Draft 1

Kenan Öztürk, University of Zurich, 27.06.2023

NOTE: Table of contents is only included in drafts and will be removed for the final thesis. It is included to give an easy overview of the document's structure.

1 Abstract

This thesis studies finite blockchain games introduced by Ewerhart [1]. We restrict ourselves to games with two miners and seven or fewer stages and determine the unique subgame-perfect equilibrium, which results in a blockchain without any forks. For the two-stage game we draw the game tree and show how quickly the game tree and the number of possible blockchain grow with each additional stage of the game. Furthermore, we explain how one can find the subgame-perfect equilibrium using backwards induction. Finally, we examine five conjectures using computer simulations and verified that all of them are true for games with two miners and seven or fewer stages. Miners never switch to a shorter chain, they never switch away from the branch they won their first block on, payoff-maximizing miners behave monotonically, and coordination is never necessary to play a Nash equilibrium in any given stage.

2 Introduction

The idea of a blockchain has been around for a number of years. According to

Sherman et al. [2] a blockchain-like system was first proposed by David Chaum [3] in 1982. However, it was Nakamoto's influential Bitcoin whitepaper [4] that introduced blockchain technology to a wider (mainstream) audience.

There are a number of works related to the Bitcoin protocol and the games it gives rise to. To begin, we want to give a quick overview of the literature in this field that is most relevant to our work specifically. When we talk about equilibria in blockchain games we (and the game-theoretic literature as a whole) refer to the Nash equilibrium [5] defined by Nobel Prize winner John Nash in 1950. In blockchain games, we have at least two miners who seek to extend the blockchain and earn block rewards by doing so. A set of actions where no miner wants to unilaterally deviate from their chosen action constitutes a Nash equilibrium. There is no consensus on what the equilibria in blockchain mining games look like. Depending on the model and conditions, there can be equilibria with forks of the blockchain, equilibria without forks, or no equilibria at all.

Carlsten et al. [6] examine Bitcoin's protocol from a practical perspective. Their key insight is that in absence of block rewards, there exist situations where there is no equilibrium at all. This result is reinforced by Eyal and Sirer [7] who come to a similar conclusion. According to them, Bitcoin's protocol does not constitute an equilibrium and its mining incentives do not incentivize honest behaviour by the miners.

Biais et al. [8] offer a formal game-theoretic model of Nakamoto's proof-of-work blockchain protocol. They model the protocol as a stochastic game with infinite horizon and find that this blockchain game has multiple equilibria, and that there exist equilibria with forks. Additionally, they show that miners benefit from coordinating on a single chain without forks, but that coordination can also lead to abandoning portions of the blockchain.

Ewerhart [1] approaches blockchains from a different angle. Inspired by Biais et al. [8] they construct a game-theoretic model of a blockchain game that makes a number of simplifying assumptions, such as a finite horizon and the absence

of mining effort, but which allows for straight-forward analysis. The proposed model can easily be understood and shows how a small set of rules can generate stable consensus formation in blockchains.

Kroll et al. [9] study Bitcoin's mining mechanism and introduce a useful concept called monotonicity. They find that there is an equilibrium where all miners behave consistently with Bitcoin's reference implementation, but that there are also infinitely many equilibria where they behave otherwise. We will examine the concept of monotonicity closely, as it can be useful when applied to Ewerhart's model. If payoff-maximizing mining in Ewerhart's model is monotonic, then we argue that we can prove other statements about finite blockchain games using that property.

The literature concerns itself mostly with blockchain games that have no defined end, i.e. they have an infinite time horizon. There exists little research related to finite blockchain games where there is a clearly defined end point. We seek to improve the understanding of this niche with this humble contribution.

We limit ourselves to Ewerhart's model and attempt to analyse it first theoretically and then computationally. The theoretical part mainly serves our understanding of the model, while the goal of the computational analysis is to discover new features of the game. We utilize the concept of subgame-perfection which was first introduced by Reinhard Selten [10] in 1965. Our analysis is restricted to subgame-perfect equilibria in degenerate behavioural strategies of a game with $n = 2$ miners and fewer than $T = 7$ stages only. The reason for this is that the game expands rapidly in complexity, making a computational analysis infeasible for larger T or n. The computational complexity of the game does not allow us to consider games with more than seven stages, even for the case with only two miners. To see why, we present formulas that describe exactly how fast the number of intermediate nodes ϑ_T of the game-tree and the number of possible blockchains ξ_T grow with the time horizon T and the number of miners n in the finite blockchain game. We answer the following questions: First,

what does the equilibrium in a finite blockchain game look like? Second, will miners ever switch to a shorter chain? And third, how does a miner behave after winning their first block? These questions are all related, and as we will see, we stumble upon additional questions during our analysis and answer them directly: Is payoff-maximizing mining monotonic in a game where all miners maximize their payoffs? And is coordination ever necessary to play a Nash equilibrium in a given stage?

In contrast to the model proposed by Biais et al. [8], which produces equilibria with forks, we find that the subgame-perfect equilibrium does not lead to forks in Ewerhart's model. Additionally, we found that the payoff-maximizing strategy is monotonic, that a miner will continue mining on the same branch they won their first block on for the rest of the game, that a miner will never switch to a shorter chain than the one they are already mining on, and that coordination between the miners is never necessary to play a Nash equilibrium in any given stage. We were only able to confirm these conjectures for games with less than seven stages and cannot say anything about games with more than seven stages, because of the rise in complexity every additional stage brings.

The rest of this document is structured as follows. We start with the theoretical part by talking about the game trees of finite blockchain games in section 3 and subsequently show how we can find the subgame-perfect equilibrium of a finite blockchain games with two stages in section 4.1. Then, we continue by looking at longer games where the help of a computer is useful. We look at three and four-stage games in detail in sections 4.2 and 4.3, and subsequently present the subgame-perfect equilibrium for longer games of up to seven stages in section 5. Finally, we present our conjectures and discuss our findings in sections 6 and conclude in section 7.

