university of groningen / faculty of science and engineering

Game-theoretic analysis of a block withholding attack on the Bitcoin consensus protocol

Bachelor's Project Thesis

Natalia Karpova, n.karpova@rug.nl,
Supervisor: Prof Dr D. Grossi

Abstract:

Bitcoin is the most popular cryptocurrency nowadays. It is based on a decentralized peer-to-peer system built on blockchain data structure. The Bitcoin network incorporates some rules (consensus protocol) under which participant in the network search for new Bitcoins and develop the blockchain. Previously it was believed that the Bitcoin consensus protocol incentivizes participants to act in accordance with the protocol since it will give them the best outcome. Recent papers showed contrary findings. In this paper one type of attack - the block withholding attack, - is researched. A simulation with pools that perform block withholding attack against each other was replicated. The convergence analysis showed that the amount of pools have a significant positive influence on the convergence time. The simulation was then extended by allowing miners to switch between pools or mine solo. Contrary to the previous results, in the extended simulation the amount of pools does not have a significant influence on convergence time, however, a greater number of miners seems to result in an increase in convergence time. With extreme amount of miners in the system it seems unfeasible to find a Nash equilibrium due to computational complexity which is representative of a real state of the Bitcoin network. The modified definition of Pure Price of Anarchy, the value that intuitively measures the amount of computational power wasted on attacks, is proposed based on the extended simulation findings.

1 Introduction

Bitcoin is the world's most famous and most used cryptocurrency, it has gained extreme popularity over the recent years. The market capitalization of bitcoin was approximately 150 billion dollars by the end of 2017, whereas the market capitalization of gold was about 7.5 trillion dollars (Coven, 2017). This has some economists discussing the idea of using Bitcoin as a "store of value" (Cochrane, 2017). Nevertheless, in order for Bitcoin to become an economic asset, it has to be secure. The main idea behind Bitcoin security lies in a robust incentive system. This system incorporates a consensus protocol to which users are required to provide an expensive proof-of-work for which they are rewarded later (Nakamoto, 2009). Ideally, the whole system is supposed to incentivise participants to act in accordance with the protocol (in this case users are believed to achieve the best outcome for themselves). However, it is not yet known whether the Bitcoin consensus protocol is indeed incentive compatible, i.e. whether following the protocol is in the true interest of the miner or not. As a counter argument to this idea Luu (2015) showed that honest participants in the Bitcoin network are vulnerable to attacks which result in unverified transactions on the blockchain (Luu, Teutsch, Kulkarni, and Saxena, 2015b). Eyal (2015) has introduced "the miner's dilemma" which shows that Bitcoin networks are prone to a specific type of attack: the block withholding attack. In this paper a possibility and consequence of a block-withholding attack in a Bitcoin system will be discussed. First, the Bitcoin network and its components will be explained. This is followed by the methods used for researching the block withholding attack. Then the analysis of the gathered data is present. The paper ends with the discussion of the findings and the conclusion.

2 Preliminaries

2.1 The Bitcoin network

A Bitcoin network is an open distributed system that uses a *blockchain* as a tool for storing user's transactions. The blockchain is a data structure that consists of singly linked blocks of records about user's transactions. A transaction is a transfer of Bitcoins committed by the owner of these Bitcoins from their account to another one. Coherent hashing of blocks and transactions is used to link blocks into one chain, and to secure the resulting blockchain.

At any point in time there exists a main blockchain in a network which has the most amount of blocks in it.

Everyone is able to join a network at any time by accepting all previous entries on the network blockchain. The Bitcoin network participants are called *miners*.

Each miner in the network is able to make a transaction. Transactions that do not conflict with the previous transactions stored on the blockchain are serialized in the chain. Bitcoin transactions are secured by hashing techniques that guarantee that only true owners of the Bitcoin are capable of transferring it. Verification of a newly created transaction is done by other miners. Verification requires a relatively small amount of computational power and time from a miner. Only once a majority of the miners on the network accept the transaction, it is added to the chain. At the same time all users are competing with each other in order to be the first to put on a new block. Doing so requires a miner to exert enough resources to solve extremely difficult cryptographical puzzles. The first user to solve this puzzle is allowed to add a new block to the blockchain, and is rewarded according to their contribution. All other users are required to verify the newly added block based on their own copy of the blockchain. Malicious blocks that do not agree with the hash structure of the blockchain are rejected in such settings.

2.2 Consensus protocol

In general terms, the consensus protocol is used to achieve an overall agreement on a single value in a distributed system. In the Bitcoin system all miners reach agreement based on the *Nakomoto proof of work (consensus) protocol* (Nakamoto, 2009).

The *proof of work* is a piece of data which is extremely costly to produce but relatively easy to verify. In the Bitcoin network this is represented by a block. While mining a new block, all miners are trying to produce a proof of work by exerting computational power that is spent on hashing a new block. Each block contains a *nonce* field that is adjusted by miners. A nonce field indicates the amount of work spent on generating a new block. For a new block to be valid its nonce must hash to a value that is less than the current network's target. The network target values are low which makes it difficult to generate new blocks.

