so the number of bitcoins rewarded to the miners decreases with every mined block. The first blocks added to the Genesis Block were rewarded with 50 bitcoins. Every 210,000 blocks this number is halved. This means that the currency is deflationary: bitcoins may be lost and never replaced because of the fixed number that is available. [[7] p.110]

The structure of each pool is as follows. Each pool has a pool manager and miners. The miners only communicate via their pool manager (also called a pool operator). The pool manager is able to send block headers (a hash to indicate the previous block) to his miners.

First, the pool manager will run a Bitcoin node on behalf of the other miners in his pool, collecting transactions and assembling them into a block. He then adds his own address to this block and sends it to the miners in his pool.

In return the miners send back nonce fields which are suitable according to the information in the received block headers. These nonce fields are basically attempts to find the beforementioned proof-of-work and they are used to show that every miner is working on this block. Each nonce that is sent, is called a share. When the right nonce is found the pool manager will receive the rewards and divide them among the miners in his pools. [[5] p.125,126]

Their are multiple ways to divide the profits in a pool. The two most basic models are the proportional and pay-per-share model (see table 1). In the first model the payoff is distributed according to one's relative computational power in the pool. This proportional model is low risk for the pool managers, because they only have to pay out when a valid block is found and they have received bitcoins. In a pay-per-share model the pool manager has to pay the miners for the shares they send, even when they are still working on a block. Of course the pool manager will also keep a part of the profit, but in this thesis we will focus on the share of the miners.

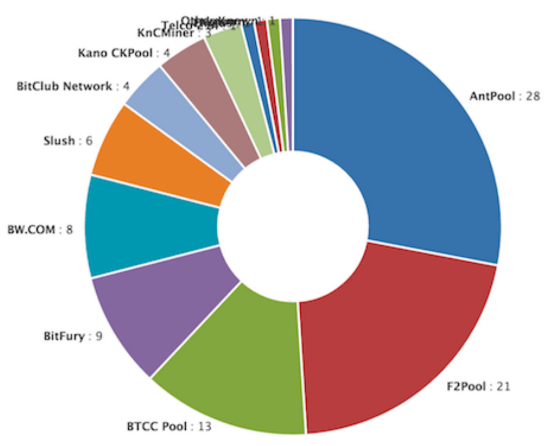
Pool managers can communicate with one another, but instead of a header, they have to send an entire block to each other. Miners within their pool only need the block headers. These forms of communication cause delays in the network, which will later on turn out to be extremely important for our main question.

Method	Approach
PPS:	The Pay-per-Share (PPS) approach offers an instant, guaranteed payout for each share that is solved by a miner. Miners are paid out from the pools existing balance and can withdraw their payout immediately. This model allows for the least possible variance in payment for miners while also transferring much of the risk to the pool's operator.
PROP:	The Proportional approach offers a proportional distribution of the reward when a block is found amongst all workers, based off of the number of shares they have each found.

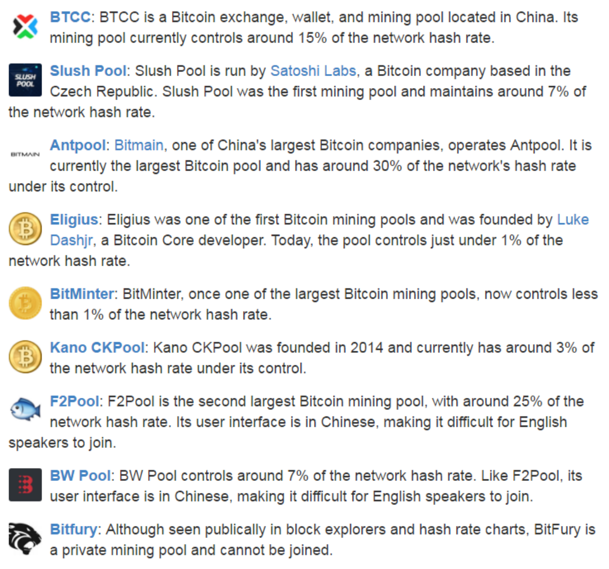
Table 1: Distribution methods

As you can see in figure 2 many different pools of varying sizes exist. Each pool has his own model to divide the rewards. Returning to the problem of the attackers, it is recommended to join the smaller pools. When a pool approaches 50% of the network's block creation rate, it is a threat to the system. Even when the members of the pool are good-natured, it is still possible that a hacker manages to compromise the pool's servers, risking a 50% attack. Therefore, many miners chose to join the smaller pools.

Nevertheless, large mining pools are not completely undesirable. In March of 2013 a bug occured in the code, which resulted in a long-lasting fork in the block chain. The large mining pools were askes to downgrade to an older version of the protocol and due to this the fork dissolved. [[7] p.112]



(a) Pie chart of current recommended pools



BTCC: BTCC is a Bitcoin exchange, wallet, and mining pool located in China. Its mining pool currently controls around 15% of the network hash rate.

Slush Pool: Slush Pool is run by Satoshi Labs, a Bitcoin company based in the Czech Republic. Slush Pool was the first mining pool and maintains around 7% of the network hash rate.

Antpool: Bitmain, one of China's largest Bitcoin companies, operates Antpool. It is currently the largest Bitcoin pool and has around 30% of the network's hash rate under its control.

Eligius: Eligius was one of the first Bitcoin mining pools and was founded by Luke Dashjr, a Bitcoin Core developer. Today, the pool controls just under 1% of the network hash rate.

BitMinter: BitMinter, once one of the largest Bitcoin mining pools, now controls less than 1% of the network hash rate.

Kano CKPool: Kano CKPool was founded in 2014 and currently has around 3% of the network hash rate under its control.

F2Pool: F2Pool is the second largest Bitcoin mining pool, with around 25% of the network hash rate. Its user interface is in Chinese, making it difficult for English speakers to join.

BW Pool: BW Pool controls around 7% of the network hash rate. Like F2Pool, its user interface is in Chinese, making it difficult for English speakers to join.

Bitfury: Although seen publically in block explorers and hash rate charts, BitFury is a private mining pool and cannot be joined.

(b) Origin of mentioned pools

Figure 2: Source: https://www.bitcoinmining.com/bitcoin-mining-pools/

1.6 (Dis)advantages

All in all, the Bitcoin Network is a system that keeps evolving. Improvements certainly are desirable. One huge disadvantage is that every miner keeps a copy of the ledger, which means a lot of useless data is being stored. Some have suggested that this wasteful replication can be reduced to some extent. For example, everyday users of the currency don't need the entire history of transactions, because some are too old to be of use. When a part of the chain is removed, they have to store less data and thus have lower storage costs. The economic influences are also getting stronger: large miners are profiting from their increasing bank accounts (e.g. they can produce their own hardware) and thereby eliminating smaller miners from the market.

Another disadvantage is the loss of privacy. All transactions are made public in the ledger, so anyone can verify them. This transparancy is useful for organizations who want to show their clients how they are using their money. Still, the privacy of individuals is at risk. Bitcoin users can create new pseudonyms for each transaction, but full anonymity will never be reached.