3 Game Trees

To begin, we assume that the reader is familiar with Ewerhart's work [1] about finite blockchain games, since we make use of the same notation, definitions, and conventions laid out therein. Nonetheless, we want to provide the reader with a quick refresher of the material and its most important features.

Ewerhart [1] defines a blockchain as follows: A blockchain $\mathbb{B}$ consists of three components. First, a sequence of blocks $B = \{b_0, b_1, \ldots, b_T\}$ where block b_t was mined in stage $t \in \{0, 1, \ldots, T\}$. Second, a parent-child relation $\Leftarrow$ on B. And third, an assignment map $\iota : B \backslash \{b_0\} \to N$, which assigns every block to a miner, where $N = \{1, 2, \ldots, n\} \subset \mathbb{N}$ is the set of miners. A blockchain where the most recent block is b_t is denoted by $\mathbb{B}_t$, we will use this notation later.

A finite blockchain game as proposed by Ewerhart [1] has a finite horizon, i.e. it ends after a predetermined number of stages T. There are n players, also referred to as miners, who are rewarded for blocks they mine that are part of the longest chain after the game ends. They receive the payoff at the end of the game and each block yields a payoff of One. Furthermore, we assume that the blockchain grows in a stochastic manner, i.e. each miner wins any given stage $t \in \{1, 2, \ldots, T\} \subset \mathbb{N}$ with equal probability. The genesis block is generated in the beginning and has no parent block and no block reward associated with it. Every intermediate stage t follows the same pattern. First, all miners $i \in N$ simultaneously choose a block $\widehat{b}_{t-1}(i) \in B_{t-1}$, where $B_{t-1} = \{b_0, b_1, \ldots, b_{t-1}\}$ is the existing sequence of blocks. The block $\widehat{b}_{t-1}(i)$ is where miner i appends the next block b_t in the event that they win the current stage t. Next, Nature picks the winner i_t^* of stage t at random and with equal probability $\frac{1}{n}$. To conclude stage t, the new block b_t is appended to the winner's specified parent block $\widehat{b}_{t-1}(i_t^*)$.

Let us continue by looking at the game tree in order to better understand the game we are studying. Game trees are useful for our purposes because they allow us to visualize equilibrium paths. They aid in our understanding. We

did find any sources that drew the game tree explicitly like we do here in this section.

We are looking at a multistage game, where every stage-game is a game of imperfect information (since all players move simultaneously). Each stage starts with simultaneous moves by miner 1 and miner 2 and ends with a move by Nature N, who picks the winner of that particular stage. At the very end of the game, after the final stage's winner is chosen, Nature moves again, picking one of the longest chains at random and with equal probability.

In Figure 1 we can see the game-tree of a finite blockchain game with at least three stages. The first stage is trivial. Both miners only have one strategy, they have no choice but to mine on b_0. The stage ends after Nature picks a winner at random and with equal probability of $\frac{1}{2}$.

In the second stage miners have two actions to choose from. They can mine on b_0 or b_1. Both miners have two information sets in this stage. The second miner's information sets span two nodes each. Both miners know who won the previous stage (and what the winner's strategy was), but they do not know where the other miner mines in the current stage.

The tree in Figure 1 is incomplete. At every large dashed circular node, another subgame starts (not to be confused with a stage-game). They are omitted for simplicity.

