On the impact of propogation delay on mining rewards in Bitcoin

Xuan Wen[1]

Abstract

Bitcoin[2] is a decentralized digital currency that is rapidly gaining in popularity. The Bitcoin system relies on "miners" whose job is to maintain a transaction log, and "mine" new Bitcoins. New transactions are incorporated into the transaction log when a "block" of transactions is created and added to the "blockchain", which is the data structure that stores the history of all transactions. It is in exchange for doing this work of creating a block and extending the blockchain that the miners receive rewards. It is generally assumed that Bitcoin miners receive rewards in proportion to their computation power, but this may not be the case in the presence of excessive propagation delays in the peer to peer network of miners. This paper presents the results of a simulation of Bitcoin that models the block propagation process in order to explore the impact of prorogation delays on mining rewards. The experiments show that when propagation in the network is very slow, miners tend to earn an unfair share even when they play honestly. We also discuss a miner strategy proposed by Ittay and Sirer[3] in which miners selectively withhold and propagate block messages, in order to increase their share of the rewards.

Introduction

As the first and most widely-used decentralized digital currency in the world, Bitcoin has not only provided a new way to exchange commodities, but has also given birth to a

---

1 This is joint work with Kira Goldner, Anna Karlin, and John Zahorjan

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

3 Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. arXiv preprint arXiv:1311.0243v5 (2013)

new industry-- the Bitcoin mining industry. The decentralized system shares all transaction history among a peer-to-peer network. Each transaction is verified in a public log in order for the transaction to be finalized. This public log, called the blockchain, is composed of individual blocks in a directed tree structure, and each block contains encrypted information of a certain number of transactions and information of the ancestor block. In such a way, each client is able to track any transaction ever made in the history as any transaction is contained in some block among the blockchain.

As an incentive for people to do the encryption work of blockchain, each person who mines a block adopted in the final blockchain receives a mining reward. As miners hear of transactions in the network, they try to include these transactions in the new block they are mining on by solving a cryptography puzzle that involves a hash of the ancestor block, hash of the transactions to be included in the current block, and the Bitcoin address of the miner as an address for the mining reward. The rate at which they solve the puzzle is proportional to their mining power, which depends on their hardware. When a miner successfully solves a puzzle and mines a new block, he propagates the block creation

message to the network, so that other miners know that the tree has been extended and so that they can extend on the new block. Only when a block has been part of the longest blockchain can the miner of this block receive the reward.

In rare cases, a block chain fork will form. When a miner has extended the tree with a new block A, the miner is required to immediately propagate the new block information to the rest of the nodes in the network, who in turn are required to pass on the information. But miners who haven't heard of the newly mined block A continue to mine on the old block B, the block before extension. There is a possibility that part of the ignorant network creates a block C as an extension of the block B, and propagates block C to the network so that the part of the network who hasn't heard of block A will adopt C as the extension of block B. As a result, block B will have two children, A and C, resulting in a bifurcation of the tree. The protocol specifies that the longest chain will be adopted. When there is more than one possibility for the longest chain, miners are required to mine on the first block they heard about. Thus, whichever fork extended earlier in a bifurcation has a larger chance of being adopted by the network.

A common assumption in the Bitcoin literature is that, as long as all miners follow the protocol honestly, each miner will earn a share of the rewards that is proportional to their mining power. That is to say, a mining pool, which owns 10% percent of the total mining power in the network, should receive approximately 10% of the rewards, while a mining pool of 40% percent of the mining power will receive approximately 40% percent of the rewards.

The purpose of this paper is to explore via simulation whether or not this assumption is correct. Specifically, we wish to determine to what extent the assumption that miners are rewarded in proportion to their computing power depends on propagation delays in the network, and specifically the relationship between propagation delay from/to a mining pool versus that mining pool's computing power. We will see that even with completely honest behavior the assumption of proportionate rewards is not always valid. In second Section, we will also discuss a further motivation for understanding the answer so this question, having to do with the behavior of Bitcoin in the presence of dishonest behavior.

Experiment Design and Purpose

I designed two simplified models that simulate Bitcoin mining and blockchain propagation process.