Miners are obligated to broadcast the block once they found it. A validity of a new (broadcasted) block must be verified by the network. Verification of a proof of work, generated during the creation of a new block, is decentralized. This implies that miners verify new block on their side and either accept it to their representation of the network or discard it. The fate of the block is dependent on the choice made by the majority of the network. This is also referred to as a *distributed consensus* (Narayanan, Bonneau, Felten, Miller, and Goldfeder, 2016). Verification is almost cost free (relative to the cost of mining) for a miner, and hence, Bitcoin miners are trusted to verify all newly added blocks and transactions for the sake of "common good". A block that is accepted by the majority of the network is added to the chain, and the mining for the next block begins. If miners accept a newly-generated block without verification, the block might appear to be malicious and might end up being added to the chain (if majority of honest miners will cheat and accept it without verification). It is not in the interests of honest miners to have malicious blocks in the chain since this will threaten Bitcoin security. For this reason, honest miners are believed to verify newly-generated blocks.

2.3 Pools

Individuals usually require years to mine a new block (Swanson, 2013). Therefore, miners tend to organize themselves into *mining pools*. A pool is a group of miners who attempt to mine a block together. Revenue for a block is distributed among all pool members proportionally to their mining

power. In both cases: pool and solo mining, the expected revenue of a miner is the same. However, a pool has much larger power and probability of mining a new block than a single miner, thus, frequency of (smaller) revenues is higher when mining in pool. Hence, pool mining allows a stable frequent income.

Each pool has a designated coinbase recipient, a *pool manager*. The pool manager controls the pool: they are responsible for dividing ranges of values for hashing between all pool members, posting blocks found by pool members on the network, and receiving network total revenue for a block. The revenue earned from mining is shared among all miners in the pool in such a way that every miner is rewarded proportionally to the computational power they exerted. This is also controlled by the pool manager. Miners are able to show the amount of work they have done by outputting *shares*. Shares are near-valid blocks which nonce value is less than the partial target. The partial target is set by the pool manager and is chosen to be relatively large such that miner's shares arrive frequently enough for the pool manager to be able to accurately estimate a miner's contribution. Miners show their shares to the pool manager in order to prove that they have indeed been working on their task, and are rewarded accordingly.

Participation in pool mining is typically not free: the pool manager charges for their services, a certain percentage cut of the block reward earned by miners (Rosenfeld, 2011). This charge is called the *pool fees*. Pool fees are usually fixed within a pool, however, they may vary depending on the exact payment system upon which a pool miner and pool manager agree (Narayanan et al., 2016). The exact trade-off payment model is not important for this study and, hence, will not be explained in more details.

There exists open (anyone is free to join) and closed (limited access) pools. As well, pools require members to pay different contribution fees. Once in a while each pool will post its per-miner revenues to the network. Together with contribution fees this makes some pools to be more profitable for miners than the other pools. This makes the basis for the *block withholding attack*.

2.4 Block withholding attack

A block withholding attack is an attack performed by a pool against a different pool (Rosenfeld, 2011). A miner from the attacking pool registers in the victim pool. This miner mines honestly in the new pool, and regularly sends its partial shares to the pool manager. However, if an attacking miner finds a new block it simply discards it. This leads to the reduction in the attacked pool total revenue and, consequently, per-miner revenue.

Notably, this kind of attack influences the attacker's revenue as well: the attacker suffers from the reduced pool revenue just like all other pool members. In turn, the attacker earnings appear to be less than its share of the total mining power in the network. Therefore, a block withholding attack is used only to sabotage competitive pools (Eyal, 2015).

It is worth mentioning that a pool is able to detect that it is attacked by comparing its total shares with the full proof of work. Even after detecting an attack, a pool is not capable of identifying which members are involved in the block withholding attack. This is due to the fact that a full proof of work (i.e. new block) is produced with a very low frequency such that a pool cannot obtain statistically significant results about the sabotaging miners based solely on their frequent shares and rare full proofs of work.

Bitcoin systems seem to be prone in theory to block withholding attacks. This type of attacks is well-known in Bitcoin society though the efficiency of block withholding attacks and corresponding incentive of miners to perform them is a point of discussion (Luu, Saha, Parameshwaran, Saxena, and Hobor, 2015a). Nevertheless, there exists some evidence from both related research articles and real life examples that block withholding attack might be profitable for miners and, therefore, takes place in the Bitcoin network. For example, such an attack was performed against Eligius mining pool in 2014 (post, 2014). Eyal (2015) has introduced a pool game where pools change the rates of attacking miners in turns. He showed that block withholding attacks always take place in a system with two pools with fixed amount of members and rational miners. Alkalay-Houlihan and Shash (2019) quantitatively described the infiltration rates in a Nash equilibrium for the case mentioned above.

2.5 Contribution of the paper

Both Eyal, and Alkalay-Houlihan and Shash investigate in their papers a very restricted case of pool mining: miners are not allowed to switch between pools or form a new pool, whereas in a real life this is a common practice. Additionally, these studies do not describe a general case: whether a game with any number of differently sized pools converges to a pure Nash equilibrium. Convergence is dependent on the infiltration rates. The rate is between two pools, it is calculated as the amount of miners from pool A who join pool B with the intention of sabotaging. A converging rate no longer changes after an arbitrary amount of time.