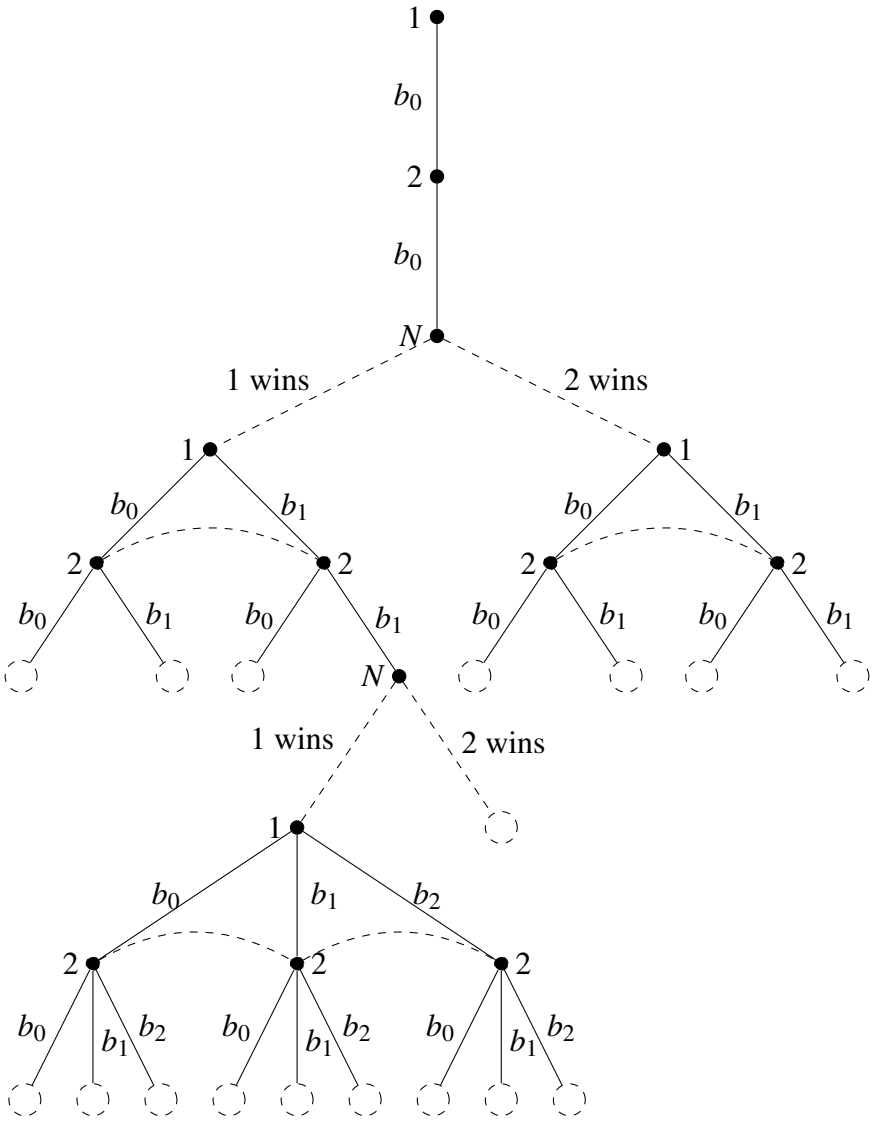

Figure 1 – A game-tree for a game with at least three stages.

An intermediate node in our context is a node in the game tree of a finite blockchain game that sits at the end of the intermediate stage t. Every stage-game ends in an intermediate node. When an intermediate node is reached, the next stage-game begins and all players make their simultaneous moves. After intermediate stage T (the very last stage), Nature moves once more and picks one of the longest chains. This step is always omitted in the game trees, but must be considered in the payoff calculations. There are two intermediate nodes

after the first stage of a finite blockchain game with T stages and $n = 2$ miners.

It is evident from looking at Figure 1 that the trees explode in size very quickly. But how quickly exactly? First consider the fact that the number of actions a miner can choose from in any given intermediate stage t is equal to t. In the first stage there is only one action (mine on b_0), in the second stage there are two actions (mine on b_0 or b_1) et cetera.

Consider a finite blockchain game with T stages and n miners and denote the number of intermediate nodes in the game tree after the last intermediate stage T by a function $\vartheta_T(n)$. By looking at the general game tree in Figure 1 we can see that the number of intermediate nodes after the final stage $\vartheta_T(n)$ must be equal to the number of intermediate nodes after the previous stage $\vartheta_{T-1}(n)$ multiplied by the factor $T^n \cdot n$. Why? Recall that all n miners can choose from T actions in stage T, which means that there are T^n possible strategy profiles for this stage. Furthermore, Nature picks one winner amongst the n miners, which means we must also multiply by n. Note that $\vartheta_T(n)$ is the number of nodes before Nature picks one of the longest chains as the winner.

After the very fist stage there are n intermediate nodes because miners only have one strategy they can play and Nature picks one of them as the winner. This allows us to define $\vartheta_T(n)$ recursively.

$$\vartheta_1(n) = n \qquad \text{and} \qquad \vartheta_t(n) = \vartheta_{t-1}(n) \cdot t^n \cdot n$$

We can rewrite this as follows:

$$\vartheta_T(n) = \prod_{t=1}^{T} (t^n \cdot n) = n^T \cdot (T!)^n$$

In the table below (Table I) we count the terminal nodes before Nature moves, because we cut the game tree before Nature picks the winner of the final stage. That number is half the number of nodes after Nature moves, because we have

two miners. This explains the factor $\frac{1}{2}$. Why do we omit Nature's last moves? Nature's last moves represents a simple lottery and can be condensed into its expected values for both miners. This is the value miners base their decisions on and the value that appears in the payoff-matrices in later sections.

Stages	$\frac{1}{2}\vartheta_T(2)$	$\xi_T(2)$
$T=1$	1	2
$T=2$	8	8
$T=3$	144	48
$T=4$	4'608	384
$T=5$	230'400	3'840
$T=6$	33'177'600	46'080
$T=7$	3'251'404'800	645'120
$\vdots$	$\vdots$	$\vdots$

Table I – Number of terminal nodes after T stages $\frac{1}{2}\vartheta_T(n)$ and the number of unique blockchains $\xi_T(n)$ for a blockchain after a game with T stages and $n=2$ miners.

Note that $\vartheta_T(n)$ only refers to the number of intermediate nodes in the game tree after T intermediate stages, not to the number of possible blockchains at the end of the game. Let us take a look how we arrive at that number, $\xi_T(n)$, which behaves a little more tame than $\vartheta_T(n)$. We proceed as before and consider a game with T stages and n miners. The number of unique blockchains at the end of the game $\xi_T(n)$ is equal to the number of unique blockchains after the previous stage $\xi_{T-1}(n)$ multiplied by the factor $T \cdot n$. Why? For every unique blockchain after the previous stage $T-1$ we can append a child block at T different parent blocks and have n possible winners.

After the very first stage there are n possible blockchains, which all have two blocks (b_0 and b_1) and only differ in the winner of b_1. We can now define $\xi_T(n)$

recursively.

$$\xi_1(n) = n \qquad \text{and} \qquad \xi_t(n) = \xi_{t-1}(n) \ \cdot t \cdot n$$

This can be rewritten as:

$$\xi_T(n) = \prod_{t=1}^{T} (t \cdot n) \ = \ n^T \cdot T!$$

Why are we interested in ϑ_T and ξ_T? We want to know how far we can go with our computational analysis, how many stages the game can have before the calculation becomes prohibitively expensive. The purpose of Table I is to show that the game explodes extraordinarily rapidly in complexity. We are especially interested in $\frac{1}{2}\vartheta_T$ because this is equivalent to the number of paths a computer program has to check to test the conjectures in section 6. Our limiting factor for the computational analysis is computing power. More about this can be found in section 6.6.