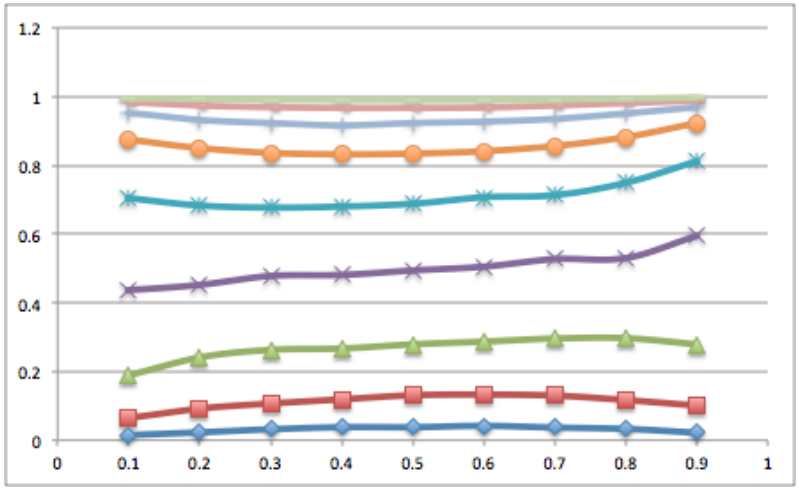
The first program

The first program focuses on the influence of propagation delay with varying computing power. Specifically, I want to show that whether withholding information is beneficial for miners of relatively high computing power, and whether it affects miners of relatively low computing power as well. As a result, we can speculate the best strategy for a miner given a certain computing power, and see whether following the protocal is the best strategy for most mining pools.

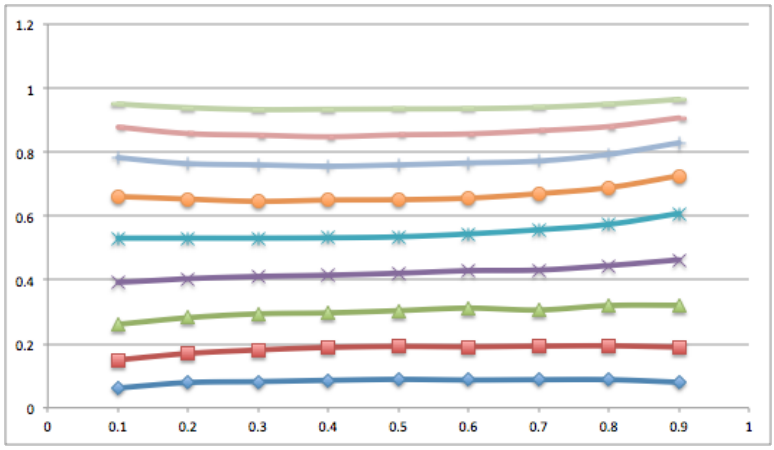
In each experiment, I will focus on the fraction of revenue of the first miner, so miners except the first miner will have equal computing power and equal propagation power. I will specify a certain number n to represent n miners or mining pools. I will assign the first miner a certain computing power ranging from 0.1 to 0.9, and the sum of computing power of all miners is always 1. I will represent propagation delay as

propagation power, which determines the chance by which a propagation event happens. The more propagation power a miner has, the faster is his speed of block message propagation. I will specify the sum of propagation power and assign a propagation power ranging from 0.1 to 0.9 so that the first miner have propagation power . The rest of the computing power and propagating power are assigned evenly among the rest of the miners.

In the following two graphs of 3 miners, the sum of propagation power is 1 and 10 respectively. That is to say, the propagation power is very low for all miners.



2 miners
sum of propagation power = 1
in 100000 steps
x axis: the propagation rate of the first miner from 0.1 to 0.9
y axis: the rate of revenue from 0.1 to 0.9
Each line represenst the revenue of the first miner given computing rate.