This paper simulates pool games with an arbitrary number of pools and miners in a system where miners are not tied to one pool, and investigate the consequences of block withholding attacks in such scenario. This paper explores the existence and a form of a Nash equilibrium, relations between convergence time and amount of pools and/or miners, and the value of price of anarchy in such simulation with different initial settings. In order to achieve this the paper analyses synthetic data generated by the simulation of Bitcoin network. The simulation was made according to Eyal's methodology. The performance of the replicated simulation was tested based on Alkalay-Houlihan's and Shash's findings.

3 Material and methods

Everything was set up on Dell G5 15 laptop with 8th generation Intel Hex-Core CPU's and 24 GB of physical memory (RAM). The laptop runs under 64-bit Windows 10. Simulation was written in Java 10 with the use of JBlas linear algebra library together with apache maven 3.6.0 which acts as a tool for building the project.

The development of simulation consist of several steps. First, the simulation from Eyal's (2015) paper was replicated. Second, the simulation was tested using Alkalay-Houlihan's and Shash's findings. Third, the simulation setting were advanced by allowing miners to change pools or to leave their pools and mine solo / organize their own pool. These steps are discussed in depth below. In this section, shares of a miner is referred as a partial proof of work and mined block is referred as a full proof of work. The full code can be found at https://github.com/nata1y/pool-mining.

3.1 Replicated simulation

The simulation takes the amount of pool miners, the amount of solo miners, the amount of pools, and the amount of repetitions from a user input. Then, the pool miners are distributed between all pools, the exact distribution is based on user preferences and can be adjusted according to certain requirements. The simulation is composed of different steps. Each step happens during a single turn of the simulation. A turn is considered as 1 time unit. A simulation is labeled as converging if no pool changes their infiltration rate. The steps are shown below:

1. At the start of the turn the solo miners will do their work.
    i Each solo miner will work on their task during this time step.
2. Afterwards, tasks are assigned and work takes place, for all pools, and each miner in the pool.
    i Now, all pool miners without a task get assigned a new task. The "level of difficulty" of a new task is decided based on the miner's partial proof of work (shares) from the previous step. Task difficulty is measured by the time required to complete this task.
    ii All pool miners will then work on their task.
3. Then, after all the work is completed. The proof-of-work submitted. The revenue is collected, and divided.
    i All pool miners send their proof of work (full and partial) to their pool manager. A proof of work for each miner is found by randomly drawing a number from a Poisson distribution *. This solution was

*Poisson distribution - probability distribution that represents the probability of events occurred in a fixed amount of time under the assumption that these events happen with some constant rate and are independent from each other (Haight, 1967). Poisson distribution is suitable for calculating an approximate probability distribution of certain events if the total amount of all possible events is extremely large.

proposed by Eyal (2015) in his paper and based on the fact that all miners are trying to "win a lottery" by randomly solving cryptographical puzzles. Since the system has a large number of possible mining outcomes where mining a block is a rare event, the probability of a miner to mine a (near-valid) block is described by a Poisson distribution. There were two different distributions implemented - one for the partial proof of work with a high expected number of occurrences (λ) and another one for the full proof of work with low λ.

ii Each pool collects the revenue from all its miners - honest and saboteur.

iii Each pool sends the revenue to all its miners and posts their own per-miner revenue.

4. The simulation concludes with a designated pool changing their infiltration rate.

   i A single pool picked up with a round-robin policy is able to change its infiltration rates of all other pools. Let's label this pool as p_i. The best infiltration rate for each "opponent pool" is decided based on the revenue density function introduced by Eyal (2015) in his study, as shown in Equation 3.2.

$$R_i = \frac{m_i - \sum_{j=1}^{p} x_{i,j}}{m - \sum_{j=1}^{p} \sum_{k=1}^{p} x_{j,k}} \tag{3.1}$$

$$r_i(t) = \frac{R_i(t) + \sum_{j=1}^{p} x_{i,j} r_j(t)}{m_i + \sum_{j=1}^{p} x_{j,i}(t)} \tag{3.2}$$

   Here, p is the number of pools in the simulation, m is a total amount of miners and m_i is amount of miners loyal to pool i. $x_{i,j}$ is an infiltration rate from pool i to pool j.

   Pool p_i chooses infiltration rates $x_{i,-}$ that maximize revenue density $r_i(t)$.

   ii If all pools decide on keeping their current infiltration rates, the simulation is labeled as converging.

The above algorithm is shown in Algorithm 3.1

Algorithm 3.1 Simulation iteration

```
Require: p ≥ 0 ∨ m ≥ 0
  for each soloMiner m do
    m.work()
  end for
  for each pool p do
    p.assignTasks()
    p.roundOfWork()
  end for
  for each pool p do
    p.updatePoW()
    p.collectRevenue()
    p.publishRevenue()
    p.sendRevenueToAll()
  end for
  take pool p which id == roundRobinNumber
  p.changeInfiltrationRates()
  simulation.checkConvergence()
```

3.2 Additional experiments

Miners are now allowed to change, join or leave a pool and mine solo. Every miner has their own revenue density. If a miner is a saboteur or a honest pool miner, their revenue density is equal to the revenue density of the pool they are mining at. Otherwise, if a miner mines solo, in order to calculate their revenue density, the miner is treated as a pool with one honest and no sabotaging members. When substituting these values into Equation 3.2, the solo miner revenue density can be calculated at every step of the game using Equation 3.3. This suggests that all solo miners have equal revenue density at the same turn. The denominator of the Equation 3.3 is the total mining power in the simulation at the current step. The total mining power of a simulation is recalculated at every turn.

$$r_i(t) = \frac{1}{m - \sum_{j=1}^{p} \sum_{k=1}^{p} x_{j,k}} \tag{3.3}$$

At every turn of the game, a miner is capable of comparing their own revenue density with the revenue densities of all other pools and solo miners. A higher revenue density corresponds to more profit for a single miner. Miners are considered rational, they will choose to join the pool with the highest revenue density or decide to mine solo if revenue density of a solo miner is the highest.