The functions ϑ_T and ξ_T are related. It holds that $\vartheta_T \geq \xi_T$ for any number of miners n and any number of stages T. Why is ϑ_T larger? There are multiple intermediate nodes that are associated with identical blockchains. That is, there exist some pairs of intermediate nodes after T stages θ, θ' with $\theta \neq \theta'$ in the game tree which result in the same blockchain $\mathbb{B}_2$. To see this, please take a look at example 1.

Before we turn our attention to the example, we first have to explain what we mean when we talk about strategy profiles in finite blockchain games. Let us consider a finite blockchain game with T stages and $n = 2$ miners. A strategy profile s_t in pure strategies for any given intermediate stage t has the following form:

$$s_t = (\widehat{b}_{t-1}(0), \widehat{b}_{t-1}(1))$$

Example 1 Consider a two-stage game with two miners, where miner 1 won the first stage. In the second stage, the strategy profiles (b_1, b_0) and (b_1, b_1) result in the same blockchain $\mathbb{B}_2$ when miner 1 wins the second stage. In both cases $\mathbb{B}_2$ consists of a sequence of blocks $\{b_0, b_1, b_2\}$ with $b_0 \Leftarrow b_1$ and $b_1 \Leftarrow b_2$ and $\iota(b_1) = \iota(b_2) = 1$. This is possible because miner 2's strategy has no effect on the blockchain when they lose the stage. They do not get to append a block of their own. We have two intermediate nodes after 2 stages, each with their unique path through the game tree, that result in the same blockchain that does not have any forks and where miner 1 won both blocks b_1 and b_2.

4 Short Games

4.1 Two-Stage Game

When we say T-stage game, we mean a finite blockchain game with T stages. A two-stage game ends after $T = 2$ stages.

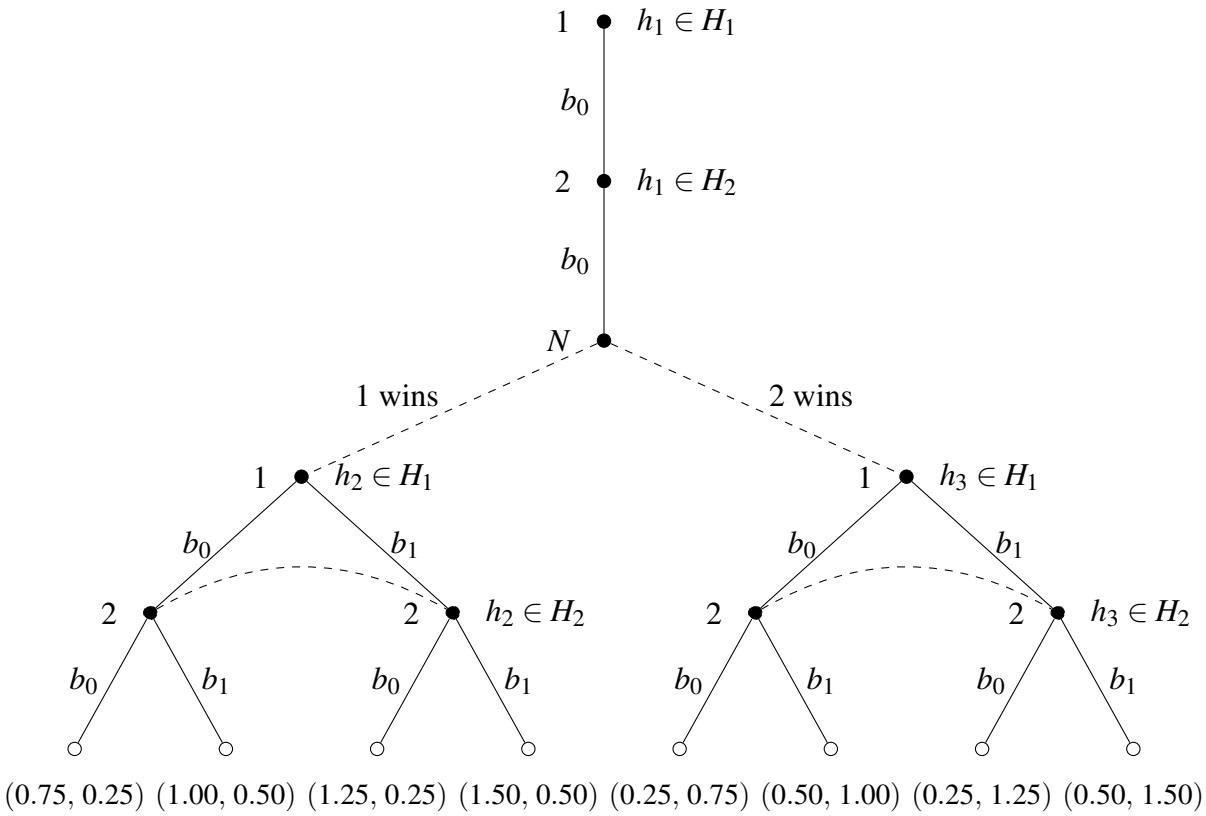

Figure 2 – The game-tree for the two-stage game.

4.1.1 Payoffs

We often talk about how the miners maximize their payoffs, this means that they are maximizing their *expected* payoff. Finite blockchain games are inherently uncertain and payoffs therefore only exist in expectation. It is important to note that this payoff-maximizing behaviour automatically results in a subgame-perfect equilibrium, as we will see later.

Consider a finite blockchain game with T stages and $n = 2$ miners. For any T, there exists only one strategy in the first stage $t = 1$ (mine on b_0). Hence, there exists only one strategy profile that can be played, which is (b_0, b_0).

