The Impact of Transaction Fees on Bitcoin Mining Strategies

Miles Carlsten

Master's Thesis

Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Master of Science in Engineering

Recommended for Acceptance
by the Department of Computer Science
Princeton University

Adviser: Arvind Narayanan

June 2016

Abstract

There are two incentives for Bitcoin miners to mine Bitcoin blocks. The miners are rewarded both from the minting reward of creating a new block and from the transaction fees on the transactions included in the block. Traditionally, the minting reward has always been significantly larger than the transaction fees, but as time goes on the minting fee is designed to diminish, and it is expected that the transaction fees will take its place. We look into the unverified claim that once incentives shift to mostly transaction fees, the security of the system will be the same as it was when it was based on minting rewards. We show that this change can lead to very uneven blocks, which can give rational miners an incentive to intentionally try to fork the blockchain. Additionally, we reexamine a previously known alternate mining strategy, known as Selfish Mining, and show that in a transaction fee based environment, it will become profitable for any miner (regardless of their hash power) to utilize a modified version of this strategy.

Acknowledgements

This research was performed at Princeton University under the oversight of Arvind Narayanan. Matt Weinberg has also contributed to the theoretical understanding presented in this work. We are hoping to publish this material in the near future.

Contents

List of Figures

Chapter 1

Introduction

Bitcoin is an electronic peer-to-peer cryptographic currency system [1]. The system relies on a public ledger, known as the blockchain, containing a list of every transaction that has ever occurred in the Bitcoin system. Sets of transactions are added to the blockchain in groups known as blocks. Most users don't participate in adding these blocks to the blockchain, but the ones who do are referred to as miners and the process of adding the blocks to the blockchain is referred to as mining [4]. The miners are all constantly trying to solve a cryptographic hash puzzle in order to be allowed to add the next Bitcoin block to the blockchain. The difficulty of this hash puzzle is what brings security to the system. If a user posts an illegitimate block to the blockchain, the other users can choose to ignore the block (this is known as forking the chain) and continue the chain off of the parent block. If the majority of the miners (weighted by their computational hash-power) side with the fork, then the original block becomes orphaned and is meaningless. As long as at least half of the users are acting with the default protocol (again, weighted by computational power) then an adversary who posts illegitimate blocks to the blockchain will eventually be beaten out by the rest of the network. Forking does not only happen with illegitimate blocks. Network latency is a real concern for Bitcoin miners, and frequently, there are forks in the blockchain

due to multiple miners finding perfectly legitimate blocks and posting their blocks before they hear about the other block [3].

The miners are incentivized through Bitcoin rewards to act according to the default mining protocol (which will be clearly defined in chapter 2). The rewards come from two sources:

- Minting rewards
- Transaction fees

The minting reward is a Bitcoin incentive that a miner can claim just by adding a block to the blockchain. The transaction fee reward allows a miner to claim all of the fees attached to the transactions that they included in their block. Currently, the vast majority of a miner's rewards comes from the minting reward as opposed to the transaction fees. Bitcoin is designed, however, so that the minting reward diminishes over time. At specific intervals, the minting reward halves, and it is anticipated that the rewards miners can get from transaction fees will increase to compensate (from larger fees on each transaction, and/or more transactions and associated fees). There is an relatively unexamined belief that the incentives after the system shifts to transaction fees will lead to the same security the system has with a block based minting reward.

We find that a transaction fee dominated system does not offer the same incentives for miners as a minting reward. The key intuition is that immediately after a block is mined, the value of mining the next block is small because there are not enough unclaimed transaction fees remaining in the system. If the reward for mining a block changes over time, it can incentivize miners to behave deviantly when the default compliant behavior would have them mining a near valueless block. For example, miners could consider refusing to mine at all if a block is not valuable enough to mine (running the mining hardware has costs like electricity and cooling). If we assume miners have to mine all the time, it can lead to miners instead spending the time

when a block is small trying to reclaim the transactions in the current head of the blockchain by forking, encouraging other miners to support their side in the fork by leaving some of the reclaimed transactions fees to be taken by the next miner.

Chapter 2

Model, methods, and simulator

The model of the Bitcoin system that we analyze is after the minting reward has dropped to zero. We consider transaction fees to be the only revenue for miners, and we model the transaction fees arriving to the Bitcoin system at a constant rate. Additionally, we assume that the miners can claim all unclaimed transaction fees that they have heard of in their block (this would be equivalent to an infinite block size, but is similar to an environment where the average number of transactions per block is well under the allowed block size). We model the network having no latency (once a miner decides to publish a block, all other miners immediately gain knowledge of the block). Finally, for this work, the assume the Bitcoin miners constantly mine.

2.1 Formalized description of a mining strategy

We consider a variety of existing and new Bitcoin mining strategies. All of the mining strategies that we consider can be formalized into the same general structure. In each instant, every miner will need to make several distinct decisions about the block they are currently mining and each of our strategies are defined by exactly how miners make these decisions. The decisions a miner will make on an instantaneous basis are the following:

- Where they are mining
- How much of the available transaction fees they are including in their block
- Whether or not to publish any found blocks

The first decision they will make is where they are going to mine. As an example, consider the default compliant miner: they will choose to mine on the longest chain that they are aware of, and in the case of multiple blocks that are tied for the longest chain, they will favor mining on the older block in the fork. This decision forms the basis for how a mining strategy will determine which side of a fork it wants to support, or, alternatively, if the miner wants to create a new fork. The next decision a miner will make is to decide how many of the available transaction fees they should claim in their block. Again, as an example, consider the default compliant miner: they will include all of the available transaction fees that can be claimed in their block. The final decision is when to publish blocks. When a miner mines a block, only they are aware of its existence. At each moment, miners can choose whether or not to alert the other miners of the block that they have found. This allows for mining strategies where miners intentionally choose to not reveal their blocks (such as selfish mining). Fully writing out the default compliant mining strategy in this formalized definition would be:

- **Where**: Mine off the highest block. In the case of a tie, if one of the blocks is yours, mine on that. Otherwise, mine on the oldest block.
- **How much**: Include all available transaction fees.
- **Publish?**: Always and as soon as a block is found.

Realistically, in most of these strategies, it best to add another rule on where to mine: in the case of a fork, if one of the blocks is your own, you should always mine on that block. No one should rationally try to fork their own block.

