8 Selfish mining

In 2013, in a seminal paper of Eyal and Sirer [15], it was demonstrated that if one entity controls more a third of the total mining power, Nakamoto consensus fails to be *incentive-compatible*.

In this chapter we unpack this concept and its ramifications. If the reader would like to skip the details, we offer the following summary.

Summary 1. Selfish mining is an exploit that is available under Nakamoto consensus, made possible by the difficulty adjustment. Selfish mining allows an entity who controls $p > 1/3$ of the hashrate to allocate to themselves a fraction $S(p) > p$ of the block rewards, thus unfairly increasing their revenue. Because the portion $S(p)$ grows non-linearly with p above $p = 1/3$, the selfish miner is incentivized to be excessively greedy in acquiring hashrate, with increasing return on investment all the way up to the point where the selfish miner controls 50.1 % of the miner power, at which point selfish mining becomes monopolistic control.

The notion of incentive-compatibility is from mechanism design. A mechanism is considered to be *incentive-compatible* if every participant achieves their best outcome by following the rules as they were designed to be followed. This concept arises in many systems such as voting; many voting methods such as plurality, approval voting, ranked choice, score-then-automatic-runoff, etc., are designed so that the voters will choose honestly rather than strategically (a famous theorem, Arrow's impossibility theorem, states that there is no voting mechanism that is free from the possibility of manipulation by strategic voting).

Definition 8.1. The *default compliant* strategy for Bitcoin miners is the following:

- Each miner works to append the heaviest chain. If there are two, they work on the first one that came to their attention.
- Each miner publishes each block the instant they have discovered it. This allows other miners to immediately start working on the next block with transactions from the mempool.
- Miners do not intentionally omit fee-paying transactions from the mempool for which there is sufficient blockspace.

Remark 8.1. The last condition is not completely deterministic. Typically miners would order available transactions by fee per blockspace and process as many as possible. There are small wrinkles in this; for example, they may or may not prioritize replace-by-fee transactions, and they may group child-pays-for-parents transactions together. Different miners may use different block-packing algorithms, and some may choose to impose minimum transaction fees.

When miners follow these rules, the network should function continuously as expected. However, Nakamoto consensus allows a miner with enough hashrate to employ

https://doi.org/10.1515/9783110772845-008

what is called selfish mining. Selfish mining works by taking advantage of the difficulty adjustment. By withholding mined blocks from the public, selfish miners can force other miners to waste resources, making the network as a whole less efficient at producing blocks. A careful calculation reveals that the selfish miner can increase their proportion of the blocks produced over a longer time period, at the cost of creating a few stale blocks before the difficulty adjusts. Because the difficulty adjustment rewards miners according to their relative block production, not their hashrate, miners with 34 % of the hashrate can increase their rewards over time.

This may seem counterintuitive at first: By keeping a block secret, there is a risk that someone else will discover a block first. In this case the selfish miner still publishes the block, but it may be too late: Other miners will have seen the other block first and will ignore the block. The selfish miner then needs to find another block on top of their previously hidden block or lose it forever. However, on the other side of this is the possibility that they will mine another block, which they also keep secret. Once they have built a two-block lead, they may continue to add to the lead in secret without any fear of losing the block. The instant that a competing miner publishes a block that comes within one block of catching up, the selfish miner reveals their secret chain to the network and this supplants the block of the competing miner or miners.

Remark 8.2. We use p (as we have been through much of this text) to denote the hashrate of the selfish miner, but much of the literature (including the original paper [15]) on selfish mining uses α.

8.1 Exploiting the difficulty adjustment

The difficulty adjustment is the only major real-time "parameter tweak" provided by the protocol. Because the difficulty parameter is something that responds to miners' behavior, it is not surprising that this is the wrinkle that allows for the most significant exploit.

If the difficulty adjustment were not present, selfish mining would not work; this has been proven in [21, Theorem 4.4]. In particular, selfish mining is a net loss for the miner in the short term, where the effect of selfish mining is to drastically slow down the block production. After block production has been slowed down, the next difficulty adjustment compensates for this, and the rate of block production returns to normal. In this chapter we present basic selfish mining and assume this is a long-term strategy used by a miner who plans on using it for several subsequent difficulty periods.