Nevertheless, there are many advantages: spreading money like information across the internet is relatively fast, anyone can join the network (as a miner and as a client) and every transaction is verifiable because of the ledger. More importantly, there is no third party or central government which can demand high fees. Decentralization is the essence of the Bitcoin Network. Its ongoing goals are to maintain this decentralization while protocol changes and other developments take place. We have seen how the miners make the network more resilient against double spending attacks and therefore keep it secure. When the network began to grow and profits were obtained less often and more randomly, the miners chose to evolve with the system by creating pools. Maybe this one of the reasons why the Wall Street critics were shown wrong about Bitcoin: it is not a perfect system, but every member (miners, developers, regulators and adopters) can reveal its shortcomings and is involved in its innovation. Step-by-step Bitcoin is proving its worth.

2 Miner Coalitional Game

We will now analyse the structure of the bitcoin miner network as a cooperative game. The pools will be called coalitions. Our miners can join a pool or work solo. They can only be part of one pool, so there is no overlap between the coalitions. A few other assumptions are made:

- the reward for a mined block is fixed,
- the miners follow the protocol as described in Chapter 1,
- the delay in communication between miners is fixed,
- and every miner i mines blocks according to a Poisson proces with parameter λp_i.

2.1 Notation

We start off with a general introduction to the notations[1] commonly used in Game Theory. For now, we leave out the coalitional structures. There always is a grand coalition, $I = \{1, 2, \ldots, n\}$, containing all agents. Every subset $C \subseteq I$, $|C| > 1$ is a coalition. We define the characteristic function $v : 2^I \to \mathbb{R}$. For every coalition this function determines the payment that can be achieved because of the collaboration. Furthermore, an imputation is a vector $x \in \mathbb{R}^{|I|}$ that divides all the gains of the grand coalition among the miners. It has the following properties:

1. the payoff of a player i is $x_i \geq 0$;
2. the payoff of a coalition $x(C) = \sum_{i \in C} x_i$;
3. $x(I) = \sum_{i \in I} x_i = v(I)$.

Furthermore, we define a weight vector $w \in \mathbb{R}^{|I|}$, with $\forall i \in I : w_i \geq 0$ and $\sum_{i \in I} w_i = 1$. Again, the definition for the coalitions is similar: $w(C) = \sum_{i \in C} w_i$.

Now, we will consider a game with coalitional structures. The grand coalition will be divided by some partition $S = \{C_1, C_2, \ldots, C_m\}$, with m coalitions and $\cup_{i=1}^{m} C_i = I$. Since no overlap is assumed, we know that $C_i \cap C_j = \emptyset$ for $i \neq j$. The utility of a coalition under the partition S of other agents is given by the partition function $v(S, C_i) = v_s(C_i)$. The set of all possible partitions of I is denoted by $CS(I)$. This set contains all the possible coalitional structures over I.

The delay between two coalitions C_i, C_j $(i \neq j)$ is denoted by D and the delay within a coalition C_i is denoted by d. So in case a miner wants to announce that he or she has succesfully mined a block, it will take d seconds for the poolmanager to be notified and an extra D seconds before the other poolmanagers receive the information. After $2d + D$ seconds all the other miners will be aware of this progress.

We define $P = \{p_1, p_2, \ldots, p_n\}$ as the distribution of the relative computational power among the miners. This will be used to determine the share of a miner in the pool. The computational power of agent i is denoted by $p_i \in [0, 1]$ and $\sum_{i \in I} p_i = 1$.

Finally, we will use λ to indicate the number of blocks which is expected to be mined by the network per second. We can now define a Miner Network as

$$\Gamma = \langle M, S, P, D, d, \lambda \rangle,$$

where M is the set of miners instead of I.

[1]All the notations are collected in appendix B.

2.2 The rate of block addition

We define β as the desired rate of block addition to the longest chain per second. The actual rate of block addition is defined by $\beta(\Gamma)$, where Γ is a Miner Network. When $\beta(\Gamma)$ increases, the network becomes more efficient, because more transactions are verified per second. A Miner Coalitional Game with Coalition Structures is defined as

$$\mathbf{C} = \langle M, P, D, d, \beta \rangle.$$

In the network a threshold (t) is often adjusted such that every 10 minutes a block is mined. This concept is called retargeting and makes sure that $\beta(\Gamma)$ stays close to β.

For example, a lower t means a miner needs more attempts to obtain the nonce that produces the right hash (p_t decreases). Therefore, more computational power is needed and thus the rate of longest chain growth ($\beta(\Gamma)$) decreases and the network can process less transactions in the same time. The adjustment of the threshold t results in a $\beta(\Gamma) \approx \beta$. Therefore, we have set $\beta(\Gamma) = \beta$ in our Miner Coalitional Game. Now let's explore the boundaries of $\beta(\Gamma)$.

Proposition 2.2.1. *Let $\Gamma = \langle M, S, P, D, d, \lambda \rangle$ be a Miner Network with $|M| > 1$ miners and $D, d > 0$. For every solo miner i, it holds that $\lambda \geq \beta(\Gamma) \geq p_i\lambda$.*

Proof. We will split the proof in two cases:

- $\lambda \geq \beta(\Gamma)$

  We know that $\beta(\Gamma)$ is the rate of block addition to the longest chain. In the best case scenario each block that is mined per second (λ) will have a hash below the threshold. So each mined block will be added to the longest chain. This means $\beta(\Gamma) = \lambda$ in the best case. In reality, not every mined block will be added to the longest chain, so therefore $\lambda \geq \beta(\Gamma)$.

- $\beta(\Gamma) \geq p_i\lambda$

  Remember that every miner i mines blocks according to a Poisson process with parameter $p_i\lambda$. Hence, the rate of block addition cannot be smaller than the speed at which an arbitrarily miner i is mining. If every miner mines blocks just as fast as the others, then $\beta(\Gamma) = p_i\lambda$ for every $i \in M$.

□

2.3 Pools share

Joining a pool makes it easier to mine, but the payoff must be divided in a suitable way. In this model each miner strives to obtain as many bitcoins as possible. Therefore, we define $\gamma(\Gamma)_i$ as the chance that a miner i has succesfully mined a block in a Miner Network Γ. We will sometimes just write γ_i instead of $\gamma(\Gamma)_i$.

Every miner wants to increase his or her γ_i, because you only receive bitcoins for mining a block which becomes part of the longest chain. Thus, γ_i is the total share of a miner i in the network. The shares are divided proportionally, so $\forall i : \gamma_i \in [0, 1]$. Furthermore, the total share of a coalition is defined by $\gamma_C = \sum_{i \in C} \gamma_i$. An increase of one's share in the longest chain means an increase in one's payoff in bitcoins. We can therefore set γ_i as our payoff function: $\forall C \in S : v_s(C) = \gamma_C$.

Now, let's explore two extreme examples of a Miner Network and observe what this will do to $\beta(\Gamma)$. We start off with a network that consists of one large pool, containing all the miners in the network:

Proposition 2.3.1. *Let $\Gamma = \langle M, \{M\}, P, D, d, \lambda \rangle$ be a Miner Network, then $\beta(\Gamma) = \frac{\lambda}{1+2d\lambda}$.*

Proof. Since $S = \{M\}$, we have a Miner Network with one pool and no solo miners. With one pool there only exists a delay between the pool manager and the miners (d). Let's assume that a block X was mined and added to the longest chain. We want to know how long it takes to mine a second block which will be added to the longest chain in order to calculate $\beta(\Gamma)$.

When the pool manager becomes aware of block X, he will need d seconds to update the miners with a new target. Since λ is the expected number of blocks mined by the network per second, a new block Y will be created after approximately $\frac{1}{\lambda}$ seconds. It takes another d second for the miner i to notify the manager.