The earlier simulation was extended by allowing a fixed amount of miners to change their pool at every turn of the simulation. These miners, similarly to the pools, are picked with the round-robin policy. If all pools decide on keeping their infiltration rates and all miners decide on remaining in the same state (solo or pool), the simulation is labeled as converging.

Changes were made to the way that the best infiltration rate for a pool is calculated. Since miners are able to change pools, it is possible that all honest members will leave a weak pool and hence its true mining power will become 0. This leads to the potential problem with the way the revenue density is calculated in Equation 3.2. While trying to find a maximum of the revenue density function, a pool with some honest members in it will go through all possible infiltration rates permutations to all other pools in order to find the one that will give the highest expected revenue density. A permutation is the possible ways that the sabotaging miners can be split among victim pools. For the case when all pools will decide on attacking the weak pool (with no loyal members) with 0 infiltration rate, the denominator of the revenue density Equation 3.2 will be 0. Hence, this infiltration permutation will not ever be chosen by a pool. It is obvious that attacking a pool with no loyal members in it is a waste of own mining power. For this reason, the above described case is treated separately when choosing a pool's best infiltration rates to all other pools. If the expected revenue density value for some permutation of the infiltration rate is not a number, such value is caused by a pool with no loyal members in it, and the attacking pool already has the majority of all miners in them, this permutation is chosen as the current best infiltration rate.

4 Results and Evaluation

4.1 Replication

4.1.1 Replication verification

The simulation with 100 miners and 2 pools converges to a Nash equilibrium that is an attacking state. The outcome of the replicated simulation was compared to the results obtained by Alkalay-Houlihan and Shah. They quantitatively described infiltration rates in a Nash equilibrium for the simulation with 2 pools and zero outside mining power. The formula for finding such infiltration rates that was introduced in *Lemma 4* (Alkalay-Houlihan and Shah, 2019), as shown in Equation 4.1.

$$(x_1^*, x_2^*) = \begin{cases} (0,\ \frac{m_2}{2}) & \text{if } m_1 \leq \frac{m_2}{4} \\ (\frac{m_1}{2},\ 0) & \text{if } m_2 \leq \frac{m_1}{4} \\ (\frac{\sqrt{m_1 m_2}(2\sqrt{m_1}-\sqrt{m_2})}{\sqrt{m_1}+\sqrt{m_2}}, & \\ \frac{\sqrt{m_1 m_2}(2\sqrt{m_2}-\sqrt{m_1})}{\sqrt{m_1}+\sqrt{m_2}}) & \text{otherwise} \end{cases} \tag{4.1}$$

Here, m_1 and m_2 are the initial amount of miners in pool 1 and 2 respectively, and x_1^* and x_2^* are the infiltration rates in a Nash equilibrium from pool 1 to pool 2 and from pool 2 to pool 1 respectively.

The simulation outputs final infiltration rates for each possible distribution of 100 miners between 2 pools in accordance with Equation 4.1. Interestingly, there is one outlier case, namely, when $m_1 = 23$ and $m_2 = 77$ (opposite distribution is symmetrical) where the simulation does not seem to converge to a stable state (as has been shown by 2000 time steps). In this case, the simulation continuously switches between 2 different attacking rates: $(2, 40)$ and $(3, 39)$, the true attacking rate in this case, according to Alkalay-Houlihan and Shah, should be $(3, 40)$.

The above outcome may happen due to precision issues. The simulation was run several times with the same initial settings and distribution, whereas the precision of a revenue density value varied between 1 and 10 decimal points. It was found that when the precision is less than 6 decimal points the simulation does converge to a Nash equilibrium. The infiltration rates in a stable state when the precision of a revenue density is 5 decimal points are $(3, 38)$. With decreasing precision of the revenue density infiltration rates in a Nash equilibrium become less similar to the one stated in *Lemma 4*: $(3, 35)$ for precision of 4 decimal points and $(8, 9)$ for precision of 3 decimal points.

Since the simulation agrees with Alkalay-Houlihan's and Shah's results in all cases except the above mentioned one, this simulation is considered to be an acceptable replication for the rest of the research. Before building on the replicated simulation, several possible correlations have been researched.

4.1.2 Replication analysis

First of all, it is important to note that for the amount of pools in a simulation larger than 2, a simulation does not necessarily converge to a stable state for numerous miner distributions between all pools. Possible correlations between convergence time and the amount of pools, the amount of miners and/or initial distribution of miners between pools are discussed below.