2.2 Bitcoin mining strategy simulator

In order to fully analyze what the game theoretic landscape will look like once the Bitcoin mining incentive becomes transaction fee based instead of block reward based, we have developed a versatile Bitcoin mining strategy simulator. The simulator runs games, and a simulation of a strategy could involve running a single game, or many consecutive games where results in past games determine parameters in future games. The simulation is time driven, and each game is composed of rounds which represent a fixed amount of time in the game. For all of our analysis, the rounds have a length of one second. We have chosen one second as the basic time unit in our simulations because realistically in the Bitcoin ecosystem, anything happening on a timescale smaller than a second is not going to be interesting because it is smaller than the timescale for blocks and transactions being relayed across the Bitcoin peer-to-peer network and much smaller than the timescale on which blocks are found. This basic time unit does represent a limitation on the simulator, but it is configurable, and could be adjusted to observe time steps smaller than one second. We have chosen to use a time driven simulation as opposed to an event driven simulation for several reasons. The most important of which is that in an event driven simulation, it is hard to clearly define all of the possible events in the game. In particular, when a new strategy is added to the simulator, it almost certainly will require adding new things to the list of possible events. One of the goals of the simulator is to make it easy to introduce new strategies, which wouldn't be possible if new events need to be added to the list of possible events constantly. Another reason to use a time based simulation was the simplicity of the implementation. We found an event driven simulator to be much more difficult to implement.

In each game, every miner is using a specific strategy. During each round, they will make decisions based on the strategy they have selected to play with. These decisions are characterized by the formal definition of a strategy that we defined earlier in this

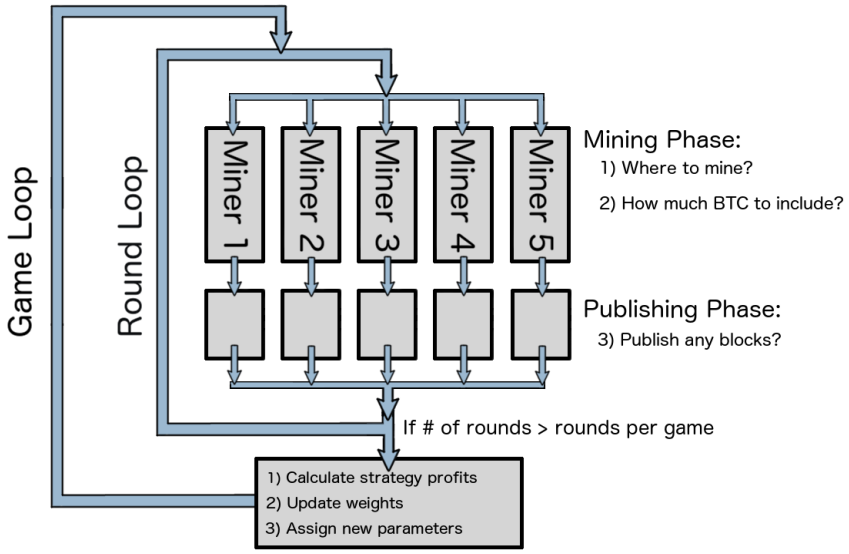


Figure 2.1: Diagram of the process inside the mining simulator. Each game consists of rounds, and in each round, miners make decisions on where to mine, how much of the available transaction fees to include in their block, and whether or not to publish any unpublished blocks they have. A miner's responses to these three decisions defines the strategy that they are using.

chapter. We separate out the first two decisions regarding where to mine and how much to mine from the final decision of what to publish into a mining phase and a publishing phase. Between these two phases in the simulator, a miner determines if they find a block.

For several of our simulations, we want miners to utilize the strategies that are doing the best. In order to accomplish this, we run several games, with hundreds of miners in each game. Miners chose strategies proportional to the weights of the strategies. We adjust the weights of the strategies from game to game, depending on how well the strategies are doing. After every game, and for every strategy, we take the profits earned by every miner using that strategy and average them, giving

us an average profit for every strategy being used. For each strategy k in game i, we calculate the weight to be the following:

$$w_k^i = w_k^{i-1}(\frac{1}{2})^{(1-\frac{profits_k^{i-1}}{maxProfits^{i-1}})} \tag{2.1}$$

where w_k^{i-1} was the k strategy's weight in game $i-1$, $profits_k^{i-1}$ are the average profits earned by the strategy k in game $i-1$, and $maxProfits^{i-1}$ is the maximal average profit of any strategy in game $i-1$. This is meant to follow the multiplicative weight update algorithm [2] with an ϵ value of $\frac{1}{2}$ to encourage the exploration of new strategies.

Chapter 3

Sequence of undercutting strategies

We define a transaction fee pool to be all of the transaction fees in the network that a block could potentially claim. Different mining locations for a block on the blockchain will naturally have different transaction fee pools because the chain being mined off of will have already claimed a different set of transactions.

3.1 Petty compliant miner

Consider the case where there is a fork: two blocks are tied for longest chain. The default compliant mining protocol would have the miner select the older of the two potential block heads and mine to support that one. However, the miner could potentially mine in either place, and if the miner is using a strategy that claims all of the transaction fees left in the transaction fee pool, it would be in the rational interest of the miner to not mine on the older block, but instead the block that leaves the largest transaction fee pool. We will use the term *wealth* to describe how many transaction fees a given block has claimed, with a large wealth indicating that a block has claimed many transaction fees. The less wealthy block will have left more transaction fees unclaimed in the transaction fee pool, which can be used by the next miner to create a more valuable block. A rational miner would want to mine

on the less wealthy block as opposed to the older block. We call this strategy petty compliant, and it's definition is the following based on the strategy formalization presented in chapter 2:

- **Where**: Mine off the highest block. In the case of a tie, if it exists, mine off your own block. Otherwise, mine on the block with the least wealth.
- **How much**: Include all available transaction fees.
- **Publish?**: Always and as soon as a block is found.

The petty compliant miner is strictly better than the default compliant miner. The two are identical except for the case where the miner is required to choose between two equal height blocks to mine on. In this case the petty compliant miner will always make the decision to mine in a location to maximize their rewards, and a default compliant miner might not. In our mining strategy simulator, we compare the default compliant strategy to the petty compliant strategy and do in fact see that the petty compliant miner outperforms the default compliant.

3.2 Forking

Although not very harmful itself, the existence of the petty compliant mining strategy paves the way for other more aggressive strategies because these strategies can encourage other miners to build on their side of a fork in a tie, incentivized via the extra transaction fees they will leave in the transaction fee pool for the next miner to claim. If the current head of the blockchain is wealthy, and the transaction fee pool it leaves is dry, then a miner could opt to try to fork the current head of the blockchain, and undercut their block with a less wealthy block, thus claiming a larger block for themselves than they could have gotten from default mining, and incentivizing the next miner to support their block because they leave a larger transaction fee pool for them to claim.

If miners begin undercutting blocks when the blocks contain more transactions than what's left in the transaction fee pool, then rational miners will stop claiming all of the transaction fees available in the pool and leave a buffer in order to reduce the chance that their block gets forked. We call miners who employ strategies that take some fraction of the available pool function forkers. We define these strategies so that when they are presented with a transaction fee pool of size x, they will claim $f(x)$ of that pool, where $f(x)$ is bounded between 0 and x for all x. The goal of these strategies will be to balance maximizing the wealth of the blocks they mine, and minimizing the chance that their block will be undercut by the other miners in the system.