In the meantime $(2d+\lambda^{-1})$ seconds have passed since the creation of block X. Therefore, the rate of block addition to the longest chain ($\beta(\Gamma)$) is equal to $(2d+\lambda^{-1})^{-1} = \frac{1}{2d+\frac{1}{\lambda}} = \frac{\lambda}{2d\lambda+1}$. □

The other extreme example is a network with only two solo miners. From Sompolinsky and Zohar [6] we obtain

Theorem 2.3.2. *Let Γ be a Miner Network with two solo miners. Then*

$$\beta = \beta(\Gamma) = \frac{(\lambda p_1)^2 e^{2Dp_1\lambda} - (\lambda p_2)^2 e^{2Dp_2\lambda}}{\lambda p_1 e^{2Dp_1\lambda} - \lambda p_2 e^{2Dp_2\lambda}} \tag{1}$$

and when $p_1 = p_2 = \frac{1}{2}$ then it holds that $\beta(\Gamma) = \lambda\frac{2+D\lambda}{2+2D\lambda}$.

Proof. For the extensive complete proof of equation (1) we refer to [6] Appendix E. The special case of $p_1 = p_2 = \frac{1}{2}$ is easier to explain. Note that by substituting $p_1 = p_2 = \frac{1}{2}$ in equation (1), we would divide by zero. We need another approach.

First, we write $p_2 = 1 - p_1$. We substitute this into equation (1):

$$\begin{aligned}
\beta(p_1) &= \frac{(\lambda p_1)^2 e^{2Dp_1\lambda} - (\lambda(1-p_1))^2 e^{2D(1-p_1)\lambda}}{\lambda p_1 e^{2Dp_1\lambda} - \lambda(1-p_1)e^{2D(1-p_1)\lambda}} \\
&= \frac{\lambda^2 p_1^2 e^{2Dp_1\lambda} - \lambda^2(1-p_1)^2 e^{2D\lambda-2Dp_1\lambda}}{\lambda p_1 e^{2Dp_1\lambda} - \lambda(1-p_1)e^{2D\lambda-2Dp_1\lambda}} \\
&= \frac{\lambda p_1^2 e^{2Dp_1\lambda} - \lambda(1-p_1)^2 e^{2D\lambda-2Dp_1\lambda}}{p_1 e^{2Dp_1\lambda} - (1-p_1)e^{2D\lambda}e^{-2Dp_1\lambda}} \qquad (\lambda > 0) \\
&= \frac{\lambda p_1^2 - \lambda(1-p_1)^2 e^{2D\lambda(1-2p_1)}}{p_1 - (1-p_1)e^{2D\lambda(1-2p_1)}} \qquad \text{(Divide everything by } e^{2Dp_1\lambda} \neq 0) \\
&= \frac{\lambda\big(p_1^2 - (1-p_1)^2 e^{2D\lambda(1-2p_1)}\big)}{p_1 - (1-p_1)e^{2D\lambda(1-2p_1)}}.
\end{aligned}$$

Then we use l'Hôpital's rule by taking the derivatives of p_1 and calculating the limit of $p_1 \to \frac{1}{2}$.

$$\begin{aligned}
\text{[Numerator]} \quad \beta_1'(p_1) &= \lambda\big(2p_1 + 2(1-p_1)e^{2D\lambda(1-2p_1)} + (1-p_1)^2 4D\lambda e^{2D\lambda(1-2p_1)}\big) \\
\text{[Denominator]} \quad \beta_2'(p_1) &= 1 + e^{2D\lambda(1-2p_1)} + (1-p_1)4D\lambda e^{2D\lambda(1-2p_1)} \\
\beta(p_1) = \lim_{p_1\to\frac{1}{2}} \frac{\beta_1'(p_1)}{\beta_1'(p_1)} &= \lambda\Big(\frac{1+1+\frac{1}{4}\cdot 4D\lambda}{1+1+\frac{1}{2}\cdot 4D\lambda}\Big) \\
&= \lambda\Big(\frac{2+D\lambda}{2+2D\lambda}\Big).
\end{aligned}$$

□

In this Two Miner Network we can also provide a formula for the share of each player.

Theorem 2.3.3. *Let Γ be a Miner Network with two solo miners. Then*

$$\gamma(\Gamma)_1 = \frac{p_1^2 e^{\mu p_1} - p_1 p_2\big(\frac{2e^{\mu}-2}{e^{\mu p_1}+e^{\mu p_2}-2} - 1\big)}{p_1^2 e^{\mu p_1} - p_2^2 e^{\mu p_2}} \tag{2}$$

where $\mu = 2D\lambda$ and $\gamma(\Gamma)_2 = 1 - \gamma(\Gamma)_1$.

Proof. Again we refer to [6] Appendix E for the involved proof. □

In the case of $p_1 = p_2 = \frac{1}{2}$ we can again use L'Hôpital's rule to obtain $\gamma_1 = \gamma_2 = \frac{1}{2}$. This means that when the computational power of both solo miners is equal, they payoff will also be split equally. When for example $p_1 > p_2$ we know that $\mu p_1 > \mu p_2$. Therefore, $e^{\mu p_1} > e^{\mu p_2}$ and also $p_1^2 e^{\mu p_1} > p_2^2 e^{\mu p_2}$. Considering this we can see that γ_1 of equation (2) will increase and therefore $\gamma_2 = 1 - \gamma_1$ will decrease. This is in agreement with our proportional payoff system.

More importantly, we will see a remarkable result when the difference between p_1 and p_2 is relatively small. To obtain figure 3a we fixed $\lambda = 1$. Furthermore we set $p_1 = 0.55$, $p_2 = 0.45$ and then plotted γ_1 as a function of $D\lambda$. Despite the small difference in computational power the first solo miner has a significant higher payoff when the delay increases.

Note that the threshold of the bitcoin network is set such that every ten minutes a block will be mined, so in this case the expected number of blocks mined by the network per second would be $\frac{1}{600}$. In figure 3b we can see for the same values of p_1, p_2 the function of the share has stabilized to the computational power of miner 1. However, we couldn't plot $D = 0$, because this would involve dividing by zero. To calculate γ_i for $\mu = 2D\lambda = 0$ we use l'Hôpital again and obtain:

$$\begin{aligned}\lim_{\mu\to 0}\gamma(\Gamma)_1 &= \lim_{\mu\to 0}\frac{p_1^2e^{\mu p_1} - p_1p_2(\frac{2e^{\mu}-2}{e^{\mu p_1}+e^{\mu p_2}-2} - 1)}{p_1^2e^{\mu p_1} - p_2^2e^{\mu p_2}}\\ &= \frac{p_1^2 - p_1p_2(\lim_{\mu\to 0}(\frac{2e^{\mu}-2}{e^{\mu p_1}+e^{\mu p_2}-2} - 1))}{p_1^2 - p_2^2}\\ &= \frac{p_1^2 - p_1p_2(\lim_{\mu\to 0}(\frac{2e^{\mu}-2}{e^{\mu p_1}+e^{\mu p_2}-2} - 1))}{p_1^2 - p_2^2},\end{aligned}$$