Convergence time versus variable amount of miners has been plotted and is shown in Figure 4.1. The amount of miners in a simulation varies from 50 to 1550 with an increase of 50. Miners were distributed between all pools in such a way to ensure that each pool received almost equal amounts of miners depending on the total amount of miners in a simulation. For every amount of miners one simulation was run due to the fact that the distribution of miners is equal and, therefore, the same settings yield the same convergence time. The amount of miners does not seem to influence the convergence time in any particular way. There is one outlier case in the Figure 4.1 with 700 miners where the convergence time shown is 0. In this case, the simulation does not seem to converge to a stable state and hence no value for the convergence time can be obtained, therefore, zero convergence time is plotted. Due to the total amount of miners not always being divisible by 4, not all pools received equal amount of miners in some runs. Thus, the initial distribution of miners might influence simulation convergence time.

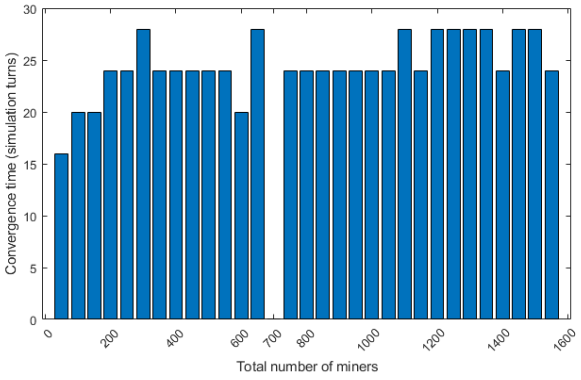

Figure 4.1: Convergence time versus variable amount of miners (50 to 1550 with increment of 50). All miners distributed almost equally among 4 pools.

Additionally, convergence time has been measured for different amount of pools (2 - 10) and a random distribution of 100 miners between these pools. Each simulation was repeated 10 times, and the average convergence time was plotted. Results are shown in Figure 4.2.

As can be seen from the Figure 4.2, convergence time is tending to increase, as the amount of pools in the simulation raises from 2 to 8, the time it takes to converge goes from 6 turns to approximately 39 turns. For the amount of pools 8 to 10 there is a slight decline in average convergence time from 39 turns to about 37 turns. Non-parametric version of the linear regression was performed. Kendall-Theil regression was calculated to predict convergence time based on amount of pools. A significant regression equation was found ($p = .004$). Simulation

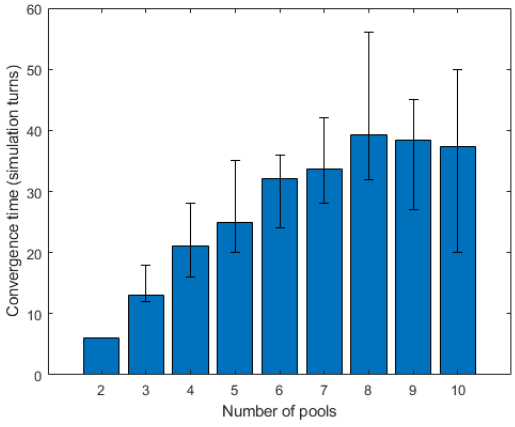

Figure 4.2: Convergence time for a simulation with 100 miners distributed randomly between 2 to 10 pools.

with a larger number of pools should be set in order to obtain reliable results. As for now, with the given simulation it is not feasible to obtain the values for convergence time for the amount of pools larger than 10 due to the computation needed.

Lastly, possible correlation between several different distributions of miners between all pools versus the simulation's convergence time is discussed.

In the case of 2 pools and 100 miners in a system, initial distribution of miners between pools does not seem to have a distinct effect on the convergence time. In most of the cases, simulation converges after 6 or 8 steps with a few outliers having convergence time 4 and 10.

As it was mentioned earlier, in the scenario with more than 2 pools in a simulation, there exist some situations when a simulation does not converge to a stable state. This complicates finding convergence times when a large amount of trials with different settings should be run consequently in order to obtain the results. For this reason, a test for all possible distributions of miners between more than 2 pools has not been performed.

A possible correlation between convergence time and a specific combination of initial distribution of miners has also been tested. The initial settings included 4 pools, and amount of miners varying from 50 to 1000 with increments of 50. There were 4 possible distributions of miners between 4 pools, namely, 1:1:1:1, 3:3:3:1, 4:4:1:1 and 2:1:1:1. This choice was based on the fact that if the difference in amounts of miners between pools is large than convergence is achieved fast since majority of mining power is in one pool and smaller pools might abstain from attacking huge pools at all (Eyal, 2015). Again, for every amount of miners and corresponding initial distribution one simulation was run due to the fact that the same settings yield the same convergence time. The results are shown in Figure 4.3. There is no obvious correlation between specific combinations and convergence time present in the Figure 4.3.

All in all, there was no influence found between the amount of miners, the amount of pools, and the amount of initial distribution of miners on the convergence time.

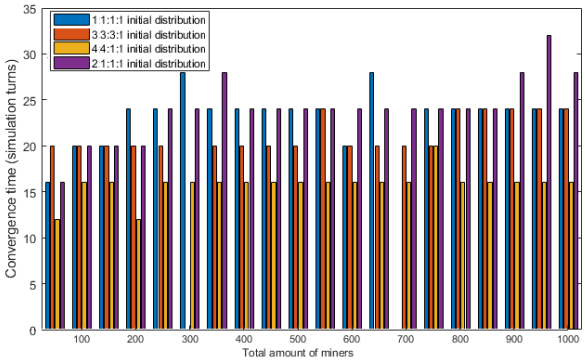

Figure 4.3: Convergence time for the 4 different distribution of variable amount of miners between 4 pools. For the amount of miners 700 and 1:1:1:1 distribution and amount of miners 300 and 3:3:3:1 distribution there are no convergence times shown since the simulation infinitely loops in these cases.