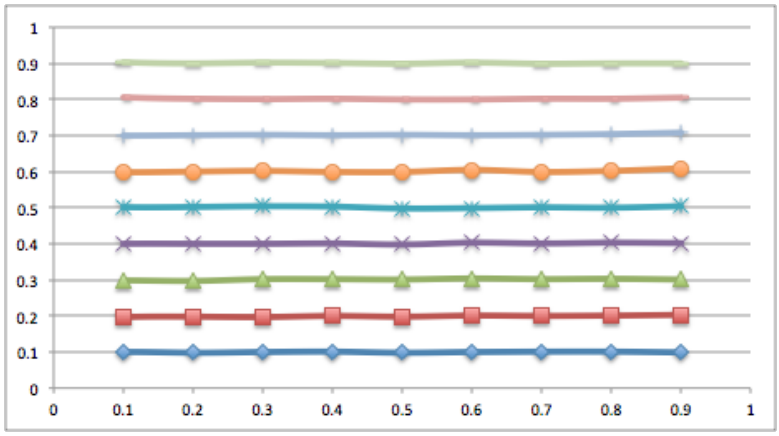


3 miners
sum of propagation power = 10
in 100000 steps
x axis: the propagation rate of the first miner from 0.1 to 0.9
y axis: the rate of revenue from 0.1 to 0.9
Each line represenst the revenue of the first miner given computing rate.

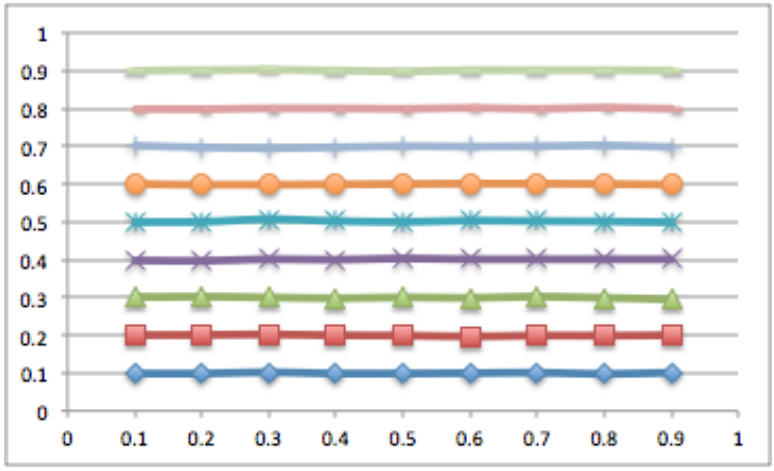
In both graphs, when the first miner has very low computing power, the other miners have a much larger chance of producing a new block. So with very low computing rate, the first miner always receives less than its fair share, since even if the first miner has produced a block, it is too slow to have been propagated before the other miner has mined another one. However, when the first miner has a relatively higher computing power, its fraction of revenue varies more with its propagation power. When the first miner has a very high computing power, the graph shows that its highest fraction of revenue is when its propagation power is very low or very high. That is to say, the best strategy for a miner of very high computing power is either to withhold its block mining messages or to have a very high propagation rate so that other miners' block messages are withheld. This is because, if the first miner has a very high computing rate as high as

0.8 or 0.9, it will be the one who extends the tree in 80% to 90% percent of the time. If it withhold such information, other miners would not be able to extend on the new block and they would continue to mine on the old block, and since the first miner has a higher chance of further extending its unpublished branch and always get the mining reward of the longest branch even before the other miners extend the old block, the first miner would almost receive all mining reward. The first miner seems to earn its fair share when its propagation rate is neither too high or too low, since if he allows other miners to hear of his blocks while allowing himself to hear and extend on other miners' blocks, other miners will have an equal chance of mining blocks, and therefore the reward is approximately fair. In contrast, if other miner's block cannot be propagated, no other miners will extend on his new block, and therefore the first miner with high computing power will still be the one who extends the longest branch.

This fluctuation is less obvious as the sum of propagation grows. When propagation events happens at a much greater chance than that of mining events, miners tend to earn their fair share, since all information is propagated in time.



3 miners
sum of propagation power = 1000
in 100000 steps
x axis: the propagation rate of the first miner from 0.1 to 0.9
y axis: the rate of revenue from 0.1 to 0.9
Each line represenst the revenue of the first miner given computing rate.