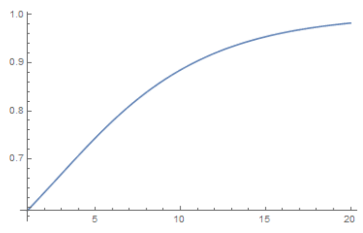
$$\textbf{L'Hôpital:}\quad \begin{aligned}\lim_{\mu\to 0}(\frac{2e^{\mu}-2}{e^{\mu p_1}+e^{\mu p_2}-2} - 1) &= \lim_{\mu\to 0}(\frac{2e^{\mu}}{p_1e^{\mu p_1}+p_2e^{\mu p_2}-2} - 1)\\ &= \frac{2}{p_1+p_2} - 1 \qquad (p_1+p_2=1)\\ &= 1.\end{aligned}$$

Substituting this back into the first equation gives us:

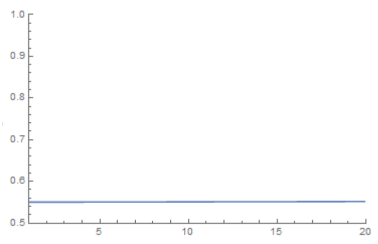
$$\begin{aligned}\lim_{\mu\to 0}\gamma(\Gamma)_1 &= \frac{p_1^2 - 2p_1p_2}{p_1^2 - p_2^2}\\ &= \frac{p_1(p_1-p_2)}{(p_1-p_2)(p_1+p_2)}\\ &= \frac{p_1(p_1-p_2)}{(p_1-p_2)} = p_1.\end{aligned}$$

Thus, when $D\lambda = 0$, the share of each miner becomes the proportion of computational power held by him or her ($\gamma_i = p_i$).

Corollary 2.3.4. *Let Γ be a Miner Network with two solo miners as defined in theorems 2.3.2 and 2.3.3. If $p_i > \frac{1}{2}$, then $\gamma_i \geq p_i$ and if $D\lambda > 0$ then $\gamma_i > p_i$.*



(a) $\lambda = 1$



(b) $\lambda = \frac{1}{600}$

Figure 3: The share γ_1 as a function of $D \in [1, 20]$ for $p_1 = 0.55, p_2 = 0.45$.

2.4 CS-core

We will now focus on Miner Networks with Coalitional Structures and investigate if there are imputations that divide the gains in a satisfying way for all the miners. Otherwise, the CS-core will be empty. In this section we won't consider γ_i as our payoff function yet. We first want to study the general case.

An imputation x is blocked when there exists a coalition $B \subseteq M$ such that $v(B) > \sum_{i \in B} x_i = x(B)$. This means that the miners are better off working as a group B, than when they stick to the situation of the grand coalition with the given imputation x. From here it follows that an imputation x belongs to the core if it is not blocked by any coalition:

$$\forall C \subset I \text{ we have } x(C) \geq v(C).$$

We don't just want to look at the core of one coalition. The CS-core is defined for the entire partition $S \in CS(M)$, so we have to consider every possible deviation from this partition. Only then can we make general conclusions about the Miner Network. The CS-core is defined by:

$$\{x \in \mathbb{R}^{|I|} \mid x_i \geq 0, \forall C \subset I : x(C) \geq v_{S_C}(C)\}, \tag{3}$$

where $S_C = \{C\} \cup \{\{D \setminus C\} \mid D \in S, \quad D \setminus C \neq \emptyset\}$. Note that: $x(C) = v_S(C)$, but not neccessarily: $x(C) = v_{S_C}(C)$. An imputation only defines the payoff for the current coalition structure. The partition S_C consists of a new coalition C and the same coalitions as S, but without the miners collected in C.

Example 2.4.1. Let $S = \{\{a, b\}, \{c, d\}, \{e\}\}$. Now let's say that miners a and e decide that they are better off working together. By creating their new coalition we get $S_C = \{\{a, e\}, \{b\}, \{c, d\}\}$. An imputation is in the CS-core of the partition S if the for every S_C we create, the payoff will be lower for the coalition C. △

Now we will see that under certain conditions this core will always be empty. We use the conditions as defined by Lewenberg et al. [4]. Assume that the payoff function v satisfies:

1. **Constant-sum**
   For every $S \in CS(I)$ we have $\sum_{C \in S} v_s(C) = c$.
   This means that the total payoff to be distributed among the grand coalition is always equal to a constant positive number c.

2. **Nonlinearity with respect to a weight vector** $w \in \mathbb{R}^+$
   A partition function v is nonlinear (with respect to a weight vector w) if:
   for every partition S with $\max_{C \in S} w(C) > \min_{C \in S} w(C)$, we have $v_s(C_i) > w(C_i) \cdot \sum_{C \in S} v_s(C)$
   for the maximum coalition $C_i \in \arg(\max_{C \in S} w(C))$.
   In other words: for every partition with different weights for the coalitions, the coalition with the biggest share (highest weight) has a payoff bigger than all the payoffs of the coalitions times its weight. This means its profit is higher than its share in the network.

Let's consider a partition $S = \{C_1, \dots, C_m\}$ of the Miner Network $\Gamma = \langle M, S, P, D, d, \lambda \rangle$ and apply these conditions to it. The proportional distribution of the computational power will be our weight vector. We are now ready to prove the emptiness of the CS-core of S.

Theorem 2.4.2 (Emptiness of CS-core)**.** *Let v be the partition function of a coalitional game with coalition structures (CS) of M. Let p be a weight vector satisfying $\forall i \in I : p_i < \frac{1}{2}$. If v is constant-sum (1) and nonlinear (2) for p, then the CS-core is empty for every coalition structure of M.*

Proof. Let v be a partition function as described in the theorem. Recall that the CS-core of S is:

$$\{x \in \mathbb{R}^{|M|} \mid x_i \geq 0, \forall C \subset I : x(C) \geq v_{S_C}(C)\},$$

with $S_C = \{C\} \cup \{\{D \setminus C\} \mid D \in S, \{D \setminus C\} \neq \{\emptyset\}\}$. The proof consists of taking a partition S of a coalition structure and then proving for every imputation x that there exists a $C \subset I$ such that $v_{S_C}(C) > x(C)$.

Take $S \in CS(M)$ and let x be an imputation associated with S. Without loss of generality we can assume that $M = \{1, \dots, n\}$ and $\sum_{A \in S} v_S(A) = 1, \forall S \in CS(M)$. The exact number of miners is not essential and for every partition the payoff function must be constant for $c > 0$, so we choose $c = 1$.

Now, let $\epsilon_i = x_i - p_i$ for every i and assume without loss of generality that $\epsilon_1 \leq \epsilon_2 \leq \cdots \leq \epsilon_n$. Note that

$$\begin{aligned}\sum_{i \in M} \epsilon_i &= (x_1 - p_1) + \cdots + (x_n - p_n) \\ &= x_1 + x_2 + \cdots + x_n - (p_1 + p_2 + \cdots + p_n) \\ &= \sum_{i \in M} x_i - \sum_{i \in M} p_i \\ &= 1 - 1 = 0.\end{aligned}$$

Let's now set $D = M \setminus \{n\}$. This coalition will disrupt the core. Since ϵ_n is the greatest ϵ we know that $\epsilon_n \geq 0$. This means $\sum_{i \in D} \epsilon_i = \sum_{i \in M} \epsilon_i - \epsilon_n \leq 0$ and so $x(D) \leq p(D)$. Likewise, $p_n < \frac{1}{2}$ and therefore $\sum_{i \in D} p_i > \frac{1}{2}$. Thus, our coalition D has the highest weight vector ($D \in \arg(\max_{C \in S} w(C))$ and because v is non-linear with respect to p:

$$\begin{aligned}v_{S_C}(D) &> p(D) \cdot \sum_{C \in S} v_s(C) \\ &> p(D) \cdot 1 \\ &= p(D) \geq x(D).\end{aligned}$$

This means that for every imputation x we can always construct a coalition D such that $v_{S_C}(D) > x(D)$. Therefore, the CS-core is empty. □

The condition of constant-sum is crucial, because this means we can assume $\sum_{C \in S} v_{S_C}(C)$ is constant. The condition of nonlinearity is necessary in order to prove $v_{S_C}(C) > p(C)$, which will lead to the desired result: $v_{S_C}(C) > x(C)$.