4.2 Simulation with dynamical pools

When miners are able to change pools, leave a pool and mine solo, or stop mining solo and join a pool, the simulation seems to always converge to a stable state. This was shown by a simulation with all possible distributions of 100 miners between 2 pools. As well, this was tested with the simulation with all possible distribution of 30 miners between 3 pools and zero outside mining power. Both experiments showed that all miners converge to one pool and mine there honestly.

In the first experiment all miners converge to the pool with the most initial mining power. Interestingly, there is one case where the simulation converges to a Nash equilibrium which is an attacking state: 100 miners equally distributed among 2 pools with zero outside mining power. In this case the simulation converges to the symmetrical infiltration rates $(25, 25)$. When outside mining power is added to the simulation, at some point a solo miner will decide to join a pool with the highest revenue density since it is more profitable for the miner. That will increase the pool's mining power and this results with all miners converging to this pool. When non-zero outside mining power is added to the simulation with 100 miners equally distributed among 2 pools, the simulation shows behavior as described above.

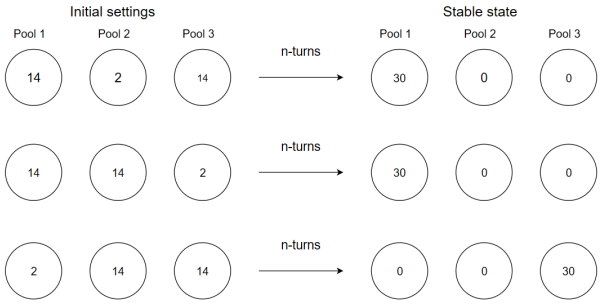

Figure 4.4: Convergence of 30 miners into 1 pool based on initial distribution.

For the second experiment the pool which has the largest mining power is the pool where all miners converge to in the end. If miners are distributed equally among all 3 pools, all miners converge to the first pool. If two pools have the same mining power whereas the third one has a smaller mining power the outcomes are different. For the case when pool 1 has 14 miners, pool 2 has 2 miners and pool 3 has 14 miners, all miners converge to the first pool. For the case when pool 1 and pool 2 have 14 miners each, and pool 3 has 2 miners, all miners converge to the first pool. Notably, for the case when pool 1 has 2 miners, pool 2 and pool 3 has 14 miners each, all miners converge to the third pool. This pattern is illustrated in Figure 4.4. With increasing the amount of miners (e.g. 90) this pattern stays. If changing the initial mining power in each pool while keeping the same logic: two pools have the same mining power whereas the third one has the smaller mining power (e.g. 8:11:11), the above described pattern still holds.

An example simulation from the previous section was tested as well: 4 pools and 700 miners distributed equally between all pools with no free mining power. In the previous simulation miners were not able to switch pools and simulation did not seem to converge to a stable state for the described case. However, in this new simulation with the same settings, it does converge to a stable state where all miners end up mining honestly in one pool.

100 simulations with a random amount of pool miners ranging between 1 and 200, a random amount of pools ranging from 1 to 5, a random distribution of miners between all pools, and a random amount of solo miners ranging from 0 to 50 were set. All these simulations converged to a stable state where all miners end up mining honestly in one pool. If there exists any outside mining power, solo miners end up mining in this pool as well. It does not seem to be the case that a pool miner will leave their pool and start mining solo.

Similarly to the case of the replicated simulation, for the advanced simulation possible correlations between convergence time and the amount of pools / the amount of miners were plotted. Due to the computational complexity smaller amounts of pools ($n \leq 6$) or miners ($n \leq 500$) have been analyzed in the advanced simulation.

Convergence time was plotted against variable amount of miners 50 to 500 by increment of 50. Each experiment was repeated only once due to the fact that the same initial settings produce the same output. The results are shown in Figure 4.5: the convergence time equals 156 turns for 50 miners in the simulation and goes up to 3008 turns for 500 miners in the simulation. It is too time consuming to calculate the convergence time to reach a Nash equilibrium for larger amounts of miners in the simulation. This result suggests that convergence time increases non-linearly with the increasing amount of miners. Adding even more miners in the system may lead it being mpossibie to find a Nash equilibrium. Indeed, in a real Bitcoin network there are many more miners than just 500 and the network is not in a Nash equilibrium. Therefore, the performance of the simulation with dynamic pools may reflect the state of a real Bitcoin network, namely, incapability of pools in the network to reach Nash equilibrium due to computational complexity.

Unlike the amount of miners, the amount of pools does not seem to influence simulation convergence time. Convergence time was measured for variable amounts of pools (2 to 6) and 100 miners randomly distributed between all pools. For every number of pools simulation was repeated 10 times. The results are shown in Figure 4.6. There is an increase in convergence time when amount of pools in the simulation is changed from 2 to 3. However, further increase in amount of pools does not seem to influence convergence time. This was also shown by a non-parametric linear regression, Kendall-Theil regression, that returned insignificant result.