Exercise 8.1. Using chainwork, explain how difficulty adjustment allows a party with extremely large hashpower to catch up and become the chain with the most work while producing only a fraction of the total blocks.

Exercise 8.2. Suppose that a miner with hashrate p waits until N blocks have been mined in a difficulty period, and then mines the 2,016th block of the previous diffi-

culty period. The miner then mines this chain in secret, producing blocks with lower difficulty. Suppose that instead of the heaviest-chain rule measuring most chainwork, Bitcoin used the longest-chain rule. What hashrate p would the miner need in order to be likely to mine 2,016 blocks in the current period before the other miners do? For this exercise, you may replace the Poisson process with a steady deterministic process, for simplicity.

8.2 Basic selfish mining

In order to analyze this game, we need to realize it as a series of repeated rounds. Each round ends with either the selfish miner capitulating and returning to mine the longest chain or the selfish miner publishing their last two secret blocks to win the round.

We can be more precise: As in Chapter 5 we set up the game states. Instead of being behind in the block race, the selfish miner may be secretly ahead in the block race. The game states will be

$$\{(i,j) \mid \text{ selfish miner has mined } i \text{ blocks total, others have mined } j \text{ blocks}\}.$$

The game starts at state $(0,0)$, at which point the selfish miner attempts to mine a block. If the others find a block first, the selfish miner starts again mining on top of the longest chain; another round begins and the state reverts to $(0,0)$.

However, if the selfish miner finds a block, they keep this secret, and the state moves to $(1,0)$. From $(1,0)$ there are two options:

1. If the others find the next block, the selfish miners typically will go ahead and publish their block, hoping that perhaps some fraction of miners will see this block first and mine on top. This puts the game at state $(1,1)$. Regardless of who sees the block, the selfish miners continue to attempt to mine on top of their own block; if they are lucky, they will find a second block and supplant the block the others produced – the round ends with the miners winning two blocks and the next round restarts at $(0,0)$. Similarly, if the others win a block from state $(1,1)$ they permanently claim for themselves two blocks and a new round starts.
2. The selfish miner finds a block and the state moves from $(1,0)$ to $(2,0)$. Now at this point the selfish miner has the upper hand and uses the following strategy: If they are at least three blocks ahead and they see the others publish a block, they publish an equalizing block. If they are two blocks ahead and the others publish a block, they publish two blocks, exhausting their secret supply. (This may or may not be the optimal strategy; the selfish miners may wish to push their luck and keep these blocks hidden. However, the strategy being used here is straightforward enough to model.)

In our model, we assume:

- A single selfish miner has hashrate p.
- The other miners follow the first-seen, longest-chain rule.
- Network propagation is near instantaneous.
- Fees are not important.

We are going to compute first the expected value of the number of blocks won by the selfish miners in each round and then the expected number of blocks won by the others in each round. This will determine what proportion of blocks is expected to be obtained by the selfish miners as the number of rounds becomes large.

Let

$$f_{\mathrm{SM}} := \text{number of blocks won by selfish miners,}$$
$$f_O := \text{number of blocks won by other miners.}$$

We would like to find

$$\mathbb{E}[f_{\mathrm{SM}} \mid (0,0)]$$

and

$$\mathbb{E}[f_O \mid (0,0)].$$

To begin, note that

$$\mathbb{E}[f_{\mathrm{SM}} \mid (0,0)] = p\mathbb{E}[f_{\mathrm{SM}} \mid (1,0)] + (1-p)\mathbb{E}[f_{\mathrm{SM}} \mid (0,1)],$$

where the rightmost term is 0 as the selfish miners win nothing and the round restarts if the others take the first block (we continue to use the conditional expectation formula to compute expectations; recall (2.6)).