3 miners
sum of propagation power = 10000
in 100000 steps
x axis: the propagation rate of the first miner from 0.1 to 0.9
y axis: the rate of revenue from 0.1 to 0.9
Each line represenst the revenue of the first miner given computing rate.

In the two graphs above, the sum of propagation power is 1000 and 10000 times the computing power respectively. When propagation events happen at a much higher chance than block mining events, almost all new blocks have a chance of being propagated in time, so that almost all miners should be extending on the longest branch.

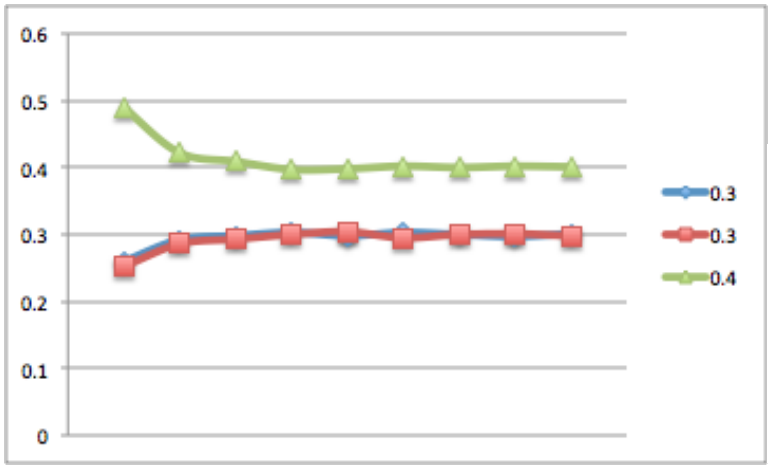
These experiments show that when propagation is generally slow in the network, miners tend to earn unfair share. When total propagation power is low, miners with high computing power can earn more than their fair share by either withholding their block or

withholding other miners' blocks from propagating. In this case, the current bitcoin protocol is not incentive-compatible since for those miners there exists strategies that benefit them more than following the protocol. However, in reality, propagation event happens at a much higher chance than that of mining event as long as the miners as nodes in the network are not isolated, and the cases in which the sum of propagation rate is 1 or 10 is not realistic. Therefore I believe when miners are honest and there is no Sybil attack (where some mining pools are isolated by Sybil nodes which do not propagate information), miners should almost receive the mining reward proportional to their computing power.

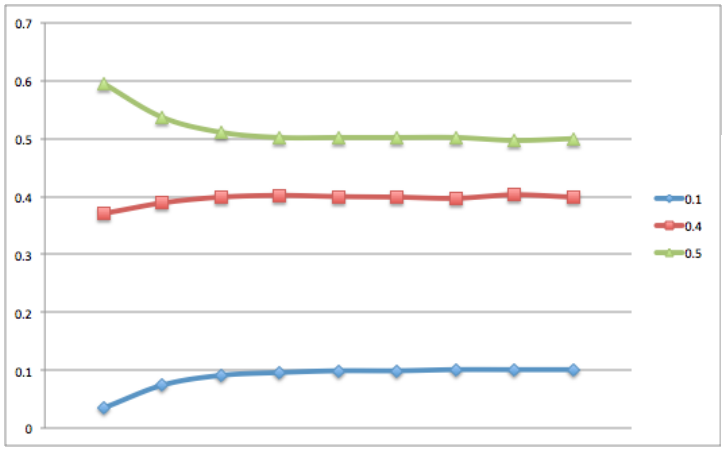
The second program

In the second program, I wanted to see a threshold propagation value such that miners should earn their fair share as long as the sum of propagation power is larger than this value. In other words, given a set of computing rate for each miner, what is the least sum of propagation power the network needs to have in order for the miners to earn their fair share?