We can assume w.l.o.g. that miner 1 wins the first stage. If miner 1 does not win the first stage, we may swap the indices of the miners. Hence, there is only one possible blockchain in stage $t = 2$ (Figure 3). This is equivalent to only looking

at the left side of the game trees in Figure 1 or Figure 2.

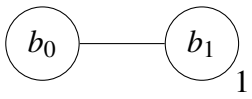

Figure 3 – The only possible the blockchain after the first stage of a finite blockchain game with $n = 2$ miners. We assume w.l.o.g. that miner 1 wins the first stage.

Now consider the final stage of the two-stage game. Therefore, let $T = 2$ and $t = 2$. The Nash equilibrium strategy profile in the last stage of the game is given by (b_1, b_1). This is a strictly dominant strategy equilibrium. To show how this Nash equilibrium is found, we proceed by explaining how the payoff-matrix (Table II) is constructed. All payoffs were mathematically rounded to two decimal places and the Nash equilibria are marked with round brackets, this is true for all payoff-matrices herein.

		Miner 2	
		b_0	b_1
Miner 1	b_0	0.75, 0.25	1.00, 0.50
	b_1	1.25, 0.25	(1.50, 0.50)

Table II – Payoff matrix in the final stage of a two-stage game. The Nash equilibrium is marked with round brackets.

Assume that miner 1 decides to mine on b_1 while miner 2 decides to mine on b_0. In this case we are faced with two possibilities. Either miner 1 wins, resulting in the blockchain (a), or miner 2 wins, resulting in the blockchain (d) at the end of the game. Figure 4 shows the blockchains (a) and (d). We are looking at the bottom left quadrant of the payoff-matrix (Table II). miner 1 receives a payoff

of 2.00 in blockchain (a), and a payoff of

$$\frac{1}{l} \cdot 1.00 + \frac{1}{l} \cdot 0.00 = 0.50$$

in blockchain (d), where $l = 2$ is the number of the longest chains in that blockchain. Recall that in the case where there are multiple longest chains at the end of the game, one is chosen with equal probability $\frac{1}{l}$. At the end of the game, miner 1's expected payoff is

$$\frac{1}{n} \cdot 2.00 + \frac{1}{n} \cdot 0.50 = 1.25$$

Recall that $\frac{1}{n}$ is the probability of winning the stage, since each miner wins with equal probability. Miner 2 receives a payoff of Zero in blockchain (a), and a payoff of 0.50 in blockchain (d). At the end of the game, miner 2's expected payoff therefore is 0.25. The other entries of the payoff-matrix at the end of the two-stage game (Table II) are computed completely analogously. Their derivation is hence omitted.

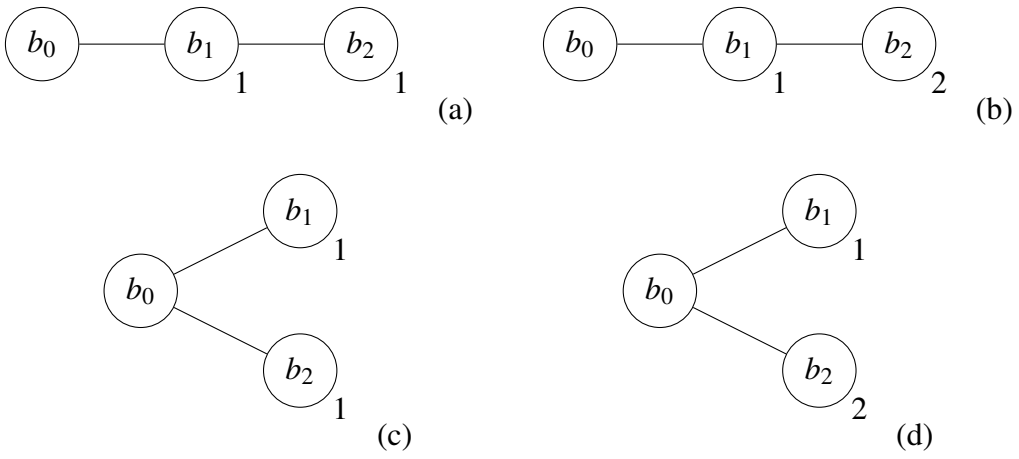

Figure 4 – Four possible situations after the second stage of a finite blockchain game with $n = 2$ miners.

4.1.2 Subgame-Perfect Equilibrium

The concept of subgame-perfection as defined by Selten [10] builds on top of the concept of a Nash equilibrium and is invaluable for our analysis. In short, a Nash equilibrium is subgame-perfect if it constitutes a Nash equilibrium in every subgame.

After the last nodes in the game tree, Nature makes at least one more move, deciding the winner of the second stage as well as deciding which chain is chosen if there is more than one longest chain. However, these steps are omitted from the tree because they only make the tree (Figure 2) unnecessarily complicated. The payoffs at the terminal nodes represents the expected value of Nature's subsequent decisions. In this way the payoffs in the tree correspond to the entries of the payoff-matrix (Table II). We have $\frac{1}{2}\vartheta_2(2) = 8$ terminal nodes in this tree. Note how the left and the right side of the tree only differ in the reversal of the payoff tuples. That is the reason we can assume w.l.o.g. that miner 1 wins the first stage.

Now we have to talk about information sets and behavioural strategies. In the two-stage game there are six information sets, three for each player. We denote an information set by $h_j \in H_i$ where H_i is the set of all information sets for miner i. All information sets are visible in Figure 2. Please note that we abuse the notation slightly and take advantage of the fact that the game is symmetrical. The information sets for both miners are equivalent, because they move simultaneously and only differ in their index, hence $H_1 = H_2$. This is why we do not differentiate between $h_1 \in H_1$ and $h_1 \in H_2$, they are the equivalent.