Then we compute

$$\mathbb{E}[f_{\mathrm{SM}} \mid (1,0)] = p\mathbb{E}[f_{\mathrm{SM}} \mid (2,0)] + (1-p)\mathbb{E}[f_{\mathrm{SM}} \mid (1,1)], \tag{8.1}$$
$$\mathbb{E}[f_{\mathrm{SM}} \mid (2,0)] = p\mathbb{E}[f_{\mathrm{SM}} \mid (3,0)] + (1-p)\mathbb{E}[f_{\mathrm{SM}} \mid (2,1)]. \tag{8.2}$$

Now

$$\mathbb{E}[f_{\mathrm{SM}} \mid (1,1)] = 2p.$$

This follows because in this tie-breaking situation, one team or the other must win, taking two blocks.

Also,

$$\mathbb{E}[f_{\mathrm{SM}} \mid (2,1)] = 2, \tag{8.3}$$

as this state results in the selfish team immediately broadcasting their two blocks (the state would only exist for a small amount of time depending on network latency, which we assume is 0).

We claim also that for $k \geq 3$, $l \geq 2$,

$$\mathbb{E}[f_{\mathrm{SM}} \mid (k, k-l)] = \mathbb{E}[f_{\mathrm{SM}} \mid (k-1, k-l-1)] + 1.$$

To make this argument, recall that we are working on a Bernoulli space. The set of paths starting from state $(k, k-l)$ and the set of paths starting from $(k-1, k-l-1)$ are identical. Each path ends with probability 1, with the selfish miners winning the round. In the former case when they win they will have one more block reward to their credit, compared to the same path in the latter case.

Now for $k \geq 3$,

$$\begin{aligned}\mathbb{E}[f_{\mathrm{SM}} \mid (k,0)] &= p\mathbb{E}[f_{\mathrm{SM}} \mid (k+1,0)] + (1-p)\mathbb{E}[f_{\mathrm{SM}} \mid (k,1)] \\ &= p\mathbb{E}[f_{\mathrm{SM}} \mid (k+1,0)] + (1-p)(\mathbb{E}[f_{\mathrm{SM}} \mid (k-1,0)] + 1).\end{aligned}$$

Now we define

$$e(x) = \mathbb{E}[f_{\mathrm{SM}} \mid (x,0)],$$

and we see that for $x \geq 3$ we have

$$p\big(e(x) - e(x+1)\big) + (1-p)\big(e(x) - e(x-1)\big) = (1-p).$$

To make this equation work for $x = 2$, we declare that $e(1) = 1$. Then, combining equation (8.2) with (8.3) we can check that with $e(1) = 1$,

$$p\big(e(2) - e(3)\big) + (1-p)\big(e(2) - e(1)\big) = (1-p).$$

Thus the function $e(x)$ satisfies

$$e(x) - pe(x+1) - (1-p)\big(e(x-1)\big) = 1-p. \tag{8.4}$$

We can set this up as a boundary problem, declaring $e(1) = 1$ as the left boundary value. There is not an obvious natural choice for the right boundary value, as there is no right boundary. However, what we can do is solve the problem for some reasonable boundary values with boundary points far to the right (and hence improbable) and then take the limit.

Specifically, recalling discussions following (5.7) and (5.18) with p and $(1-p)$ swapped, we know that

$$e(x) := c_1\left(\frac{1-p}{p}\right)^x + c_2 + \frac{1-p}{1-2p}x$$

is the general solution to the non-homogeneous finite difference equation (8.4).

We can deal with the lack of a right boundary value by imposing a harmless condition on the strategy: Suppose we assume that for some large value N, if the selfish miner is N values ahead they will broadcast their transactions and start the round over. This is an increasingly improbable event for larger and larger values of X. In this case we can set the right boundary value to be $e(N) = N$. Then solving for the boundary conditions

$$c_1\left(\frac{1-p}{p}\right) + c_2 + \frac{1-p}{1-2p} = 1,$$
$$c_1\left(\frac{1-p}{p}\right)^N + c_2 + \frac{1-p}{1-2p}N = N,$$