Limitations were added, as well, on the analysis of an influence of initial distribution of miners on

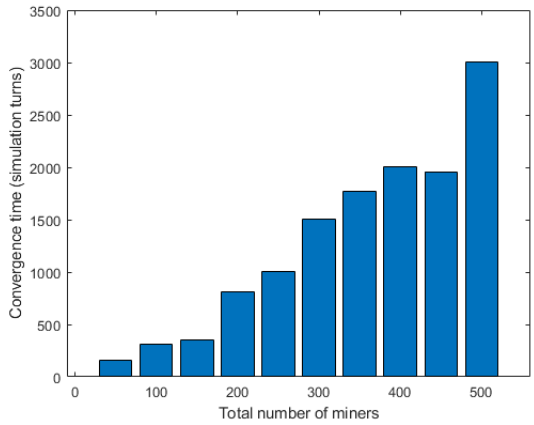

Figure 4.5: Convergence time versus variable amount of miners (50 to 500 with increment of 50) for the advanced simulation. All miners distributed almost equally among 4 pools.

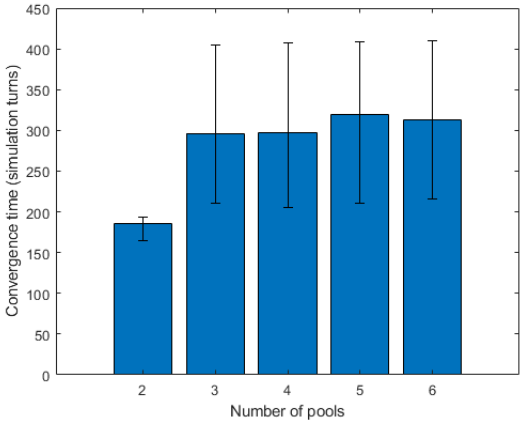

Figure 4.6: Convergence time in the advanced simulation with 100 miners distributed randomly between 2 to 6 pools

convergence time. In order to find a possible correlation between convergence time and the initial distribution of miners, the simulation with all possible distributions of 100 miners among 2 pools was run and corresponding convergence times were noted. Convergence times seem to fluctuate between 181 and 195 simulation turns without any clear pattern. For the extreme cases (when one pool initially contains no more than 4 members) convergence time is 103-104 simulation turns.

4.3 Price of Anarchy

Price of Anarchy (PoA) is a measure of the efficiency of the system that degrades with the selfish behavior of its agents. It is defined as a ratio between optimal social welfare and the worst-case social welfare in any Nash equilibrium (Koutsoupias and Papadimitriou, 1999). In their work Alkalay-Houlihan and Shash (2019) introduce the notion of PoA for the bitcoin simulation. This formula in a general case is shown in the Equation 4.2.

$$PoA = \frac{\sum_{i=1}^{p} m_i}{min_{(x_1^* \dots x_p^*) \in N} \sum_{i=1}^{p} m_i - \sum_{i=1}^{p} x_i} \quad (4.2)$$

where N is the set of all pure Nash equilibria and x_i^* is a set of attacking rates from pool i to all other pools.

It is socially desirable that as much mining power as possible is used to mine new block (i.e. mine honestly). This is explained by the fact that the Bitcoin system is more secure against attacks if more computational power is spent on honest mining (Zohar, 2017).

Equation 4.2 was used to analyze the efficiency of the advanced simulation. For the numerous different initial settings the advanced simulation converges to a Nash equilibrium where all miners mine honestly in one pool. In this equilibrium all miners mine honestly and thus the attacking rates are all zeros. This results in the value of PoA for the Nash equilibria that the advanced simulation converges to is 1. This means that there is no degradation in the system efficiency in comparison to the initial system state.

Eyal (2015) argued that the situation where a pool controls a majority of the mining power is not realistic for the Bitcoin system. Large pools hinder the distributed nature of Bitcoin system due to the

fact that they have a few centralized authorities with a lot of mining power in their hands. Such authorities are able to take control of the Bitcoin system by means of generating the longest chain and ignoring blocks generated by other miners. If this happens, the Bitcoin system becomes unstable (Eyal, 2015). Therefore, centralized mining is not a desirable outcome in the Bitcoin network since it endangers security of the network.

Hence, from this point of view the PoA of a Nash equilibrium where all miners mine in a single pool cannot be 1 since large pools degrades efficiency of the Bitcoin system. The definition of a PoA for the Bitcoin simulation should account for the variety and the size of the pools in the system. Equation 4.3 is a modification of the definition of PoA suggested by Alkalay-Houlihan and Shash that accounts for amount of pools in the system.

$$PoA = \frac{p_i}{p_n} \cdot \frac{\sum_{i=1}^{p} m_i}{min_{(x_1^* \dots x_p^*) \in N} \sum_{i=1}^{p} m_i - \sum_{i=1}^{p} x_i} \quad (4.3)$$

Here, p_i is a number of pools in the initial simulation settings and p_n is a number of pools in a converged state. In this way, an increase in amount of pools will result in smaller value of PoA (more efficient system) and reduction of the amount of pools will result in larger value of PoA (less efficient system).