What does the emptiness of this core mean? We considered a grand coalition of miners M, who were divided into multiple coalitions $\{C_1, \ldots, C_m\}$. The total amount of bitcoins which are to be distributed is constant (condition 1). Furthermore, the partition function v is nonlinear with respect to a weight vector (condition 2), namely the computational power of each miner (and no miner has an overbalance in computational power). The result is an empty CS-core of S, so there will always be a coalition which benefits from deviating from the current situation S. In short, there exist no distribution of the payoffs that will keep the miners in every coalition satisfied.

2.5 $\mathbb{D}_{MS}$-CS-core

The emptiness of the CS-core is a strong notion. An imputation mustn't be blocked by any possible coalition. Yet, there might be situations where certain subsets of agents can't form a new coalition. We will constrain the number of possible partitions by only allowing the coalitions to split or merge. Therefore, we will introduce the concept of $\mathbb{D}$-stability. We define $\mathbb{D} : CS(M) \to 2^M$ as the defection function. This function maps each partition to a set of possible coalitions. This means that a coalition C can only deviate from the current partition of S if $C \in \mathbb{D}(S)$. With this additional demand we might find an imputation that isn't blocked by any coalition.

There are two options for a coalition C:

- a subset of C can*merge* with a subset of another coalition $\{C' \cup D' \mid C' \subset C, D' \subset D\}$;
- C can *split* into two subsets $C = C_1 \cup C_2$.

An imputation x is $\mathbb{D}_{MS}$-stable if for every $C \in \mathbb{D}(S)_{MS}$ we have $x(C) \geq v_{S_C}(C)$. The $\mathbb{D}_{MS}$-CS-core exists of all the $\mathbb{D}_{MS}$-stable imputations that are associated with S.

Example 2.5.1. To illustrate that there are less constraints on an imputation x for the $\mathbb{D}_{MS}$-CS-core, consider a partition $S = \{\{a\}, \{b\}, \{c\}, \{d, e\}\}$. While checking if an imputation x is in a core of S, we must consider each possible alternative partition (S_C) of S. The partition $S_2 = \{\{a\}, \{b\}, \{c, d\}, \{e\}\}$ is the result of a split and a merge of the coalitions in S. Therefore, it will be considered as a possible deviation from the current situation S for both the CS-core and the $\mathbb{D}_{MS}$-CS-core.

However, the partition $S_3 = \{\{a,b,c\},\{d,e\}\}$ needs to be considered for the CS-core of S, but not for the $\mathbb{D}_{MS}$-CS-core, because $\{a,b,c\}$ is the result of two merges. Thus, due to $\mathbb{D}$-stability, there are less possibilities that can block the imputation x. △

In order to prove the emptiness of this core, we will give a proof similar to the one of theorem 2.4.2. Only besides finding a coalition D such that $v_{S_C}(D) > x(D)$, we must also show that D is a result of either a split or a merge: $D \in \mathbb{D}(S)_{MS}$. Therefore, we first have to define another condition [4]:

3. **Monotonic**
   If for every $S \in CS(I)$ with $S = \{C_1, C_2, \dots, C_m\}(m > 2)$ and $v_s(C_1) \leq v_s(C_2) \leq \cdots \leq v_s(C_m)$, then for $C = C_1 \cup C_2$ we have $v_s(C_1) + v_s(C_2) < v_{S_C}(C)$.
   The two lowest pools will always have a larger payoff when they collaborate by merging.

Now we can prove the emptiness of the $\mathbb{D}_{MS}$-CS-core using the three conditions. Again, our weight vector will be the computational power.

Theorem 2.5.2 (Emptiness of $\mathbb{D}_{MS}$-CS-core)**.** *Let v be a partition function of a coalitional game with coalition structures and let p be a weight vector such that $\forall i \in I : p_i < \frac{1}{2}$. If v is constant-sum (1), nonlinear with respect to p (2) and monotonic (3), then for every coalition structure the $\mathbb{D}_{MS}$-CS-core is empty.*

Proof. Let v be a partition function as described in the theorem. Again we assume that $M = \{1, \dots, n\}$ and for every $S \in CS(M) : \sum_{A \in S} v_S(A) = 1$ without loss of generality. Let $S \in CS(M)$ be the coalition structure. We consider an imputation x associated with S and show that there is a $D \in \mathbb{D}_{MS}(S)$ such that $v_{S_C}(D) > x(D)$. We will consider two cases for the number of coalitions:

- $|S| < 2$
  We either have one or two coalitions. This is similar to the situation in the proof of theorem 2.4.2, where we first had the grand coalition M and then considered a situation with two coalitions: $D = M \setminus \{n\}$ and $\{n\}$. Since the conditions of constant-sum and nonlinearity still apply, we can use the result of the proof of 2.4.2:
  $$v_{S_C}(D) > x(D).$$
  All we need to verify, is that $D \in \mathbb{D}_{MS}(S)$. If $|S| = 1$, then D is a subset of M which has chosen to split. If $|S| = 2$, then two subsets have chosen to merge into D. In both cases $D \in \mathbb{D}_{MS}(S)$. Since $v_{S_C}(D) > x(D)$ and x was chosen arbitrarily, there exist no imputation x which is $\mathbb{D}_{MS}$-stable.

- $|S| \geq 3$
  Let S be a partition of $m \geq 3$ coalitions. We will order these coalitions such that the coalition with the lowest payoff will become the first coalition and vice versa: $v_s(C_1) \leq v_s(C_2) \leq \cdots \leq v_s(C_m)$ for $S = \{C_1, \dots, C_m\}$. One can already suspect that coalitions C_1 and C_2 will be used for monotonicity.

  Let $D = C_1 \cup C_2$. Because D is a merge of two coalitions we know that $D \in \mathbb{D}_{MS}(S)$. All that is left to prove is $v_{S_C}(D) > x(D)$.

  We know that the payoff in the current partition S is the same as the reward for the entire coalition as defined by the imputation x. In short $x(C) = v_s(C)$ and so
  $$\begin{aligned} x(D) &= x(C_1) + x(C_2) \\ &= v_s(C_1) + v_s(C_2) \\ &< v_{S_C}(D). \end{aligned}$$
  The last inequality follows from the monotonicity of v. We have found our coalition $D \in \mathbb{D}_{MS}(S)$ such that $v_{S_C}(D) > x(D)$.

In the first case we know that there is either a merge or a split, but both result in a higher payoff. For $|S| \geq 3$, the accumulated payoff of the two lowest coalitions C_1 and C_2 increases when they merge. In both cases there exists no imputation which is $\mathbb{D}_{MS}$-stable. The $\mathbb{D}_{MS}$-CS-core is therefore empty. □

From this theorem follows that under the given conditions on v there will always be a split or merge that will result in a higher payoff. This means that it will always be better for some miners to switch pools, never achieving an equilibrium in this kind of Miner Network.