we get

$$c_1\left[\left(\frac{1-p}{p}\right) - \left(\frac{1-p}{p}\right)^N\right] = \frac{p}{1-2p}(N-1),$$

so

$$c_1 = \frac{\frac{p}{1-2p}(N-1)}{(\frac{1-p}{p}) - (\frac{1-p}{p})^N}.$$

Taking $N \to \infty$, this goes to zero, as the geometric term in the denominator grows faster than the linear term on the top. So we take $c_1 = 0$, and we can immediately solve

$$c_2 = -\frac{p}{1-2p},$$

so

$$e(2,0) = -\frac{p}{1-2p} + \frac{2-2p}{1-2p} = \frac{2-3p}{1-2p}.$$

Now

$$\begin{aligned}\mathbb{E}[f_{\mathrm{SM}} \mid (0,0)] &= p^2\mathbb{E}[f_{\mathrm{SM}} \mid (2,0)] + 2p^2(1-p)\\ &= p^2\frac{2-3p}{1-2p} + 2p^2(1-p).\end{aligned}$$

Computing the expected number of blocks recorded by the others is a bit easier:

$$\begin{aligned}\mathbb{E}[f_O \mid (0,0)] &= (1-p) + p\mathbb{E}[f_O \mid (1,0)]\\ &= (1-p) + 2p(1-p)^2.\end{aligned}$$

So now we may compute the ratio of the expected number of blocks won by the selfish miner to the expected total of the blocks mined on the longest chain over a long time period, namely,

$$\frac{\mathbb{E}[f_{\mathrm{SM}} \mid (0,0)]}{\mathbb{E}[f_{\mathrm{SM}} \mid (0,0)] + \mathbb{E}[f_O \mid (0,0)]} = \frac{p^2 \frac{2-3p}{1-2p} + 2p^2(1-p)}{p^2 \frac{2-3p}{1-2p} + 2p^2(1-p) + (1-p) + 2p(1-p)^2} = \frac{4p^4 - 9p^3 + 4p^2}{p^3 - 2p^2 - p + 1}.$$

Define

$$S(p) := \frac{4p^4 - 9p^3 + 4p^2}{p^3 - 2p^2 - p + 1}. \tag{8.5}$$

The following is a graph of $\frac{S(p)}{p}$ (see Fig. 8.1). When p crosses the value $\frac{1}{3}$, $\frac{S(p)}{p}$ becomes larger than 1 and increases towards 2.

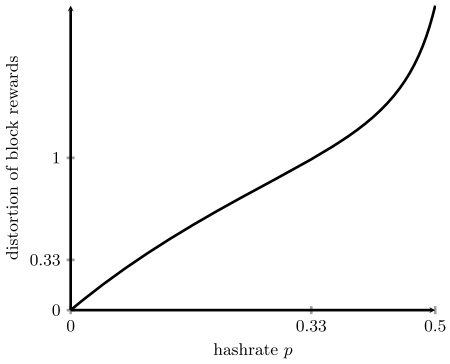

Figure 8.1: Distortion of hashrate to selfish miners $\frac{S(p)}{p}$.

For example, a miner with 35 % of the hashrate can increase their fraction of block rewards to 36.651 %, while a miner with 40 % can increase their portion to 48.372 %.

We may also consider the graph of

$$\frac{1 - S(p)}{1 - p},$$

which describes the distortion of the returns of the other miners (see Fig. 8.2). For example, at $p = 0.4$ the others will notice an 86.047 % return of what they would expect were selfish mining not occurring.

Open-ended Question 8.1. What logistical problems would a pool attempting to selfishly mine need to overcome? Are there ways to overcome this?

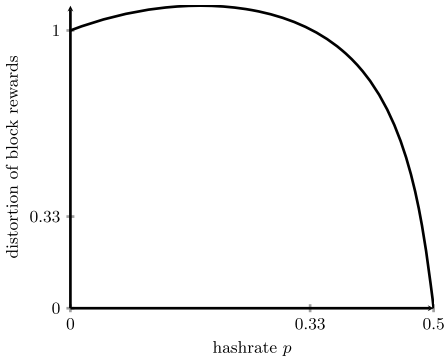

Figure 8.2: Distortion of hashrate to non-selfish miners $\frac{1-S(p)}{1-p}$.