If the current maximum public height of the blockchain is h, we restrict the function forking miners to only mine a block at height h or $h + 1$ (this is the height of the block mined – mining off of the tallest block, or forking 1 block back). We make this restriction because mining anywhere else will leave the miner in a position where they need an additional block mined on top of their first in order for it to be even tied for the longest chain. The function forking miner will look at all the potential blocks at height h that they could mine on top of, and they will select the least wealthy block at this height because it leaves the largest available transaction fee pool. Let this be known as the continuing pool. They will also consider all the potential blocks to mine on top of at height $h - 1$, selecting the least wealthy block that leaves the largest value available in the transaction fee pool. Let this be known as the forking pool. The function fork miner will then compute $f(x)$ for both of these pools. If the miner was to fork the chain, they would need to make sure that the block they fork with undercuts the current best head (least wealthy block) at height h. Let the wealth of this block be B_h.

$$Value_{continue} = f(pool_{continue}) \tag{3.1}$$

$$Value_{fork} = min(f(pool_{fork}), B_h) \tag{3.2}$$

If $Value_{continue} > Value_{fork}$, fork the chain by mining with access to $pool_{fork}$. Otherwise continue and mine with access to $pool_{continue}$. This gives the following formalized definition for the function forking mining strategy:

- **Where**: If you have a block at height h, mine off that block. Otherwise, consider forking or continuing. If $Value_{continue} > Value_{fork}$, continue the chain in the location of the $pool_{continue}$. Otherwise fork in the location of $pool_{fork}$.
- **How much**: Include $Value_{continue}$ if continuing, or $Value_{fork}$ if forking.
- **Publish?**: Always and as soon as a block is found.

The function fork miner will mine in the location that allows them to mine the most valuable block. If we make the assumption that any reasonable choice for $f(x)$ is monotonically increasing, then $f(pool_{fork})$ is always going to be larger than $f(pool_{continue})$, so the decision to fork will come down to a comparison of B_h and $f(pool_{continue})$.

We have most closely examined the linear coefficient family of function fork miners. In these strategies, we consider

$$f(x) = kx \tag{3.3}$$

for some k between 0 and 1. If we take a group of these strategies, and run them in the simulator with multiplicative weights discussed in the previous chapter, what we find is that it is always best to be slightly more aggressive at forking than the majority of the other miners in the system.

Strategies that take less and less wealthy blocks and are more aggressive when it comes to forking other miners become more successful. Because miners will always mine on their own block if it is a competitor in a fork, we see that if the current

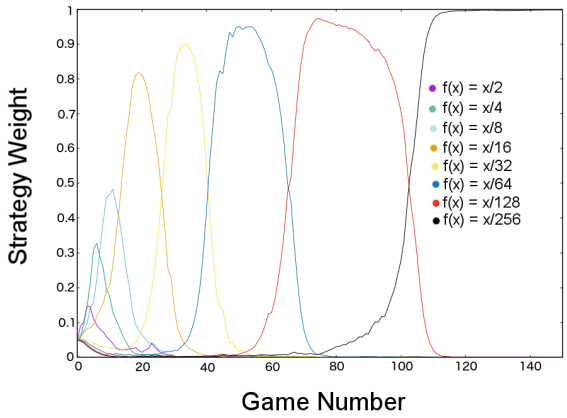


Figure 3.1: Normalized weights of different linear coefficient function forking strategies over a series of games. Strategies that are slightly more aggressive than the most common strategy perform the best and have their normalized weights increase. This trend continues until we get to $f(x) = \frac{x}{256}$ which is the smallest function forking strategy run in this simulation.

dominant coefficient gets small enough, the larger coefficient function forking strategies start to do well again. What is happening here is the transaction fee pool grows so large that if one of the strategies with a large coefficient can get lucky and mine two blocks in a row (making it so none of the other miners, who are limited in their ability to fork more than one block back, are able to fork), the block that they mine is so large that even if this happens only once in a game, they out perform the other strategies. Once the larger coefficient strategies start to gain weight, we see the same sequence of slightly more aggressive strategies doing better and better again, until the next reset when the coefficient gets small enough.

In order to find an equilibrium, we consider all function fork miners mining with the same function $f(x)$. When the function fork miner mines a block, they are interested in maximizing the value of the blocks times the chance that the block makes it into the final chain and isn't undercut. Consider the expected profit for a miner of a block. Let this be R, given by

$$R = B\rho \tag{3.4}$$

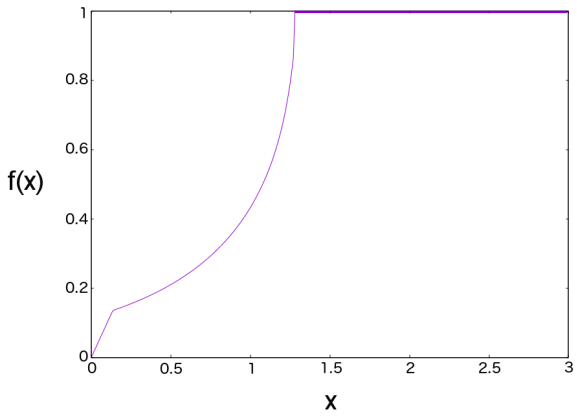


Figure 3.2: The function $f(x)$, described by equations 3.5-3.7, that leads to an equilibrium of function fork miners. The expected block size has been normalized to 1.

where B is the value of the block mined, and ρ is the chance that that block makes it into the final longest chain. If we make the assumption that the $f(x)$ leading to an equilibrium is monotonically increasing, then the other miners will try to fork a block of value B until $f(pool_{continue}) > B$. The chance that the block of value B is forked is equal to the chance that another block is found in the time period where $f(pool_{continue}) \leq B$. By maximizing equation 3.4 with regards to $f(x)$, we find that there exists an equilibrium where $f(x)$ is the following piecewise function, with the expected block size normalized to 1:

$$f(x) = x \quad \forall\ x <= c \tag{3.5}$$

$$f(x) = -W_0(-ce^{x-2c}) \quad \forall\ c < x < 2c - \log(c) - 1 \tag{3.6}$$

$$f(x) = 1 \quad \forall\ x \geq 2c - \log(c) - 1 \tag{3.7}$$

where W_0 is the zero branch of the Lambert function, and c is some constant such that $2c - \log(c) - 1 \geq 1$.

When run in our simulator, we verify that this is, in fact, an equilibrium. However, this equilibrium has a caveat: it is only in equilibrium if the function fork miners decide to fork blocks in the case of a tie instead of continuing. If the miners decide to continue in the case of a tie, we have not been able to find an equilibrium.

Chapter 4

Selfish mining with transaction fees

Selfish mining is an already known alternate mining strategy detailed in [5]. With our mining strategy simulator, we wanted to investigate how this strategy might change in an environment with transaction fees being much more important than the block reward. We consider this strategy operating in the model that we've been working with, where the transaction fees accumulate in the network at a constant rate and the block reward is 0.