3 Delay: the key parameter

This isn't the end of the proof of the emptiness of the core. Up to now we have imposed conditions on our payoff function v in order to prove the emptiness of the CS- and $\mathbb{D}_{MS}$-CS-core. However, we can't always be certain these conditions apply. We need a more general approach.

This is where the delay within a pool (d) comes into play. For certain constrains on d the conditions of 3.2.4 will be satisfied. That is, we will need to find the boundarie(s) for this delay: $d < \min\{d_1, d_2\}$. If d meets this requirement, then we will see that the partition function of a coalitional game $\mathbf{C} = \langle M, P, D, d, \beta \rangle$ is constant-sum, monotonic and nonlinear with respect to P (our weight vector). From now on, we consider the share of a miner in the network as our partition function, $\gamma_C = v_S(C)$, which we want to satisfy the beforementioned conditions. As follows from theorem 2.5.2, the $\mathbb{D}_{MS}$-CS-core of every coalition structure will be empty.

3.1 Boundary d_1

Before we head on to the first boundary of d, we first need to define a new denotation. We define $\alpha_s(C)$ as the probability that *a block that was mined by one of the miners of C will also end up in the longest chain.* This resembles the definition of $\gamma(\Gamma)_C$ (the pools share in the longest chain), which becomes even more evident in the following equivalence:

$$\beta \cdot \gamma(\Gamma)_C = \lambda \cdot p(C) \cdot \alpha_s(C). \tag{4}$$

On the left hand side we have the number of blocks mined per second (β) times the pools share in the longest chain ($\gamma(\Gamma)_C$). Together this is the expected number of blocks mined by the miners of pool C, which will be added to the longest chain.

The right hand side is another way to calculate the same thing. The collective computational power of the coalition ($p(C)$) is multiplied by the expected number of blocks mined by the entire network (per second) (λ) and the probability that a block that was mined by C will end up in the longest chain ($\alpha_s(C)$).

We will now consider the case of $d = 0$. This means there is no delay between the miners and their poolmanagers, so every pool is in communication similar to a solo miner. They act as one individual. Intuitively, this means that if the computational power of a coalition C_i greater is than the one of C_j, then chances are more likely that a block, which is added to the longest chain, is mined by one of the members of C_i. They have more computational power and can therefore mine faster. For this reason,

$$p(C_i) > p(C_j) \longrightarrow \alpha_s(C_i) > \alpha_s(C_j). \tag{5}$$

In consequence we can define d_1 as the maximal delay for which (5) applies. This is in accordance with our results in figure 3a. Even when the difference in computational power in our Two Miner Network is small, the share of γ_1 becomes quickly larger when $D > 0$. We can use this information together with equation (4) to prove the following lemma.

Lemma 3.1.1. *Let $\boldsymbol{C} = \langle M, P, D, d, \beta \rangle$ be a miner coalitional game with coaliton structures. Assume that $D, \beta > 0$ and $\forall i : p_i \in (0, 1)$ (so no miners are left out or too dominant). For every $S \in CS(M)$ and $\forall C_i, C_j \in S$, set d_1 as the maximal delay for which $p(C_i) > p(C_j)$ implies $\alpha(C_i) > \alpha(C_j)$.*

If $d < d_1$, then $\forall S \in CS(M)$ and $\forall C_i, C_j \in S$:

$$\textit{if } p(C_i) > p(C_j), \ \textit{then } \frac{v_s(C_i)}{p(C_i)} > \frac{v_s(C_j)}{p(C_j)}. \tag{6}$$

Proof. We fix M, P, D and β. Because of equivalence (4) we know that λ is also fixed. Let the delay within pools be bounded by $0 < d < d_1$. Assume that for $i, j \in M$ we have $p(C_i) > p(C_j)$. Then also

$\alpha_s(C_i) > \alpha_s(C_j)$. By rewriting equation (4) we obtain

$$\frac{\gamma_C}{p(C)} = \frac{\lambda}{\beta}\alpha_s(C).$$

Remember that $v_s(C_k) = \gamma(\Gamma)_{C_k}$. The payoff of a coalition is equal to its share in the network. Consequently, we get:

$$\frac{v_s(C_i)}{p(C_i)} = \frac{\gamma_{C_i}}{p(C_i)} = \frac{\lambda}{\beta}\alpha_s(C_i) > \frac{\lambda}{\beta}\alpha_s(C_i) = \frac{v_s(C_j)}{p(C_j)}.$$

□

If d is larger than d_1, then it is possible that a larger computational power of a coalition doesn't always result in a higher chance to enlarge the blockchain. In other words: there is no certainty that switching to another coalition with more computational power will lead to a larger payoff. Therefore, d_1 is one of the boundaries of d.

3.2 Boundary d_2

The inequality shown by (6) will be useful for lemma 3.2.1, where we will observe different miner networks. Both networks have a within-delay of zero. This unusual value for d is chosen, because it is the basic case which we use to define the conditions for d_1 and d_2. For example, before we defined d_1 in lemma 3.1.1 we started reasoning starting from the case of $d = 0$ and from this situation we concluded that in order to prove (6), it was necessary to first show that $p(C_i) > p(C_j) \to \alpha_s(C_i) > \alpha_s(C_j)$. Hence, we defined d_1 to satisfy this condition. The next boundary is defined likewise, but before we head on to this, let's first discuss the meaning of lemma 3.2.1, derived from Lewenberg et al. [4].

Lemma 3.2.1. *Let $\Gamma_k = \langle M, S, P, D, 0, \lambda_k\rangle (k = 1, 2)$ be two miner networks, with*

$$\lambda_1 > \lambda_2, \quad S = \{C_1, \ldots, C_m\} \quad (m > 2), \quad p(C_1) \leq \cdots \leq p(C_m) \text{ and } p(C_1) < p(C_m).$$

Then, $\gamma(\Gamma_1)_{C_1} + \gamma(\Gamma_1)_{C_2} < \gamma(\Gamma_2)_{C_1} + \gamma(\Gamma_2)_{C_2}$.

Proof. For the proof see [4]. □

Remark 3.2.2. Since $p(C_1) < p(C_m)$, we know that there is no uniform distribution. Though, we still don't know if $p(C_1) = p(C_2)$ or $p(C_1) < p(C_2)$. Furthermore, $\lambda_1 > \lambda_2$, so the first Miner Network mines more blocks per second than the other network. The result of the lemma, however, is the complete opposite: the share of the two lowest coalitions in the first network are less than the share of the same coalitions in the less achieving second network. One would expect a higher payoff when a network is more efficient. Apparently, the more succesful a network is, the less the lower coalitions obtain.

Since the proof is omitted, we will explain the result of the lemma by heuristic reasoning. When more blocks are mined, it is also more likely that more blocks will be added to the longest chain. Being the coalitions with the lowest computational power, C_1 and C_2 don't have much chance that one of their miners will add a block to the longest chain.

So despite the higher number of blocks being mined, the chances of coalitions C_1 and C_2 mining a block, which will added to the longest chain, are increasingly getting lower.