8.3 Market distortions

Selfish mining creates obvious market distortions. Selfish miners enjoy an increase in revenue while the others suffer a decrease in revenue. For an individual miner who is given the option to join a selfish mining cartel, the difference can be quite significant. A selfish miner will always be willing to expand their cartel: Each increase in hashrate increases the distortion itself, further increasing the return of selfish mining. Thus the moment a miner with 33.33 % of the hashrate begins selfish mining and welcomes others to join, this creates a prisoners' dilemma among the remaining miners, possibly leading to *snowballing*, as the rewards for joining the cartel become increasingly steeper as the cartel grows. Defecting to the selfish mining pool will always be a dominant strategy. The distortion will also be passed onto the ASIC market, countering the anti-centralization tendencies described in Chapter 7.

In fact, if we consider examples such as Exercise 7.2 in the regime of selfish mining, we see much different dynamics. A miner with the lowest energy costs, instead of mining to the point where returns are diminished, will aggressively take over the entire block creation game: Selfish mining with 50 % of the hashrate should return 100 % of the rewards with high probability.

8.4 Markov chains

The original selfish mining paper used Markov chains to analyze the expected benefits of selfish mining. In this section we give a quick overview of Markov chains. These will be useful in later chapters as well.

A Markov chain is a process in which the state changes between a set of possible states, such that the probability of changing from one state to another does not depend on the current time or history. These processes are often analyzed using the transition

matrix, which consists of the probabilities of moving between states. The matrix $\{a_{ij}\}$ is determined by

$$a_{ij} := P(\text{next state is } j \mid \text{current state is } i).$$

Such a matrix is called a *stochastic matrix* (note that we are using left matrix multiplication, which is non-traditional).

Example 8.1. Suppose that a flea is randomly hopping around the vertices of a triangle. Each second, with probability p, the flea hops counterclockwise, and with probability $1-p$ the flea hops clockwise. If we order the vertices moving counterclockwise N_1, N_2, and N_3, the system has three possible states. The transition matrix is

$$\begin{pmatrix} 0 & p & 1-p \\ 1-p & 0 & p \\ p & 1-p & 0 \end{pmatrix}.$$

In general the transition matrix will be *right-stochastic*, that is, the rows must sum to 1. This simply means that the conditional probabilities a_{ij} for a fixed current state i must sum to 1.

Example 8.2. Suppose a miner with hashrate p attempts to mine from behind, starting from one block behind, giving up only if they fall three blocks behind. This can be set up as a Markov chain. There are five states: (1) The miner has given up, (2, 3) the miner is two blocks or one block behind, (4) the miner has tied but is not first-seen, and (5) the miner pulls ahead and has won. With this ordering, the transition matrix is

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1-p & 0 & p & 0 & 0 \\ 0 & 1-p & 0 & p & 0 \\ 0 & 0 & 1-p & 0 & p \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Observe that this is not *left-stochastic*: The columns do not sum to 1. Also observe that the first and last rows contain only a single non-zero entry, which must necessarily be a 1. This means that these states are fixed, or terminal. Once the miner has given up, the game is over. Once the miner has won, the game is over, and the game remains in the state "miner won." If the game is over, there is 0 probability of moving to any other state.

The utility of a transition matrix becomes clear when iterating: One can take powers of the matrix to determine the probabilities of the various game states at future times. This is because of the following fact:

$$P\{\text{state after next is } j \mid \text{current state is } i\}$$
$$= \sum_k P\{\text{state after next is } j \mid \text{next state is } k\} P\{\text{next state is } k \mid \text{current state is } i\},$$

or

$$P\{\text{state after next is } j \mid \text{current state is } i\} = \sum_k a_{ik} a_{kj} = \left(A^2\right)_{ij}.$$