Next, we want to characterize the subgame-perfect equilibrium by writing down the subgame-perfect behavioural strategy profiles s_i^* for both players $i = 1, 2$. A behavioural strategy profile specifies an independent probability distribution over a player's actions in every information set $h_j \in H_i$. In our case these probability distributions are always degenerate along the subgame-perfect equilibrium

path, i.e. players never mix between actions in any information set (at least for games with less than seven stages). Consequently, to keep things tidy, we only write down the strategy which is chosen with a probability of One, e.g. we write $s_i^*(h_j) = b_t$ where b_t is miner i's chosen strategy in information set h_j.
The subgame-perfect Nash equilibrium is

$$\begin{aligned} s^* &= \{s_1^*(H_1),\, s_2^*(H_2)\} \\ &= \{\big(s_1^*(h_1),\, s_1^*(h_2),\, s_1^*(h_3)\big),\ \big(s_2^*(h_1),\, s_2^*(h_2),\, s_2^*(h_3)\big)\} \\ &= \{(b_0,\, b_1,\, b_1),\ (b_0,\, b_1,\, b_1)\} \end{aligned}$$

where both players play b_0 in the first stage and b_1 in the second (no matter who wins the first stage).
Remember that the individual stages are games of imperfect information. In the second stage this means that miner 2 cannot make their strategy dependent on miner 1's strategy *in the same stage* (and vice versa), because both players move simultaneously.
The payoff-maximizing strategy is straight forward in the two-stage game. All miners mine on b_0 in the first stage and on b_1 in the second stage. Nonetheless, the path through the game-tree is stochastic and depends on Nature's decisions. Since the game explodes in complexity very quickly, writing down the subgame-perfect Nash equilibrium like this becomes impractical for the four-stage game at the latest. Conveniently, saying that the strategy profile (b_{t-1}, b_{t-1}) is played in every stage t (regardless of Nature's decisions) is equivalent. It is equivalent because s^* says exactly that, that (b_0, b_0) is played in the first stage and that (b_1, b_1) is played in the second stage, no matter what.
Hence, for the sake of simplicity, we characterize the subgame-perfect Nash equilibrium by saying that each player mines on the most recent block, i.e. that subgame-perfect behavioural strategy profiles for every stage t are given by (b_{t-1}, b_{t-1}). As we will see, this subgame-perfect equilibrium is *unique* for

games with less than $T = 7$ stages, meaning that the game will always follow one of them when both players maximize their payoffs. We do not know about games with $T \geq 7$ stages yet, although we strongly suspect that the pattern will continue forever.

To end this section we would like to address a concern some astute readers may have. In games of imperfect information, backwards induction may not always result in a subgame-perfect equilibrium in *pure strategies*. **[citation needed]** We are lucky however, because in any given (on-path) situation we look at there always exists a Nash equilibrium in pure strategies. Not only that, but the equilibrium in any given (on-path) situation is also always one in strictly dominant strategies, i.e. it is unique. The game is well-behaved in that way. However, off-path there exist situations where there is no strictly dominant strategy. Please have a look at the payoff-matrix in Table V for such an off-path situation in the three-stage game. We will inspect that table more closely in the next section 4.2.

4.2 Three-Stage Game

Next, we turn our attention to a three-stage game. We show how to derive the payoff-matrix in intermediate stage $t = 2$ by the means of backwards induction. Assume again w.l.o.g. that miner 1 wins the first stage.

4.2.1 Final Stage

Let $T = 3$ and $t = 3$. Figure 4 shows all possible the blockchain in stage $t = 3$. Some, as we will see, can never be reached if both players maximize their payoffs. When the Nash equilibrium was played in stage $t = 2$, only blockchains (a) and (b) are possible.

Tables III, IV, V and VI show the payoff-matrices for the cases (a), (b), (c) and (d) respectively. The entries of the payoff-matrices are calculated analogously to the previous chapter. The only difference is, that we now have three instead

of two strategies for each miner.

		Miner 2		
		b_0	b_1	b_2
Miner 1	b_0	2.00, 0.00	1.75, 0.25	2.00, 0.50
	b_1	2.00, 0.00	1.75, 0.25	2.00, 0.50
	b_2	2.50, 0.00	2.25, 0.25	(2.50, 0.50)

Table III – Payoff-matrix for blockchain (a) in the three-stage game.

		Miner 2		
		b_0	b_1	b_2
Miner 1	b_0	1.00, 1.00	1.00, 1.00	1.00, 1.50
	b_1	1.25, 0.75	1.25, 0.75	1.25, 1.25
	b_2	1.50, 1.00	1.50, 1.00	(1.50, 1.50)

Table IV – Payoff-matrix for blockchain (b) in the three-stage game.

		Miner 2		
		b_0	b_1	b_2
Miner 1	b_0	0.83, 0.17	0.83, 0.50	0.83, 0.50
	b_1	1.33, 0.17	(1.50, 0.50)	(1.50, 0.50)
	b_2	1.33, 0.17	(1.50, 0.50)	(1.50, 0.50)

Table V – Payoff-matrix for blockchain (c) in the three-stage game.

		Miner 2		
		b_0	b_1	b_2
Miner 1	b_0	0.50, 0.50	0.83, 0.83	0.33, 1.17
	b_1	1.17, 0.33	1.50, 0.50	(1.00, 1.00)
	b_2	0.67, 0.83	1.00, 1.00	0.50, 1.50

Table VI – Payoff-matrix for blockchain (d) in the three-stage game.

At this point we want to take a closer look at Table V. What is interesting about this table is the fact that there is no Nash equilibrium in strictly dominant strategies. Once we consider blockchain (c) in Figure 4 however, we can see why. Case (c) represents a symmetrical blockchain where miner 1 won both the first and second stage, and appended both blocks to the genesis block b_0. Both miners have to be indifferent between mining on b_1 or b_2, because the situation is symmetrical.