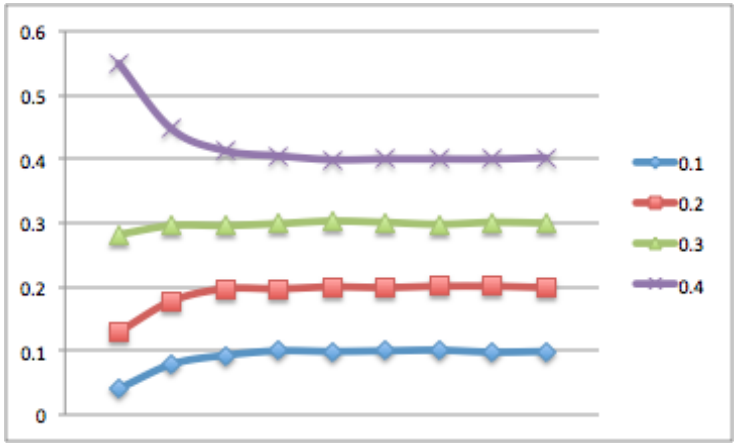
In this program, a group of miners are represented by a set of vectors with the sum =1. Each vector represents one miner's computing power. Their propagating power is assigned evenly among all miners regardless of their computing power, so propagation delay only depends on the sum of propagating power. The sum of propagation ranges from 1,5, 25,125,625,3125,15625 in each round. The curve of each miner's percentage of revenue is recorded which indicates their revenue with regard to the sum of propagation power.



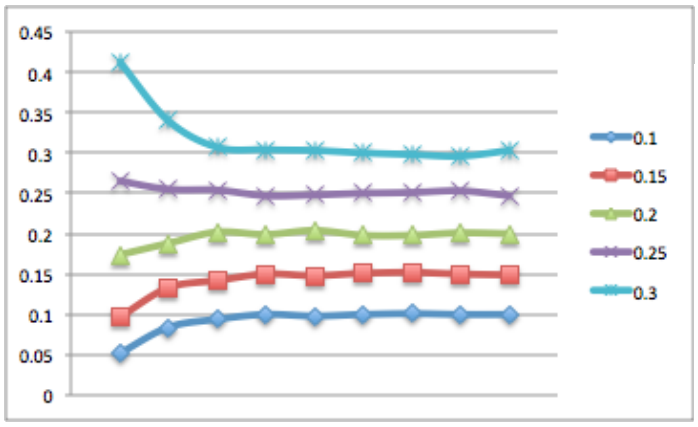
3 miners
a computing power: 0.3
b computing power: 0.3
c computing power: 0.4
x axis: the sum of propagation power as power of 5 1,5,25,125,625,3125...
y axis: the percentage of revenue
Each line represenst the revenue of the one miner



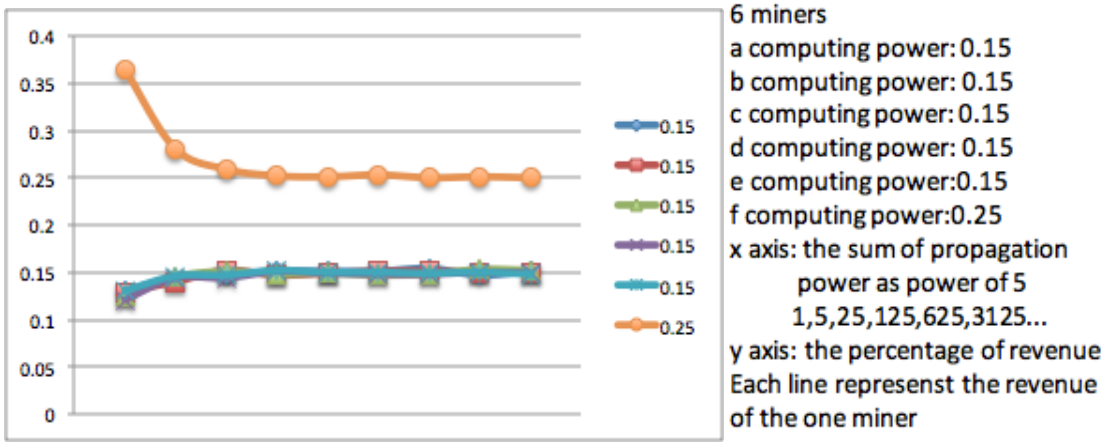
3 miners
a computing power: 0.1
b computing power: 0.4
c computing power: 0.5
x axis: the sum of propagation power as power of 5 1,5,25,125,625,3125...
y axis: the percentage of revenue
Each line represenst the revenue of the one miner