4.1 Selfish mining in a transaction fee system

The goal of a miner employing the selfish mining strategy is to essentially trick the other miners in the Bitcoin network to mine on top of a block that is guaranteed to not make it into the final longest chain. By having other miners not mine in a useful position, the selfish miner is capable of exaggerating their own portion of the overall network hash-rate. Selfish miners do this by trying to create a chain in private that only they know about. When the selfish miner initially finds a block, they will not announce their block to the rest of the network. They will continue to mine on their private block, hoping to find a second block before the rest of the network finds a block (leading to two cases– one in which the rest of the network finds the next

block, and the other in which the selfish miner finds the next block). In the first case, if the rest of the network finds a block, the selfish miner will immediately publish their private block to establish a race between their block and the block just found to see which will end up being selected to be in the longest chain. In the second case, where the selfish miner finds the next block, the selfish miner will also keep this block private. At this point, however, the selfish miner has a private chain that is long enough that after the rest of the network finds a block, the selfish miner can immediately publish their entire private chain and have that instantly be accepted as the longest chain. The selfish miner will continue to mine on their own private chain, trying to increase its length, while the other miners will mine on what they believe to be the longest chain– in a place where any block they find will not make it into the longest chain. As soon as the rest of the network finds enough blocks that the selfish miner has a lead of only 1 block, the selfish miner will immediately publish, securing all of their private blocks into the final block chain, and ensuring that all blocks in the opposing fork are orphaned.

Assuming the selfish miner has less than half of the overall hash power of the network, they will eventually need to publish their private chain. We assume in our model that the selfish strategy will include all claimable transaction fees in every block that they mine. We model the system with one selfish miner, and we define α to be the fraction of the overall hash-rate possessed by the selfish miner, and γ to be the fraction of the mining network that will support the selfish miner's block in the case of a tie and race condition. If we consider the longest public chain to have length h, this gives the following according to our formal definition of a strategy:

- **Where**: Mine off of the last block you mined, unless that block is less than height h. Then mine on the highest block, and in the case of a tie, the oldest block.
- **How much**: Include all available transaction fees.

- **Publish?**: If there exists a private block at height h, publish it. If there exists a public block at height h that is not your own, and you have a public block at height h, and you have a private block at height $h+1$, but not at $h+2$, publish the block at height $h+1$.

Let the fraction of the total profits that the selfish miner earns be denoted as $R_{selfish}$. The expected $R_{selfish}$ can be expressed as the following:

$$R_{selfish} = \sum p_i f_i \tag{4.1}$$

where p_i is the probability for the system to be in state i, and f_i is the probability that a transaction arriving to the system while it is in state i eventually winds up in a block mined by the selfish miner. The possible states can be defined as the lead the selfish miner has in blocks on the rest of the network. We will define state 0 to be the state in which there are no private blocks. Similarly, state 1 will be defined as the state where the selfish miners have mined a private chain whose length is 1 longer than that of the rest of the network. State 2 will be defined as the state where the selfish miners have mined a private chain whose length is 2 longer than that of the rest of the network, and so on. Additionally, we must define state $0'$ where the selfish miner has found a block, but has a lead of 0 (the case where the selfish miner finds a block, but then the rest of the network finds the next block and there is a fork with equal length sides). Let the probabilities of these states happening be defined as p_0, $p_{0'}$, p_1, p_2, ...etc. Following the analysis in [5], it can be shown that the values for the probabilities of being in each state of the system are the following:

$$p_0 = \frac{1-2\alpha}{2\alpha^3 - 4\alpha^2 + 1} \tag{4.2}$$

$$p_{0'} = \frac{(1-\alpha)(\alpha - 2\alpha^2)}{2\alpha^3 - 4\alpha^2 + 1} \tag{4.3}$$

$$p_i = (\frac{\alpha}{1-\alpha})^{i-1} \frac{\alpha - 2\alpha^2}{2\alpha^3 - 4\alpha^2 + 1} \tag{4.4}$$

In order to calculate the associated f_i's, consider the chance of a transaction arriving in each of the states ending up in a block mined by the selfish miner. Let us start by calculating f_0. In state 0, there are no hidden blocks, only the main chain. If a transaction arrives at this time, there are several possible outcomes. First, the rest of the network could mine the next block before the selfish miner. In this case, the transaction will be in this block, and this block will make it into the longest chain. This happens with probability $(1-\alpha)$. Alternatively, the selfish miner could find the next block. If the selfish miner finds the next block, they will include the transaction in their block, but they keep this block private after they find it. This block is not guaranteed to make it into the final chain, yet, and happens with probability α. Now there are two possible outcomes: the selfish miner finds the next block after that as well (another probability of α), or the rest of the network finds the next block (probability $(1-\alpha)$). If the selfish miner finds the next block, then it is guaranteed that their block holding the transaction we are considering makes it into the final chain. If the rest of the network finds the next block, then a race condition is triggered between the selfish miner's block with the transaction and a different block with the transaction. In this case, whoever wins the race will collect this transaction. The selfish miner will win the race if they find a block, or if the fraction of the network that will mine on the selfish miner's block in the case of a tie finds a block. This happens with a probability $\alpha + (1-\alpha)\gamma$. Thus, the chance that the selfish miner will eventually claim a transaction arriving when the system is in state 0 is given by

$$f_0 = \alpha^2 + \alpha(1-\alpha)(\alpha + (1-\alpha)\gamma) \tag{4.5}$$

Next, consider the chance of the selfish miner claiming a transaction arriving when the system is in state $0'$. In this state of the system, a race is currently happening between the selfish fork and the default fork. The next block mined (regardless if it is mined by the selfish miner, or the rest of the network) will contain the transaction that just arrived, and it is guaranteed to make it into the final longest chain. Thus, the chance that a transaction arriving to the system when it is in state $0'$ being claimed by the selfish miner is equal to the chance that the selfish miner finds the next block (which has probability α).

$$f_{0'} = \alpha \tag{4.6}$$

Now, consider the selfish miner claiming a transaction arriving to the system when it is in state 1. In state 1, the selfish miner has found a single block, which is hidden. If the selfish miner finds the next block, they will have a private chain of length 2, in which case both blocks are guaranteed to make it into the final block chain, hence securing the transaction in a block the selfish miner owns (this happens with probability α). Alternatively, the rest of the network could find the next block (with probability $(1 - \alpha)$). This triggers the race condition between the selfish miner's block and the default block that we have seen before, however, this time, the selfish miner's racing block does not contain the transaction we are considering, but the default racing block does. If the default block wins, the transaction is guaranteed to not be claimed by the selfish miner. Thus, the only way for the selfish miner to claim this transaction fee is to mine with tie winning block (which itself will include the transaction fee, and is guaranteed to make it into the final longest chain). This has a probability of α. This gives the chance for the selfish miner to claim a transaction arriving to the system in state 1 to be:

$$f_1 = \alpha + (1 - \alpha)\alpha \tag{4.7}$$

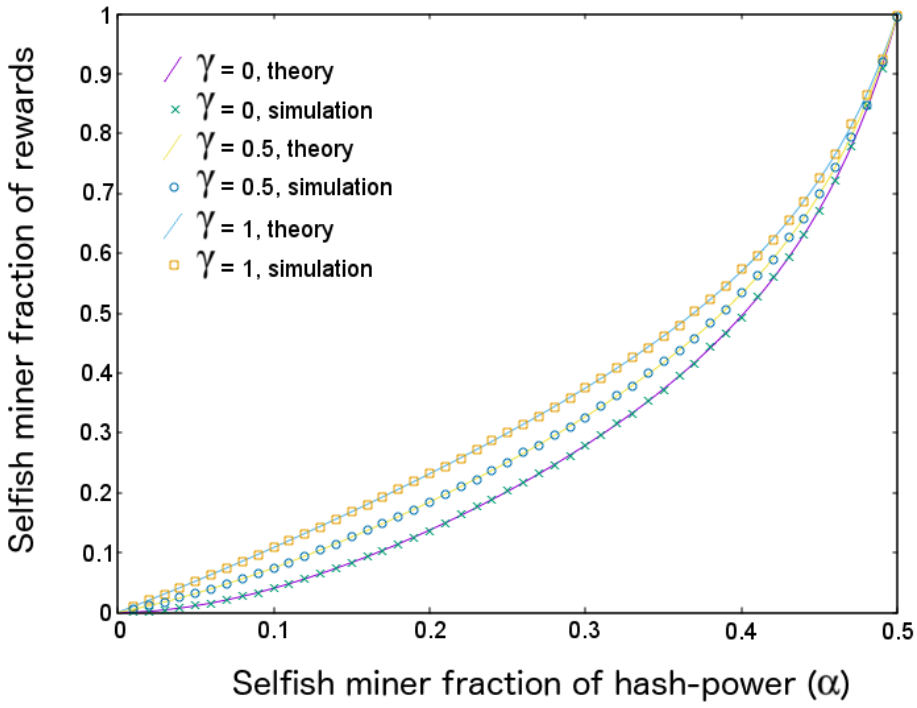


Figure 4.1: We see simulation matching the theory for selfish mining in a transaction based model for $\gamma = 0$, 0.5, and 1.

Finally, consider a transaction arriving to the system in any state i, $i > 1$. In these states, perhaps it is easier to consider what must happen for the transaction to **not** end up in a block the selfish miner owns. For the rest of the network to be able to claim the transaction, they will need to mine enough blocks to trigger the selfish miner to release their entire private chain (all blocks that they mine in this process will be discarded when the selfish miner reveals their chain). This has a probability of $(1-\alpha)^{i-1}$ because the default miners must mine $i-1$ blocks to cause the selfish miner to publish their private chain. This now brings us back to state 0. We have solved this state and know that the chance of the default miner being able to claim the transaction is $1 - f_0$. Considering that the transaction must either end up in a block that the selfish miner owns or not, the chance that the selfish miner claims a transaction arriving to the system in state i, $i > 1$ is given by:

$$f_i = 1 - ((1-\alpha)^{i-1}(1 - f_0)) \tag{4.8}$$

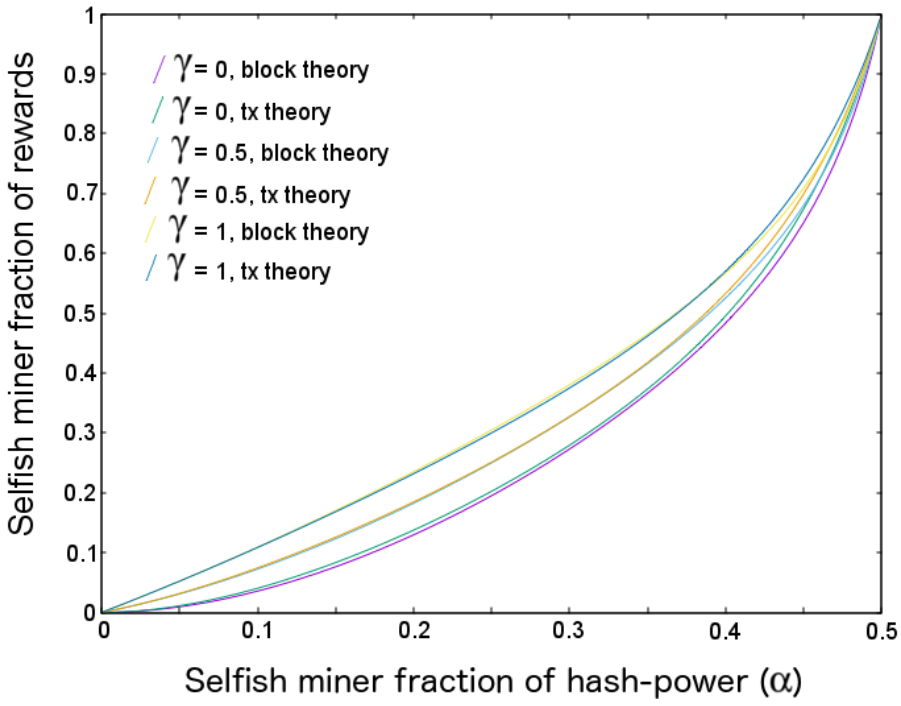


Figure 4.2: We show a comparison of the theory between selfish mining in a block based model and in a transaction based model, for $\gamma = 0$, 0.5, and 1. The theory ends up being very close.

Using all of these expressions for the chance that a transaction makes it into a selfish miner's block given a state of the system when the transaction arrives, and the associated probabilities of these states, we can utilize equation 4.1 to calculate the total fraction of the rewards that the selfish miner is expected to earn. We are also able to use our simulator running the selfish mining strategy on blocks that have a transaction based reward system in order to verify this theory.

In the model where the rewards are from transaction fees, we find that the selfish mining strategy performs very similarly to how it performs in the block reward based model, however, there is one critical and important difference. With traditional selfish mining, the selfish miner does not immediately make additional profits– in fact, on an immediate time scale, they make less. They claim a larger fraction of the blocks than they normally would have been able to claim, but the rate at which they claim blocks is equal to, or less than (due to the lost blocks), the rate at which they could claim

blocks as a default miner. The benefit for them is not that they find more blocks, it is that they cause many of the other blocks in the system to be wasted, and on the next difficulty readjustment, the difficulty threshold will drop accordingly. Only then can the selfish miner mine blocks faster, and earn more profits. The difficulty readjustment only happens every 2 weeks, so the selfish miner will need to selfishly mine at no additional profits for this period before the strategy starts to benefit them. An additional problem from this waiting period is that the selfish miner's fraction of the network hash-rate could easily change over the span of 2 weeks, especially if there is a drop in difficulty because it is quite plausible that miners who have retired equipment that cannot turn a profit at the current difficulty, but could at the lessened difficulty, turn back on these mining rigs. In a transaction fee based system, however, it is immediately profitable to selfishly mine. When the selfish mining strategy is working, many of the blocks mined by the network will become orphaned, and because of this, on average, the inter-arrival times between blocks in the final longest chain will be larger. Because the transaction fees come into the system at a constant rate, this means that these blocks will be larger on average than blocks found with the default inter-arrival time. The immediate responsiveness of the strategy makes it significantly more tantalizing for miners to utilize.