4.2.2 Second Stage

Let $T = 3$ and $t = 2$. Much like the two-stage game, the unique Nash equilibrium is given by (b_1, b_1). To show how this Nash equilibrium is found, we proceed by explaining how to use backwards induction to calculate the entries in the payoff-matrix in the second stage (Table VII).

		Miner 2	
		b_0	b_1
Miner 1	b_0	1.25, 0.75	1.50, 1.00
	b_1	1.75, 0.75	(2.00, 1.00)

Table VII – Payoff-matrix in intermediate stage $t = 2$ in the three-stage game.

Assume that miner 1 decides to mine on b_0 while miner 2 decides to mine on b_1. In this case we are faced with two possibilities. Either miner 1 wins, resulting in the blockchain (c), or miner 2 wins, resulting in the blockchain (b) in the last stage of the game (stage $t = 3$). We are looking at the top right quadrant of the payoff-matrix (Table VII).

From our analysis of stage $t = 3$ we know what the payoffs will be when the game reaches that stage. We can assume that one of the Nash equilibria will be played. If miner 1 wins second stage, then the miner's payoff in the third stage are given by one of the Nash equilibrium payoffs in Table V. Conveniently, they are all the same, which means that no matter which Nash equilibrium is played, miner 1 and miner 2 receive expected payoffs of 1.50 and 0.50 respectively. If miner 2 wins the second stage, then the miner's payoffs in the third stage are given by the unique Nash equilibrium payoffs in Table IV. In this case miner 1 and miner 2 both receive an expected payoff of 1.50.

We can now compute the expected payoffs in the second stage. miner 1 receives 1.50 in any case and miner 2 receives 1.50 or 0.50 each with probability $\frac{1}{n}$. We yield payoffs of 1.50 for miner 1 and 1.00 for miner 2 in stage $t = 2$ when miner 1 mines on b_0 and miner 2 mines on b_1. All other entries of the payoff-matrix are calculated completely analogously and their calculation is hence omitted.

Finding the subgame-perfect equilibrium path is straight forward. In every stage we know what each miner will do. Hence, we can simulate a playthrough stage

by stage. In stage $t = 1$ both miners mine on b_0 and in stage $t = 2$ both miners mine on b_1 (which can be seen in Table VII). In the third stage we are either in blockchains (a) or (b). In both cases, both miners will mine on b_2 (which can be seen in Tables III and IV). Therefore, there is only one subgame-perfect equilibrium path resulting in one chain without any forks. The strategy profiles in every stage are given by (b_{t-1}, b_{t-1}) along the subgame-perfect equilibrium path.

4.3 Four-Stage Game

The analysis of the four-stage game is done analogously to the analysis of the two- and three-stage games. Once again, we assume w.l.o.g. that miner 1 wins the first stage.

4.3.1 Final Stage

There are $\frac{1}{2}\xi_3(2) = 24$ different possible blockchains after stage $t = 3$, six for each of the four possible blockchains (a), (b), (c) and (d). In total there are $\xi_3(2) = 48$, but we assumed that miner 1 wins the first stage, hence we only look at half of them (the left side of the game tree).
Please be aware that some blockchains are strategically equivalent and only differ in the order the blocks were mined in, i.e. they are the same except for the block indices. In Figure 5 below and in Figure A.1 in the Appendix we can see some blockchains in the final stage of the four-stage game.

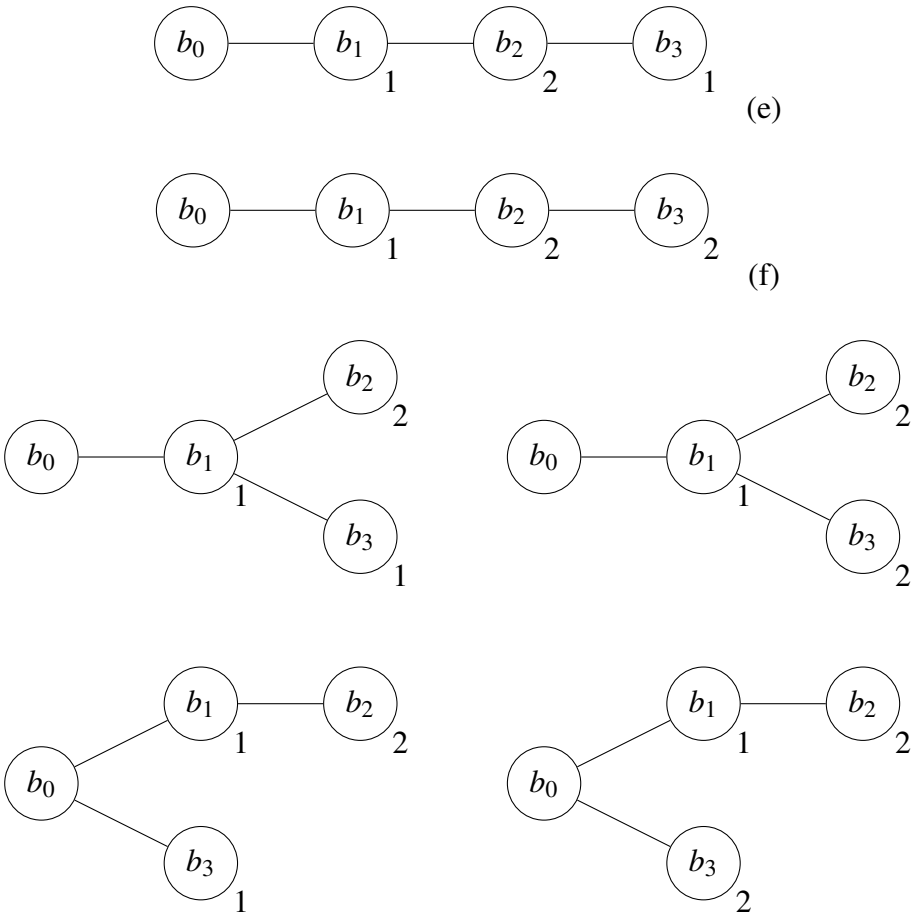

Figure 5 – Six possible situations after the third stage, when case (b) materialized after the second stage.