4 miners
a computing power: 0.1
b computing power: 0.2
c computing power: 0.3
d computing power: 0.4
x axis: the sum of propagation power as power of 5 1,5,25,125,625,3125...
y axis: the percentage of revenue
Each line represenst the revenue of the one miner



5 miners
a computing power: 0.1
b computing power: 0.15
c computing power: 0.2
d computing power: 0.25
e computing power:0.3
x axis: the sum of propagation power as power of 5 1,5,25,125,625,3125...
y axis: the percentage of revenue
Each line represenst the revenue of the one miner



We observe that when the sum of propagation is lower than 125, which is impossible in real world as propagation happens at a much faster rate than that of mining event, the miner with the highest computing power always earns more than his fair share, and the rest of the miners earn less than their fair share as a result. Another pattern we can conclude from the graphs is that, the miner with the highest computing power always benefits from propagation delay even if his computing power is not very high. The explanation of the gap between the strongest miner's real reward and his fair reward when the sum of propagation power is low lies in two parts: 1) when another miner first extends the block, other miners in the network fail to hear of this block mining event because of the low propagation speed. As everyone continues to mine on the old block, the strongest miner has the highest possibility to extend the old block or even further

extend to a length-2 branch. In this case, the strongest miner and the other miner has the same chance of being heard, but since the strongest miner has a greater chance of mining the longest branch, the miner who extends first would have wasted his computation power. 2) when the strongest miner extends the tree first, other miners fail to extend on his block because of the low propagation speed. Since the strongest miner has mines faster, he has a greater chance to extend his private branch to a length of 2 before other miners extend the old block. No matter which miner's block is published first, the strongest miner always has a higher chance of extending to the longest branch and win the final reward. In both cases, other miners' computation is wasted. This pattern exists until propagation reaches 100 or higher. That is to say, in reality when all nodes in the network are properly connected, miners should earn their fair share.

From the two experiments we see that when there is huge propagation delay, miners of high computing power can use strategies to receive more reward and the protocol is not incentive-compatible in that case. But in reality propagation delay is negligible

comparing to the speed of mining, we conclude that miners should earn their fair share as long as they are honest.

However, some literatures have pointed out cases in which miners could have earned more or less than its fair share. In *Majority is not Enough: Bitcoin Mining is Vulnerable*, Ittay and Sirer have pointed out that the current Bitcoin protocol is not as incentive-compatible as people assume. That is to say, for a rational mining pool, they could have some strategies different from what the protocol requires that will help them earn more than what they can earn if they follow the protocol. They suggested a strategy of selfish mining by which mining pools could earn more than their fair share as they can keep a private chain of blocks and withhold block creation information as a way to waste other people's time and energy spent in computation.

Mining Strategy that manipulates propagation

In *Majority is not Enough: Bitcoin Mining is Vulnerable*, Ittay and Sirer suggested a strategy called Selfish-Mine Strategy and proved both mathematically and statistically

that it could be adopted by a dishonest mining pool as a way to cheat, especially if the selfish mining pool has a relatively high computing power.

The selfish pool, composed of multiple miners scattered in the network, keeps a private blockchain unrevealed as opposed to the public blockchain publicized in the network. When the length of private chain is 0, the pool acts as an honest pool, as it will switch to the longer chain when a new block is publicized. When the length of private chain is not 0, the Selfish-Mining Strategy specifies actions:

1. Upon hearing block created by other pools
    a. if the private chain is one block longer than the public chain before other's block is mined:
        - the selfish pool will publicize it soon afterward. By setting up sybil nodes around other pools, the selfish pool is able to isolate other pool and decrease their speed of propagation or even isolate other nodes. In this case, when the selfish pool publish its own block as soon as it hears of another block, a Sybil attack on other pools

could have delayed the propagation of other's block and therefore wasted their computation.

b. if the private chain is two block longer than the public chain before other's block's extension:
    - publish whole private chain and wins because the private chain are now one block longer than what has been published, because by protocol the blockchain will be extended to the longer fork.

c. if the private chain is three or more blocks longer than the public chain:
    - publish the first unpublished block. As the selfish pool has at least two more unpublished blocks, it withholds block propagation until others publish further blocks so that it wastes other pool's computation.