4.2 Implementing a block size cutoff

The transaction fee model also allows for an additional optimization: the selfish miner can institute a cutoff value. For blocks that have grown above this cutoff value the selfish miner will instead choose to use the default mining strategy to mine the block instead of selfishly mining the block. The intuition is that the selfish miner wants the rest of the network to mine in the wrong location, but in order to do this, they have to hide their blocks, and specifically, risk losing the first hidden block if it comes to a

race condition (the system described by state $0'$). If the blocks are not all the same size (as they are in the block reward model), miners can now decide that some blocks are too valuable to not publish and risk losing.

We modify the selfish mining strategy so that if the system is in state 0, the selfish miner will immediately publish any block they find like a default miner if the size factor of the block (denoted β) is above a certain threshold. We define the size factor to be the the size of the block compared to the size of a block found at the average block arrival time. For example, say the inter-arrival time of the blocks is 600 seconds. If the rate at which transaction fees accrue in the network is 25/600 BTC, then the average size of a block is 25 BTC. A size factor of $\beta = 0.5$ would indicate a 12.5 BTC block. Similarly, a size factor of $\beta = 2$ would be a 50 BTC block. This tweaks the formal definition of the selfish miner to the following (recall the height of the highest public block is h):

- **Where**: Mine off of the last block you mined, unless that block is less than height h. Then mine on the highest block, and in the case of a tie, the oldest block.
- **How much**: Include all available transaction fees.
- **Publish?**:If there exists a private block at height h, publish it. If there exists a public block at height h that is not your own, and you have a public block at height h, if you have a private block at height $h+1$, but not at $h+2$, publish the block at height $h+1$. Also, if you have a private block at height $h+1$, but not at height $h+2$, and the value of the block at $h+1$ is larger than β, publish it.

In order to repeat a similar analysis on this strategy to what we've done above, we choose to rework the possible states of the system. We add a state to the system, state $0''$. In this state, it is identical to state 0, except from this state, if the selfish miner mines the next block, they will always mine it as a default block (meaning they will

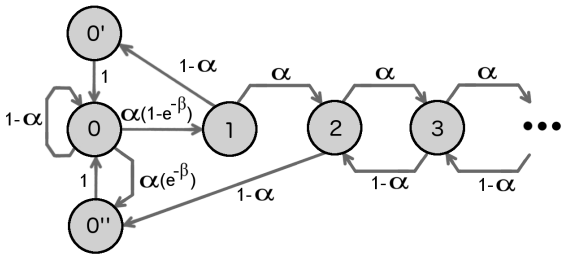


Figure 4.3: State machine for selfish mining with a cutoff, introducing state $0''$.

publish immediately). This represents the case when a block is too valuable to risk losing by employing selfish mining. We make an additional simplifying assumption that if the selfish miner has just been forced to publish their entire hidden chain, they will transition into state $0''$ instead of state 0. This is a simplification we have made to the strategy to make the mathematical analysis cleaner. In the next section, we will show why this simplification is needed and justify that this simplifying assumption is acceptable.

In order to calculate the selfish miner's expected revenue, we must again calculate the probability of the system being in any given state, and the chance that a transaction arriving to the system while in one of these states eventually ends up in a block mined by the selfish miner. From looking at the state transitions in figure 4.3, we can derive the following formulas relating the probabilities of being in each state:

$$p_i \alpha = p_{i+1}(1-\alpha) \tag{4.9}$$

$$\implies p_i = (\frac{\alpha}{1-\alpha})^{i-1} p_1 \tag{4.10}$$

$$p_{0''} = (1-\alpha)p_2 = \alpha p_1 \tag{4.11}$$

$$p_{0'} = (1-\alpha)p_1 \tag{4.12}$$

$$p_0 = \frac{p_1}{\alpha(1-e^{-\beta})} \tag{4.13}$$

We also know that the system is guaranteed to be in some state, which means the following.

$$p_0 + p_{0'} + p_{0''} + \sum_{i=1}^{i=\infty} p_i = 1 \tag{4.14}$$

Which can be used to show that

$$p_1 = \frac{\alpha(2\alpha-1)(e^{\beta}-1)}{3\alpha^2(e^{\beta}-1)+2\alpha-e^{\beta}} \tag{4.15}$$

With equations 4.10-4.13, this gives expressions for the probabilities of all the possible states the system could be in. In state p_0, it also matters how long the system has already been in this state when the transaction arrives (because it shifts to state $p_{0''}$ after a fixed time), so we must expand this expression to give the probability to be in this state, having had been here for all possible values of x units of time. We will denote this as $p_0(x)$. The probability that the system has been in state 0 for x units of time is equivalent to the time since the last block was found, which is given by a decaying exponential. This allows us to write the following:

$$p_0(x) = p_0 e^{-x} dx \tag{4.16}$$

We must now calculate the associated f_i values in order to calculate the expected fraction of the rewards claimed by the selfish miner. If we consider state 0, the chance that a transaction arriving to the system in this state ending up in a block owned by the selfish miner is dependent on how long it has been since the last block was found

(let this time be x). Also, let the expected block arrival time be normalized to 1 to simplify the derivation. If $x \geq \beta$, then the miner will immediately switch into state $0''$, and if they find a block, they will not hide it, so it is guaranteed to end up in the final longest chain (this happens with probability α). If, on the other hand, $x < \beta$, there are several possible outcomes. For all cases, the selfish miner must win the next block, which happens with probability α. If the selfish miner was not able to mine this block quickly (explicitly, longer than $\beta - x$ time, so that the miner transitions to state $0''$), then this is very similar to the last case, and this block will certainly make it into the final longest chain and the selfish miner will claim the transaction fee. The probability of this happening is the probability that enough time passes for the state transition to happen (which has probability $e^{-\beta+x}$) times the probability of the selfish miner finding a block (α). Alternatively, if the selfish miner is able to mine the block quickly enough that they will attempt to hide the block (mining the block in less than $\beta - x$ time, which happens with probability $(1 - e^{-\beta+x})$), then the selfish miner will be left with a hidden block, and this block (which contains the transaction) making it into the longest final change depends on who mines the next block. If the selfish miner mines a second block in a row, then both of these blocks will definitely make it into the longest final chain and the selfish miner will claim the transaction fee. The additional probability of this happening is α. If the rest of the network finds the next block instead, then it will trigger a race condition between the selfish block, and the newly found default block. Whoever wins the race will claim the transaction fee. The race happening has a total probability of $\alpha(1-e^{-\beta+x})(1-\alpha)$, and the probability that the selfish block is selected as the winner of the race is $(\alpha + (1 - \alpha)\gamma)$. Considering

the case where $x \geq \beta$ and the 3 cases when $x < \beta$, we get the following expression for $f_0(x)$:

$$f_0(x) = \alpha(e^{-\beta+x} + (1 - e^{-\beta+x})(\alpha + (1-\alpha)(\alpha + (1-\alpha)\gamma))), \forall x < \beta \tag{4.17}$$

$$f_0(x) = \alpha, \forall x \geq \beta \tag{4.18}$$

If the system is in state $0'$ when the transaction arrives, we can calculate the chance that the transaction fee is claimed by the selfish miner, $f_{0'}$. In this state, there is a selfish miner block racing against a default block. The next block mined will determine the outcome of the race and will both guarantee the transaction fee, and make it into the longest final chain. The chance that the selfish miner mines this block is α. This gives:

$$f_{0'} = \alpha \tag{4.19}$$

If a transaction arrives at the system in state $0''$, the next block that is mined will make it into the final chain (if the selfish miner mines the block in this state, they will immediately publish). Thus, the chance of the selfish miner claiming the transaction fee is equal to the chance that they mine with next block, which is α.