This can be iterated. In the above example,

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1-p & 0 & p & 0 & 0 \\ 0 & 1-p & 0 & p & 0 \\ 0 & 0 & 1-p & 0 & p \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}^3$$
$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ p(p-1)^2 - p + 1 & 0 & -2p^2(p-1) & 0 & p^3 \\ (p-1)^2 & 2p(p-1)^2 & 0 & -2p^2(p-1) & p^2 \\ -(p-1)^3 & 0 & 2p(p-1)^2 & 0 & p - p^2(p-1) \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

The matrix on the right-hand side offers the transition probabilities of three rounds into the future. For example, the entry

$$a_{25} = p^3$$

represents the probability that a system starting at state 2 will move from state 2 (two blocks behind) to state 5 (miner wins) in three rounds.

Some Markov chains (like the example above) have terminal states. As long as there is a chain of positive probabilities that lead to a terminal state for each state in the system, the system will arrive in a terminal state with probability 1 as the process is repeated indefinitely. In the above example, the game must end at some point. Iterating the transition matrix infinitely many times results in matrices which converge to a matrix which should have non-zero entries only in terminal columns.

Some Markov chains do not have terminal states (consider the flea in Example 8.1). These processes are analyzed using steady-state distributions: Suppose there is a vector $\vec{\mu} = \{\mu_1, \mu_2, .., \mu_n\}$ with the property that

$$\vec{\mu} = \vec{\mu} A,$$

that is, $\vec{\mu}$ is a right eigenvector with eigenvalue 1. This is called a *steady-state distribution*. In the flea example, suppose instead that there are 30,000 fleas. If the vector $\vec{\mu}$ represents the number of fleas at each vertex and

$$\mu_1 = \mu_2 = \mu_3 = 10{,}000,$$

then the expected number of fleas after each flea hops once again is going to be again 10,000 on each node. Note also that if 30,000 fleas start on one node and randomly jump, the distribution at first will appear not random, but as the game is repeated, the distribution of fleas will move closer toward the steady state until only random noise is what separates the distribution from the steady state.

The flea triangle is an example of an ergodic system: Eventually we expect a flea to pass through every node, regardless of where the flea started. The mining game is not. A miner starting in terminal state will never play enough rounds of the game to arrive in a non-terminal state.

8.5 Selfish mining analyzed via a Markov process with cuts

The selfish mining as we have presented above has infinitely many states. Because selfish mining is a repeated process with every would-be terminal state reverting back to the starting state, it is ergodic. When an honest block is mined compelling the selfish miner to reveal all of their hidden blocks, the game state returns to the original state and selfish mining starts again. There are no terminal states. We are interested in computing the steady state. That is, if we start the game now, 10,000 blocks into the future, what is the probability that we will be at each state of the game? There will be a steady-state distribution describing these probabilities:

$$\begin{aligned}
\mu_0 &= P(\text{game is at state 0}),\\
\mu_{0'} &= P(\text{game tied at (1,1); selfish miners' block is not first-seen}),\\
\mu_1 &= P(\text{selfish miners have secret one-block lead}),\\
\mu_2 &= P(\text{selfish miners have secret two-block lead}),\\
&\text{etc.}
\end{aligned}$$

One can explicitly compute all of the transition probabilities. For example, the probability of moving from state (0,0) to (1,0) is p, the probability from moving from state (1,0) to (1,1) is $1 - p$, etc.

At first one may be tempted to compute the eigenvector for the process. But this can be difficult. Instead, there is a trick that makes this easier, which is used in the computation [15]. Instead of computing the eigenvector, draw several *graph cuts* on the state diagram. A graph cut is simply a partition of the set of states into two bins, and represents a line that can be crossed one way or the other in each transition. By the definition of the steady state, the probability of crossing the cut in one direction must necessarily be equal to the probability of crossing in the other direction – this is the only way the distribution will remain steady. This means we compute the probability of both of these two events (crosses one way, crosses the other) and turn these into a useful equation.