2. Upon mining a block

. if the private chain has the same length as the public chain and the length of the private chain is now 2 after mining a new block:

- publish all blocks and wins by 1 block

a. otherwise:

- publish nothing and keep mining on the longest branch

The paper analyzes the system as a state machine. In the simulation, states indicate the difference between the selfish pool's private chain and the public chain. State 0 represents a no-branching state, and state transitions represent the happenings of mining events. For a selfish pool of a mining power of , mining events of the selfish pool or the other pools happen at exponential intervals with an approximate frequency of and 1- respectively. By an analysis on the state transition probabilities, the probability distribution gives equations that yield the following results:

$$p_0 = \frac{\alpha - 2\alpha^2}{\alpha(2\alpha^3 - 4\alpha^2 + 1)}$$
$$p_{0'} = \frac{(1-\alpha)(\alpha - 2\alpha^2)}{1 - 4\alpha^2 + 2\alpha^3}$$
$$p_1 = \frac{\alpha - 2\alpha^2}{2\alpha^3 - 4\alpha^2 + 1}$$
$$\forall k \geq 2 : p_k = \left(\frac{\alpha}{1-\alpha}\right)^{k-1} \frac{\alpha - 2\alpha^2}{2\alpha^3 - 4\alpha^2 + 1}$$

The paper uses to denote the fraction of miners who chooses to mine on the blocks created by the selfish pool when a bifurcation happens. It then further categorizes cases

of events and calculates expected revenue in each event type to see whether the selfish pool wins more than its fair share. The papers gives 8 cases (refer to paper, references or graphs here) and describes the revenue in the following expressions

$$r_{\text{others}} = \overbrace{p_{0'} \cdot \gamma(1-\alpha) \cdot 1}^{\text{Case (c)}} + \overbrace{p_{0'} \cdot (1-\gamma)(1-\alpha) \cdot 2}^{\text{Case (d)}} + \overbrace{p_0 \cdot (1-\alpha) \cdot 1}^{\text{Case (e)}}$$

$$r_{\text{pool}} = \overbrace{p_{0'} \cdot \alpha \cdot 2}^{\text{Case (b)}} + \overbrace{p_{0'} \cdot \gamma(1-\alpha) \cdot 1}^{\text{Case (c)}} + \overbrace{p_2 \cdot (1-\alpha) \cdot 2}^{\text{Case (g)}} + \overbrace{P[i>2](1-\alpha) \cdot 1}^{\text{Case (h)}}$$

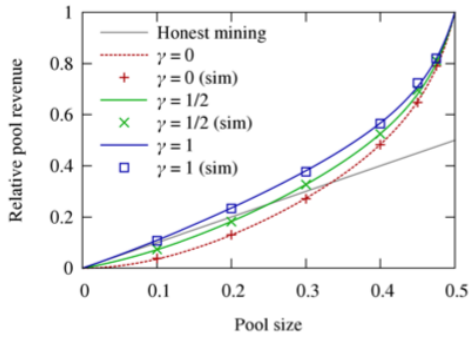
and calculates the fraction of revenue of the selfish pool in the following expression:

$$R_{\text{pool}} = \frac{r_{\text{pool}}}{r_{\text{pool}} + r_{\text{others}}} = \cdots = \frac{\alpha(1-\alpha)^2(4\alpha + \gamma(1-2\alpha)) - \alpha^3}{1-\alpha(1+(2-\alpha)\alpha)} .$$

so that a selfish pool of a computing power of will earn more than its fair share when

$$\frac{1-\gamma}{3-2\gamma} < \alpha < \frac{1}{2}$$

That is to say, a difference in the pool's propagation power is going to affect the pool's revenue as the graph produced:



In reality, the value of can be as high as 1, if the selfish pool is large enough to propagate information fast. Even if we assume its propagation power to be ½, it still wins more than its fair share as long as it has a computing power more than approximately 25%. The Selfish-Mining strategy shows that by selectively reveal and withhold block information, a pool is able to win more than its fair share.

References

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

2. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. arXiv preprint arXiv:1311.0243v5 (2013)