Now we are ready to define d_2. Let C_1, C_2 be the coalitions with the lowest computational power. From lemma 3.1.1 we know that if $p(C_i) > p(C_j)$, then $v_s(C_i) > v_s(C_j)$. Since $\forall i \in M : p(C_i) \geq p(C_1)$, we can conclude that $v_s(C_2) \geq v_s(C_1)$. We define d_2 as the maximal positive d for which $v_s(C_1) + v_s(C_1) < v_{S_C}(C)$, where $C = C_1 \cup C_2$.

The definition for d_2 seems like a typo, but remember that $v_s(C_2) \geq v_s(C_1)$. Therefore, $v_s(C_1)$ acts as a lower bound. We will use d_2 to prove that monotonicity is present in a network bounded by this within-delay.

Lemma 3.2.3. *Let $\boldsymbol{C} = \langle M, P, D, d, \beta \rangle$ be as in lemma 3.1.1. We define d_2 as the maximal positive d for which $v_s(C_1) + v_s(C_1) < v_{S_C}(C)$ for every coalition structure $S \in CS(M)$. Assume that $d < d_2$. Then for every $S \in CS(M)$ with $S = \{C_1, \ldots, C_m\}$ $(m > 2)$ and $p(C_1) \leq \cdots \leq p(C_m)$ we have $v_s(C_1) + v_s(C_2) < v_{S_C}(C)$ for $C = C_1 \cup C_2$.*

Proof. We fix M, P, D and β. Let $S = \{C_1, C_2, \ldots, C_m\}$ be the coalitional structure as defined in the lemma. We set $C = C_1 \cup C_2$ and use $S_C = \{C, C_3, C_4, \ldots, C_m\}$ to denote the new coalitional structure. We must now show that after this merge the payoff increases; $v_s(C_1) + v_s(C_2) < v_{S_C}(C)$.

Let's consider two events:

- $p(C_1) = p(C_m)$
  In this case we use the equality (4). Since $p(C_1) = p(C_m)$ and $p(C_1) \leq \cdots \leq p(C_m)$, it is clear that $p(C_1) = p(C_2)$. Therefore

$$\begin{aligned} \beta \cdot \gamma_{c_1} &= \lambda \cdot p(C_1) \cdot \alpha_s(C_1) \\ &= \lambda \cdot p(C_2) \cdot \alpha_s(C_2) \\ &= \beta \cdot \gamma_{c_2}, \end{aligned}$$

  and so $\gamma_{c_1} = \gamma_{c_2}$. We know that d_2 is the maximal positive d for which $v_s(C_1) + v_s(C_1) < v_{S_C}(C)$. Thus, we can conclude

$$\begin{aligned} v_s(C_1) + v_s(C_2) &= \gamma_{c_1} + \gamma_{c_2} \\ &= \gamma_{c_1} + \gamma_{c_1} \\ &= v_s(C_1) + v_s(C_1) < v_{S_C}(C). \end{aligned}$$

- $p(C_1) < p(C_m)$
  For this case we need the result of lemma 3.2.1. Let $\Gamma_1 = \langle M, S, P, D, d, \lambda_1 \rangle$, $\Gamma_2 = \langle M, S, P, D, d, \lambda_2 \rangle$ be the Miner Networks and $\Gamma_{2,S_C} = \langle M, S_C, P, D, d, \lambda_2 \rangle$ the second Miner Network after the merge. Let $\lambda_1 > \lambda_2 > 0$ such that it holds that $\beta(\Gamma_1) = \beta(\Gamma_{2,S_C}) = \beta$. This means the second network was updated such that the desired rate of block addition to the longest chain was reached.

  Still, $\lambda_1 > \lambda_2$. This follows from the higher efficiency of the network Γ_{2,S_C} than Γ_1, since Γ_{2,S_C} is the result of a merge. This shift has caused less competition, hence a more concentrated network arose.

  Now we know that $\lambda_1 > \lambda_2$, $p(C_1) \leq \cdots \leq p(C_m)$ and $p(C_1) < p(C_m)$ as assumed. All the conditions for lemma 3.2.1 are satisfied and so we can use the result for Γ_1 and Γ_2:

$$\begin{aligned} v_s(C_1) + v_s(C_2) &= \gamma(\Gamma_1)_{C_1} + \gamma(\Gamma_1)_{C_2} \\ &< \gamma(\Gamma_2)_{C_1} + \gamma(\Gamma_2)_{C_2} \\ &< \gamma(\Gamma_{2,S_C})_C = v_{S_C}(C). \end{aligned}$$

  The last inequality is justified by the loss of conflict. When two pools merge, there will be less competition. In addition, they have more hash power, so the new partition of S_C will be beneficial for both C_1 and C_2.

□

This lemma shows that if d is bounded by some d_2, then every miner coalitional game with coalition structures (for which $D, \beta > 0$) will be monotonic. We recognize this as one of the conditions leading to an empty core. That's why the next theorem doesn't come as a surprise:

Theorem 3.2.4. *Let $\boldsymbol{C} = \langle M, P, D, d, \beta \rangle$ be as in lemma 3.1.1. If $d < \min\{d_1, d_2\}$, then the partition function of $\boldsymbol{C}$ is constant-sum, monotonic and nonlinear with respect to P.*

Proof. Let $d < \min\{d_1, d_2\}$. We will prove the conditions piece by piece.

- Constant-sum: for the partition function v we have for every $S \in CS(M)$ that $\sum_{C \in S} v_s(C) = \sum_{C \in S} x(C) = 1$.

- Monotonicity: follows from lemma 3.2.3, because $d < d_2$.
- Nonlinear with respect to P: let $S \in SC(M)$ be a coalition structure with $\max_{C \in S} p(C) > \min_{C \in S} p(C)$. Furthermore, let $C_i \in \arg\max_{C \in S} p(C)$ be the coalition with the highest computational power. All that is left to prove is that $v_s(C_i) > p(C_i)$.

  Lemma 3.1.1 ensures that $\forall C \in S : \frac{v_s(C)}{p(C)} < \frac{v_s(C_i)}{p(C_i)}$. By using this we obtain:

$$\begin{aligned}\sum_{C \in S} v_s(C) &= \sum_{C \in S} p(C) \cdot \frac{v_s(C)}{p(C)} \\ &< \sum_{C \in S} p(C) \cdot \frac{v_s(C_i)}{p(C_i)} \\ &= \frac{v_s(C_i)}{p(C_i)} \sum_{C \in S} p(C) \\ &= \frac{v_s(C_i)}{p(C_i)}.\end{aligned}$$

  Since v is constant-sum with $\sum_{C \in S} v_s(C) = 1$, we can conclude:

$$\frac{v_s(C_i)}{p(C_i)} > 1 \quad \text{thus} \quad v_s(C_i) > p(C_i).$$

□

This theorem states that for every miner coalition game it is proven that if the delay between miners within each pool is low enough, the partition function of **C** has three characteristics: constant-sum, monotonic and nonlinear with respect to P. Due to these three conditions we can safely conclude that the $\mathbb{D}$-CS-core is empty for every coalition structure:

Corollary 3.2.5. *Let $\boldsymbol{C} = \langle M, P, D, d, \beta \rangle$ be as in the theorem 3.2.4. If for all i : $p_i \in (0, \frac{1}{2})$, then $\forall S \in CS(M)$ the $\mathbb{D}_{MS}$-CS-core is empty.*

Conclusion

In this thesis we have discussed the basics of the Bitcoin Network and modified it to a cooperative game with coalitional structures. The payoff was divided proportionally according to the computational contribution of each miner. We have examined the importance of three conditions on a Miner Network: constant-sum, nonlinearity and monotonicity. The claims of constant-sum and nonlinearity were neccessary in order to prove the emptiness of the CS-core. Still, not every coalition can deviate from the current coalitional structure. That's why monotonicity was required. We showed that the payoff will also increase for a coalition in the case of a merge or split. Thus, the $\mathbb{D}_{MS}$-CS-core is also empty. This means that under the aforementioned conditions on our payoff function v, there isn't an imputation which will keep every miner content with his or her share.