The new definition of PoA does not change the analysis of the replicated simulation proposed by Alkalay-Houlihan and Shash (2019) in their paper since miners are not allowed to change pools and the number of pools is constant throughout the whole simulation, i.e. $\frac{p_i}{p_n} = 1$. Nevertheless, new formula for PoA makes difference in case of the advanced simulation. When all miners end up mining honestly in one pool the value of PoA will become equal to the initial amount of pools in the simulation (p_i). Therefore, Equation 4.3 reflect any decrease or increase in amount of pools in the system.

5 Conclusion

This paper explored a block withholding attack in the Bitcoin system based on the findings by Eyal (2015) and Alkalay-Houlihan and Shah (2019). It was shown by means of multi-agent simulation that even in the extended system with dynamic pools, i.e. the system where agents are free to switch between pools, pools still decide on sabotaging each other. This implies that even in the modified settings it is profitable for pools to perform a block withholding attack against some other pools since it is their best response in current settings, and at least large pools benefit from attacking in the long run.

This paper analyzes the correlation between convergence time and simulation variables as total amount of pools, total amount of miners, and initial miner's distribution among pools. This analysis faced several difficulties related to simulation complexity and amount of computational power needed to obtain results. Current findings suggest that in the case of the replicated simulation, convergence time may depend on amount of pools in the system, whereas in the case of the simulation with dynamic pools convergence time seems to correlate with amount of miners in the system. Convergence analysis faced difficulties related to computational complexity of finding Nash equilibrium. Interestingly, but this may be representative of a real state of the Bitcoin network: where a large amounts of miners in system pools are incapable of finding a Nash equilibrium and they just continuously switch their attacking rates against each other. This outcome suggests that iterative computation of agent's best response strategies might not be the feasible way to find Nash equilibrium in the Bitcoin network.

Lastly, the definition of Price of Anarchy for Bitcoin system suggested by Alkalay-Houlihan and Shah was modified in order to reflect the decentralized nature of Bitcoin. The suggested formula fits results of the simulation with dynamic pools better than the previous one as well as gives the same result as Alkalay-Houlihan's and Shah's formula for the replicated simulation.

Currently, in most of the cases, the simulations converge to a state where all miners mine honestly in a single pool. This happens partly due to the fact that the simulation is not capable of accounting for miner's preferences or uncertainties: all miners have the same preferences and can clearly find and calculate the best choice for their next step. It is not the case in the dynamic Bitcoin system as well as some agents may have predilection to one or another pool or types of mining. Simulating this aspect is interesting for closer-to-life simulation.

Additionally, the fact that pools and miners are capable of fully calculating the required information about the environment around them in the simulation is considered unrealistic. In contrast to the real life Bitcoin network where uncertainty is always present, leading to inaccurate representations of the world around them and inefficient choices. Due to this, the approach of simulating Bitcoin networks by means of a game with full information may not be representative of a real life situation. This leads to the idea that a better approach will be to simulate Bitcoin networks as a simultaneous game where only partial information is available.

One possible way to approach this is to introduce pool believes about other pools types. Pool type is determined by pool infiltration rates against other pools. These infiltration rates should not be directly observable by other pools, nevertheless, pools may have probability distributions corresponding to other pools types. These distributions, or believes, may be based on such factors as pool size or mining fees, history of attacks and revenue. In such settings, instead of fully observe other pools infiltration rates, pools can form and adjust their own beliefs regarding type of a different pool and their own best strategy.

References

C. Alkalay-Houlihan and N. Shah. The pure price of anarchy of pool block withholding attacks in bitcoin mining. In *AAAI 2019*, 2019.

J. Cochrane. Bitcoin and bubbles, 2017. URL `https://johnhcochrane.blogspot.com/2017/11/bitcoin-and-bubbles.html`.

T. Coven. Bitcoin is a bit of a miracle at any price, 2017. URL `https://www.bloomberg.com/view/articles/2017-12-11/bitcoin-is-a-bit-of-a-miracle-at-any-price`.

I. Eyal. The miner's dilemma. *2015 IEEE Symposium on Security and Privacy*, pages 89–103, 2015.

A. F. Haight. In *Handbook of the Poisson Distribution*. John Wiley & Sons, 1967.

E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. *Computer Science Review*, 3:65–69, 1999.

L. Luu, R. Saha, I. Parameshwaran, P. Saxena, and A. Hobor. On power splitting games in distributed computation: The case of bitcoin pooled mining. *2015 IEEE 28th Computer Security Foundations Symposium*, pages 397–411, 2015a.

L. Luu, J. Teutsch, R. Kulkarni, and P. Saxena. Demystifying incentives in the consensus computer. *IACR Cryptology ePrint Archive*, 2015: 702, 2015b.

S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. URL `http://www.bitcoin.org/bitcoin.pdf`.

A. Narayanan, J. Bonneau, E.W. Felten, A.K. Miller, and S. Goldfeder. In *Bitcoin and Cryptocurrency Technologies - A Comprehensive Introduction*, chapter 5. Princeton University Press, 2016.

Blog post. Block withholding attack on eligius mining pool, 2014. `https://bitcointalk.org/?topic=441465.msg7282674`, visited 2019-05-20.

M. Rosenfeld. Analysis of bitcoin pooled mining reward systems. *CoRR*, abs/1112.4980, 2011.

E. Swanson. Bitcoin mining calculator, 2013. URL `https://alloscomp.com/`.

A. Zohar. Securing and scaling cryptocurrencies. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, volume 17, pages 5161–5165, 2017.