$$f_{0''} = \alpha \tag{4.20}$$

The logic for a transaction arriving to the system in state 1 is identical to the case without the block size cutoff, giving the following for f_1:

$$f_1 = \alpha + (1-\alpha)\alpha \tag{4.21}$$

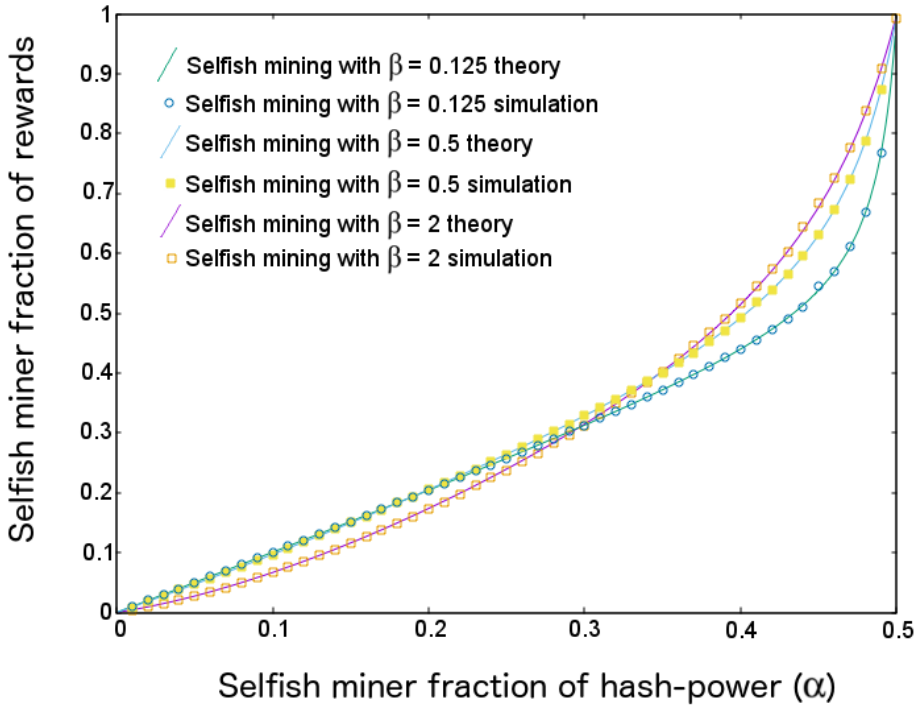


Figure 4.4: Theory matching simulation for a variety of cutoff thresholds for selfish mining, all with $\gamma = 0$. The smaller cutoffs do better for a miner with a smaller hash-power (α) and the larger cutoffs do better with a larger hash-power. Intuitively, this makes sense as a more powerful miner should be willing to risk a larger block to try to selfishly mine.

Our final case is if a transaction arrives to the system when it is in state i, $i > 1$. Because of our simplifying assumption and addition of state $0''$, in order for the transaction to **not** end up in a block mined by the selfish miner, the rest of the network must successfully mine $(i-1)$ blocks to cause the selfish miner to publish their entire hidden chain, and then (now that the system is in state $0''$) mine the next block. Thus, the chance that the selfish miner claims this transaction is given by:

$$f_i = 1 - (1-\alpha)^i \tag{4.22}$$

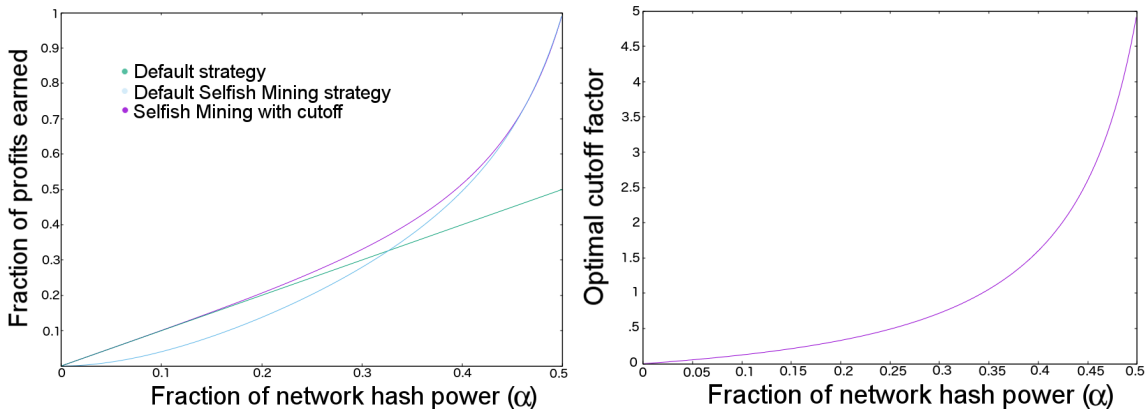
The total rewards earned by the selfish miner, in a transaction based system, using a cutoff is now given by the following:

$$R_{selfish} = \int_0^\infty p_0(x) f_0(x) + p_{0'} f_{0'} + p_{0''} f_{0''} + \sum_{i=1}^{i=\infty} p_i f_i \tag{4.23}$$

4.3 Simplification justification

Our simplifying assumption is that a selfish miner will default mine the next block after publishing their private chain of length 2 or more. This assumption is rooted in the fact that it takes finding many blocks in order to force the selfish miner to publish the private chain. Consider the system in state i (the selfish chain has i more blocks in its hidden fork than the default miners have in their fork, and i is at least 2). Say the last selfish block (the one bringing the system to state i) was found a time t_0. Also say that this is the last block the selfish miner finds before they are forced to publish their private chain (every time the selfish miner has to publish their chain, which is inevitable with $\alpha < \frac{1}{2}$, there will always exist a last block found by the selfish miner before the publishing). If the selfish miner is to publish their chain, it means that the rest of the network will need to find $i - 1$ consecutive blocks. Let these blocks be found at times t_1, t_2, ..., t_{i-1}. Finally, after publishing their private chain and at time t_i, the selfish miner finds a block (this is the case this simplification exists for). All of the transaction fees that have accumulated in the network since time t_0 will be included in this block. Because there have been i blocks found in the time $t_i - t_0$, the chances that the block found at t_i has a size factor less than the cutoff β (which is typically less than 1) is small. So, while this assumption requires that the next block is default mined, it is extremely likely that the selfish miner would make this decision anyway based on the size factor of the block.