In the selfish mining example, we draw a cut between a bin containing the two states $(0,0)$ and $(1,1)$ and a bin containing the rest of the possible states. Note that the only way to move away from the set of states $(0,0)$ and $(1,1)$ is to move from $(0,0)$ to $(1,0)$: The state $(1,1)$ moves to the state $(0,0)$. On the other hand, one can go to state $(2,0)$ from $(0,0)$ – this happens with probability $(1-p)$, and one can go from $(1,0)$ to $(1,1)$, which also happens with probability $(1-p)$. Since the probability of moving from $(0,0)$ to $(1,0)$ is p, we have the following equation:

$$p\mu_0 = (1-p)\mu_1 + (1-p)\mu_2. \tag{8.6}$$

Next we isolate the state $(1,1)$ as a bin of its own. With probability 1 this transitions to $(0,0)$. The only way to get to state $(1,1)$ is the event that happens with probability $(1-p)$ from state $(1,0)$. This gives us

$$\mu_{0'} = (1-p)\mu_1. \tag{8.7}$$

Next consider the cut involving states $(0,0)$, $(1,1)$, and $(1,0)$. To move from this, there is one way: Start at $(1,0)$ and move to $(2,0)$, which happens with probability p. To enter, however, this can only happen from state $(2,0)$. So the resulting equation is

$$p\mu_1 = (1-p)\mu_2. \tag{8.8}$$

In fact, for any $k \geq 2$ we also get

$$p\mu_k = (1-p)\mu_{k+1}. \tag{8.9}$$

Our next step is to combine the information contained in equations (8.6), (8.7), (8.8), and (8.9) to get information about the steady state.

One way to do this is to express everything in terms of μ_1. Directly substituting the left-hand side of (8.8) with the rightmost term in (8.6), we get

$$p\mu_0 = (1-p)\mu_1 + p\mu_1.$$

So we have

$$\begin{aligned}
\mu_0 &= \frac{1}{p}\mu_1, \\
\mu_{0'} &= (1-p)\mu_1, \\
\mu_2 &= \frac{p}{(1-p)}\mu_1, \\
\mu_{1+k} &= \left(\frac{p}{(1-p)}\right)^k \mu_1.
\end{aligned}$$

Now, we add all the probabilities up, knowing that they must sum to 1:

$$\mu_1\left[\frac{1}{p} + (1-p) + 1 + \sum_{k=1}^{\infty}\left(\frac{p}{(1-p)}\right)^k\right] = 1.$$

This leads to

$$\mu_1\left[\frac{2p^3 - 4p^2 + 1}{p(1-2p)}\right] = 1.$$

However, we do not actually need to compute μ_1. What we can do is compute the ratio with the top and bottom in terms of μ_1.

To compute the expected portion of blocks won by the selfish miners, we look at the states at which they may notch blocks. If they find the next block from state $(1,1)$ with probability p, they notch two blocks. From state $(2,0)$ with probability $(1-p)$ they fail to find the block but end up notching two blocks by publishing their secret chain. For states $(k,0), k > 2$, with probability $(1-p)$ a single block is notched. So these add up as

$$\begin{aligned} &2p\mu_{0'} + 2(1-p)\mu_2 + \sum_{k=2}(1-p)\mu_{1+k} \\ &= p\left(3 - 2p + \frac{1-p}{1-2p}\right)\mu_1 \end{aligned}$$

after some computations. For the honest miners,

$$\begin{aligned} &(1-p)\mu_0 + (1-p)\mu_{0'} \\ &\quad = (1-p)\left(\frac{1}{p} + 2(1-p)\right)\mu_1. \end{aligned}$$

Dividing the expected number of blocks to the selfish miners by the sums of the expected blocks to both types of miners, we get

$$\frac{p(3 - 2p + \frac{1-p}{1-2p})}{(1-p)(\frac{1}{p} + 2(1-p)) + p(3 - 2p + \frac{1-p}{1-2p})},$$

which one can check is equal to $S(p)$ defined by (8.5).