As it turns out there is one factor which ultimately controls the emptiness of the core: the delay within pools. Corollary 3.2.5 says if d is low enough, then the three conditions will also be met. Thus the $\mathbb{D}$-CS-core will again be empty. A split means that the collective computational power of the seperated coalition apparanlty is strong enough to hold on his own. A merge of two coalitions will result in a more concentrated network, with less competition and more hash power. With a low delay within coalitions the quick communication between miners within a pool ensures that it will always be beneficial for at least one miner to switch pools or create a new one. Therefore, there exist no cooperative miner game with coalition structures that doesn't lead to an empty $\mathbb{D}$-CS-core.

A Proof-of-work

Let us consider the network as a block tree. The longest chain in this tree is called the block chain. It is a shared data structure, so every miner has the possibility to add a block to the chain. The chain contains all the verified transactions of bitcoins.

At one end is the first block (the genesis block), which was created together with the Bitcoin network. Every other block is (in)directly linked to the genesis block and contains a set of recent transactions it approves. Furthermore, each block has a header. This contains a pointer (which indicates the preceding block), a compressed representation of the set of transactions and a nonce. We will now introduce and clarify a few concepts, meanwhile explaining the process of mining step-by-step:

Mining is the process of adding a new block to the block chain. If the block becomes part of the longest chain, then the miner will be rewarded with bitcoins. The header of each block contains a nonce. This is a byte string. When the nonce is inserted into a cryptographic hashfunction, it will produce a hash (a large number of a fixed length). The value of the nonce determines with how many zero's the hash begins. This is important because the hash of each block has to be lower than a given threshold. When the right nonce is found, the block becomes part of the block chain and all the transactions included in the block become valid. The nonce is part of the block, so other miners can verify if the creator really solved the problem. The resulted hash is called the proof-of-work and will be used as the block's identity (which is needed to mine the next block). [[2] p2,3]

Since, cryptographic hashes are one-way functions, a miner can only find a desired nonce by arbitrarily calculating the hash of a given block for all possible nonces. The same data will always produce the same hash and a small change in the data will result in a completely different hash. This important quality of the cryptographic hash functions is called *collision resistance*. This means that different input values will never obtain the same hash output. Therefore, two miners will never be able to produce exactly the same hash.

Retargeting means adjusting the threshold to stabilize the rate of block addition to the longest chain. A lower threshold reduces the number of nonces which meet the requirement of leading-zero bits. Therefore, the computational burden on the miners will increase, but they must not be disheartened. The goal of the network is that every ten minutes a single block is mined by the entire network. In order to maintain this design the threshold t is adjusted everytime 2016 blocks have been mined.

In conclusion, miners continuously try to solve cryptographic puzzles in the form of hash computation. They use the information provided by the header of the previous mined block to collectively determine the new threshold t and then try to find a nonce which will result in a hash below this threshold. When such a nonce is found, the proof-of-work is solved and other miners are able to verify if this nonce indeed does provide a hash with the given number of leading-zero bits. The new block is added to the chain and miners will use it's header to find a new nonce. After 2016 mined blocks a new threshold t will be determined and the mining continues.

B Notation overview

Notation	Meaning
$I = \{1, \ldots, n\}$	The grand coalition.
$v : 2^I \to \mathbb{R}$	The payoff function.
$x \in \mathbb{R}^{\|I\|}$	An imputation.
$C \subset I$	A coalition.
$S = \{C_1, \ldots, C_m\}$	The partition function of I into pools.
$S_C = \{C\} \cup \{\{D \setminus C\} \mid D \in S, \{D \setminus C\} \neq \{\emptyset\}\}$	The new partition after a creation of a new coalition C.
$v(S, C) = v_s(C)$	The utility of a coalition C under the partition of other agents, as given by S.
$v(S_C, C) = v_{S_C}(C)$	The utility of a coalition C after split(s) and/or merge(s) resulting in a new coalitional structure S_C.
$\mathbb{D} : CS(I) \to 2^I$	The defection function.
$M = \{1, 2, \ldots, n\}$	The set of miners (and the grand coalition in this model).
$P = \{p_1, p_2, \ldots, p_n\}$, with $\sum_{i=1}^n p_i = 1$.	Distribution of the computational power, where p_i is the fraction of the computational power of agent i compared to the other agents.
D	The delay in communication between pools.
d	The delay in communication within a pool.
λ	The expected number of blocks mined by the netwerk (per second).
$\Gamma = \langle M, S, P, D, d, \lambda \rangle$	A miner network.
β	The desired rate of block addition to the longest chain (per second).
$\beta(\Gamma)$	The actual rate of block addition to the longest chain (per second). Γ is the given miner network.
$\mathbf{C} = \langle M, P, D, d, \beta \rangle$	A Miner Coalitional Game with Coalition Structures.
$\gamma_i(\Gamma) = \gamma_i \in [0, 1]$	The probability that a block belonging to the longest chain was mined by miner i in a network Γ.
$w \in \mathbb{R}^{\|I\|}$	A weight vector.
CS-core	$\{x \in \mathbb{R}^{\|M\|} \mid x_i \geq 0, \forall C \subset I : x(C) \geq v_{S_C}(C)\}$.
$\mathbb{D}_{MS}$-CS-core	$\{x \in \mathbb{R}^{\|M\|} \mid x_i \geq 0, \forall C \subset I : x(C) \geq v_{S_C}(C), C \in \mathbb{D}_{MS}\}$.
$\alpha_s(C)$	The probability that a block that was mined by one of the miners of C will also end up in the longest chain.

References

[1] J. Chokun. *Who Accepts Bitcoins As Payment? List of Companies, Stores, Shops*, 6 2016. URL: `https://99bitcoins.com/who-accepts-bitcoins-payment-companies-stores-take-bitcoins/`.

[2] C. Decker and R. Wattenhofer. *Information Propagation in the Bitcoin Network*. 13-th IEEE International Conference on Peer-to-Peer Computing.

[3] B. Kelly. *The Bitcoin Big Bang: How alternative currencies are about to change the world*. John Wiley & Sons, 1 edition, 1 2015. ISBN 978-1-118-96366-1.

[4] Y. Lewenberg, Y. Bachrach, Y. Sompolinsky, A. Zohar, and J. Rosenschein. *Bitcoin Mining Pools: a cooperative game theoretic analysis*. In Bordini, Elkind, Weiss, and Yolum, editors, *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 5 2015.

[5] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 41 William St # 1, Princeton, NJ 08540, USA, 2016. ISBN 9780691171692.

[6] Y. Somplinsky and A. Zohar. *Secure high-rate transaction processing in Bitcoin*. In T. O. Rainer Bhme, editor, *Financial Cryptography and Data Security*. 19th International Conference, Springer, 1 2015.

[7] A. Zohar. Bitcoin: Under the hood. *Communications of the ACM*, 58(9):104–113, 9 2015.