This simplification drastically simplifies the analysis for f_i. In order to do the full analysis, one would need to treat f_i as $f_i(y)$ where y represents some time value that has passed since the last block in the private chain was found. When we simulate this strategy using our Bitcoin mining simulator, we implement the full version of the strategy that will consider selfish mining a block, even after publishing. As we can see from figure 4.4, even with this simplification in the theory, it very closely predicts the results of the simulation.



(a) (b)

Figure 4.5: On the right, we show the ideal cutoff factor, β, for a selfish miner with mining power α, and $\gamma = 0$. On the left we show that a selfish miner using the optimal cutoff outperforms both the original selfish mining protocol and default mining for all values of α, with $\gamma = 0$.

4.4 Analysis of selfish mining with cutoff

The optimal cutoff β for a selfish miner to use is dependent on α and γ because both of these parameters are important to describe the risk of losing a hidden block mined with the selfish mining strategy before a hidden chain of length 2 or more can be found. We focus mainly on the case where $\gamma = 0$, as this is a worst case for the selfish miner, but this could easily be generalized to other values of gamma.

As can be seen in figure 4.5a, the selfish mining strategy with a cutoff is better than the selfish mining without a cutoff for all α. Additionally, the selfish mining with a cutoff is better than default mining for all α as well. As γ increases from 0, the margin by which the selfish mining with cutoff outperforms selfish mining without a cutoff decreases. At the point were $\gamma = 1$, the selfish mining with cutoff and the traditional selfish mining align, because with γ equalling 1, the risk of losing the first hidden block drops to 0, the optimal cutoff jumps to infinity, and the strategies become identical.

Chapter 5

Future Work

There is a lot more work that needs to be done in this area of Bitcoin. All of this analysis relies on miners acting rationally with regards to earning as many profits as possible (in Bitcoin). Certainly, some Bitcoin miners will refuse to deviate from the default compliant strategy, even if this strategy is inferior to other potential strategies. Work that we are currently pursuing investigates how the forking strategies evolve in the presence of a variable fraction of the network that refuses to abandon the default mining strategy. Also worth considering are forking strategies that are not limited to just forking the most recent block, but can fork multiple blocks back if necessary. It could be very revealing to consider what will happen if miners are using these strategies and the difficulty readjusts. Finally, relaxing the model to be only mostly transaction fee based, and including a small block reward would be worth looking at.

There is much that can still be done with regards to selfish mining and Bitcoin transaction fees. An avenue that needs to be explored is the combination of selfish mining with the other forking strategies. The advantage for the selfish miner changes if much of the rest of the network is utilizing the forking strategies (which would make it much easier for the selfish miner to get a longer private chain), and this needs to be explored. Additionally, because selfish mining could be done profitably by miners

of any size in a transaction fee environment, it would be valuable to thoroughly investigate how multiple selfish miners would interact in a system with each other. Finally, it would be interesting to consider how other modifications to selfish mining (such as stubborn mining strategies presented in [6]) would be affected by the switch to transaction fees.

Analysis on where miners chose to mine in a fork is also an important area to look at. Petty compliant mining should beat the default strategy in the system today, but the margins would be small because so little of a miner's profits come from transaction fees, and the circumstance that causes the petty behavior to deviate from the default behavior is relatively uncommon. Looking to see if miners are already choosing the less wealthy side in a fork as opposed to the oldest side could lead to interesting results, especially if it reveals a fraction of the network that has switched off the default strategy.

Chapter 6

Impact for Bitcoin

There are several results here that important for the Bitcoin system. Let us first consider the forking strategies that we introduced in section 3. If miners are aggressively forking each other, then the number of orphaned blocks increases substantially. This has several impacts, many of which are already known to be bad. The difficulty will drop because the longest chain over a period of two weeks will be missing a large portion of the blocks it should have, given the network hash-rate. Also, because so many blocks will become orphaned, a dedicated attacker trying to perform a 51% attack will have a much easier time executing the attack because the longest chain created by the rest of the network will be shorter than it should be. If all miners act rationally with regards to profits, and we do, in fact, see miners utilizing function forking with smaller and smaller coefficients, then there are very serious concerns for the usability of the system. If miners are only including a very small portion of the possible transactions in their blocks, a user who broadcasts a transaction to the network will have to wait, on average, a very long time for a miner to actually include their transaction in a block. Additionally, they cannot expedite the process by adding a larger fee to their transaction (thus making a transaction market), because a more

valuable transaction fee is not any more appealing to a miner who is only trying to take a small fraction of the fees (by total value, not by number of transactions).

The root of all these problems is that the value to a miner for mining a block must not change with time. One of the assumptions of the model made in chapter 2 was that miners have the capability of claiming all possible transactions available in their next block. This closely relates to the block size debate that is ongoing at the time of writing. This assumption does not hold if the block size is small and there is a backlog of transactions. In this case, immediately after a block is found, all miners searching for the next block can immediately fill the block with transactions, making the system more similar to how it is with a block reward. A backlog, however, significantly impacts the usability of the system. In order to solve the problem, the backlog must contain a significant volume of value in transaction fees, not just a large number of transactions (if there is mostly dust in the backlog, miners will fill their blocks with dust and await larger transaction fees to arrive on the network, and swap out dust in their block for these fees. In this case, it would be very similar to a mostly transaction based system with a small minting reward). The backlog must also be large enough that it rarely, if ever, empties. This means that users who submit transactions to be published may need to wait potentially unbounded amounts of time for their transactions to make it into the blockchain.

Many of the arguments about block size (potentially even infinitely large blocks) focus on the impact the size of the block has on the value of transaction fees [7]. This work goes beyond that to show that even if we assume that the transaction fees are large enough in magnitude to completely replace the minting reward, other problems can arise if the value to a miner of mining a block changes over the timescale of the block, as it does in the case of transaction fees and large blocks.

Bibliography

[1] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008.

[2] Sanjeev Arora, Elad Hazan, and Satyen Kale. "The Multiplicative Weights Update Method: a Meta-Algorithm and Applications." In: *Theory of Computing* 8.1 (2012), pp. 121–164.

[3] Christian Decker and Roger Wattenhofer. "Information propagation in the bitcoin network". In: *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*. IEEE. 2013, pp. 1–10.

[4] Michael Bedford Taylor. "Bitcoin and the age of bespoke silicon". In: *Proceedings of the 2013 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*. IEEE Press. 2013, p. 16.

[5] Ittay Eyal and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable". In: *Financial Cryptography and Data Security*. Springer, 2014, pp. 436–454.

[6] Kartik Nayak et al. *Stubborn mining: Generalizing selfish mining and combining with an eclipse attack*. Tech. rep. IACR Cryptology ePrint Archive 2015, 2015.

[7] R Peter. "A Transaction Fee Market Exists Without a Block Size Limit". In: (2015).