8.6 Selfish mining with partial cooperation

In the above example it was assumed that the selfish miner is trying to break ties alone. However, it is very possible that they will be assisted by other miners who mine on top of the selfish chain at state $(1,1)$. When the selfish miner publishes an equalizing block, perhaps within seconds or milliseconds of the others, there is the possibility that honest miners may still mine on top of the selfish miners' block. One possibility is that these miners are indeed honest, but due to network latency reasons, may actually be shown

the selfishly mined block first, in which case they will ignore the other block that arrives milliseconds later. This effect would be something that a selfish miner would be wise to attempt to maximize, by deploying listening and broadcasting nodes throughout the globe, ready to release the block the instant any nodes on the network hear of a competing block.

Another reason (discussed in §9.1.1) is that the miners may be mining strategically and amenable to fee-undercutting. For example, if a selfishly mined block leaves higher-fee transactions in the mempool, a strategic miner may choose to build on this block instead. Another possibility is that if the selfish miner is powerful enough (closer to 50 %) it may be more profitable for them to continue a race, even when they have fallen behind. This adds risk to anyone attempting to mine on a block opposing the selfish miners, and they may simply chose to mine on top of the selfish chain as this would be the safer of two options.

The parameter γ describes the ratio of miners not in the selfish pool that will mine the selfish chain when the equalizing block is posted. This gives three groups:

$$\begin{aligned} &p, && \text{selfish miners;}\\ &\gamma(1-p), && \text{miners who join selfish miners in a tie; and}\\ &(1-\gamma)(1-p), && \text{miners who never join the selfish miners.} \end{aligned}$$

To compute the effectiveness, we repeat the computations from §8.2 noting that

$$\mathbb{E}[f_{\mathrm{SM}} \mid (1,1)] = 2p + \gamma(1-p),$$

leading to

$$\mathbb{E}[f_{\mathrm{SM}} \mid (0,0)] = p^2\mathbb{E}[f_{\mathrm{SM}} \mid (2,0)] + (2p + \gamma(1-p))p(1-p),$$

while

$$\mathbb{E}[f_O \mid (0,0)] = (1-p) + p(1-p)(2(1-p) - \gamma(1-p)).$$

Thus

$$\begin{aligned} S_\gamma(p) &:= \frac{p^2\frac{2-3p}{1-2p} + (2p+\gamma(1-p))p(1-p)}{p^2\frac{2-3p}{1-2p} + (2p+\gamma(1-p))p(1-p) + (1-p) + p(1-p)(2(1-p)-\gamma(1-p))}\\ &= \frac{4p^2 - 9p^3 + 4p^4 + \gamma(p - 4p^2 + 5p^3 - 2p^4)}{p^3 - 2p^2 - p + 1}. \end{aligned}$$

Exercise 8.3. Find γ such that the break-even p for selfish mining with γ is 0.25.

Note that there are variations of the selfish mining strategy that can be explored. "Stubborn mining" is a variation of selfish mining where the selfish miner may continue to mine from behind or use different block-withholding strategies when ahead.

8.7 Defense against selfish mining

It makes most sense for the selfish miner to restrict the pool participants to those that the organizer can trust. If the pool is unrestricted, it would be trivial for any pool participant to release to the public any block that the selfish pool finds.

Selfish mining, if it is successful, will be detectable. It is blatant. There is no way to hide it if it is successful: Stale blocks are quite rare in the absence of selfish mining, whereas selfish mining requires not only frequent stale blocks, but frequent two-block reorgs which appear suddenly. Everybody will see this. The question remains whether the "honest" majority can do anything about it.

The problem, as we will discuss in §10.2.2 and §13.2.2, is that coordinating a response requires some sort of authority or consensus outside of the protocol and could destroy features of the network. For example, if the network detected an attack, one proposal to frustrate the selfish miner would be to refuse to accept two-block reorgs, rejecting them on the spot. However, this would be tantamount to a rolling one-block checkpoint (see §13.3). Such a rule is not in the spirit of Bitcoin, and would leave the protocol vulnerable to chain splitting at very low cost. It is not clear that anything can be done to stop selfish mining without introducing centralization or other serious problems for the protocol.