At this point we want to show one example of a payoff-matrix in the last stage of the four-stage game. The payoff matrix below (Table VIII) is on the subgame-perfect equilibrium path, as we will see later. There is only one Nash equilibrium and the entries are calculated in the same way as in previous sections. The matrix was generated using a computer program and then checked manually for errors.

Note that such a payoff-matrix exists for all 24 blockchains in stage $t = 4$. Doing the analysis by hand, while possible, is impractical. The game explodes rapidly in complexity.

		Miner 2			
		b_0	b_1	b_2	b_3
Miner 1	b_0	2.00, 1.00	2.00, 1.00	1.75, 1.25	2.00, 1.50
	b_1	2.00, 1.00	2.00, 1.00	1.75, 1.25	2.00, 1.50
	b_2	2.00, 1.00	2.00, 1.00	1.75, 1.25	2.00, 1.50
	b_3	2.50, 1.00	2.50, 1.00	2.25, 1.25	(2.50, 1.50)

Table VIII – Payoff-matrix for blockchain (e) in the four-stage game.

		Miner 2			
		b_0	b_1	b_2	b_3
Miner 1	b_0	1.00, 2.00	1.00, 2.00	1.00, 2.00	1.00, 2.50
	b_1	1.00, 2.00	1.00, 2.00	1.00, 2.00	1.00, 2.50
	b_2	1.25, 1.75	1.25, 1.75	1.25, 1.75	1.25, 2.25
	b_3	1.50, 2.00	1.50, 2.00	1.50, 2.00	(1.50, 2.50)

Table IX – Payoff-matrix for blockchain (f) in the four-stage game.

4.3.2 Third Stage

The payoff-matrix below (Table X) was generated using a computer program. Let's take a closer look at the Nash equilibrium (b_2, b_2). Both miners mine on the most recent block, resulting in either blockchain (e) or blockchain (f). In case (e) the expected payoffs in the next stage will be 2.50 for miner 1 and 1.50 for miner 2 (as seen in the Nash equilibrium in Table VIII). In case (f) the payoffs are reversed, 1.50 for miner 1 and 2.50 for miner 2 (Table IX). Recall that (e) and (f) both materialize with probability $\frac{1}{n}$, hence the expected payoff in

the third stage when mining at b_2 is 2.00 for both players.

		Miner 2		
		b_0	b_1	b_2
Miner 1	b_0	1.50, 1.50	1.50, 1.50	1.50, 2.00
	b_1	1.75, 1.25	1.75, 1.25	1.75, 1.75
	b_2	2.00, 1.50	2.00, 1.50	(2.00, 2.00)

Table X – Payoff-matrix for blockchain (b) in the four-stage game.

Alternatively, blockchain (a) may materialize instead of (b). In that case we have two different chains (g) and (h) in the final stage. The analysis is done in the same way. All the payoff-matrices are listed in the Appendix (Tables A.1, A.2, and A.3).

4.3.3 Second Stage

The payoff-matrix below (Table XI) was generated using a computer program. We arrive at it by applying backwards induction multiple times until we arrive at the payoff-matrix for the second stage. As we can see, there is a Nash equilibrium in strictly dominant strategies (b_1, b_1).

After the second stage we have a situation with either blockchain (a) or (b). Then the game may end up in (e) or (f) when (b) materializes, or (g) or (h) when (a) materializes. The payoffs will be different depending on which blockchains the game goes through. As explained in previous sections, we take the expected value, which can be easily computed because all probabilities are fixed and known. The payoffs for miner 1 are either 3.00 in (a) or 2.00 in (b), and the payoffs for miner 2 are either 1.00 in (a) or 2.00 in (b). Hence, we end up with expected payoffs of 2.50 and 1.50 in the second stage Nash equilibrium (Table

XI). If the reader wishes to go through the calculations themselves, the payoff-matrices for blockchains (a), (g), and (h) in the four-stage game are listed in the Appendix (Figure A.1 and Tables A.1, A.2, and A.3).

		Miner 2	
		b_0	b_1
Miner 1	b_0	1.69, 1.19	2.00, 1.50
	b_1	2.19, 1.19	(2.50, 1.50)

Table XI – Payoff-matrix in intermediate stage $t = 2$ in the four-stage game.

The subgame-perfect equilibrium path through the four-stage game is found completely analogously to the path in the three-stage game. We once again start in the first stage and simulate a playthrough while assuming that the Nash equilibrium is played in every stage. Conveniently, no situations with multiple Nash equilibria will be reached along the path of play, meaning that we once again end up with only one chain without any forks.

5 Long Games

To start, let us recapitulate. The subgame-perfect equilibrium paths through games with $T \leq 4$ stages are all similar. In the first stage of the two-stage game, the players play (b_0, b_0) and in the second stage they play (b_1, b_1). The three- and four-stage games behave in the same way. In every intermediate stage t the strategy profile (b_{t-1}, b_{t-1}) is played. Miners always mine on the most recent block.

We used a computer program to verify that games with $T = 5, 6$ and 7 stages follow the same pattern. The time it took for the program to terminate give some clue about how computationally intensive the task can get. Refer to Table

XII for an overview. An average desktop PC with an 8th generation Intel i7 Processor was used for the computation.

Stages	Time to Termination
$T \leq 4$	$<$ 1 second
$T = 5$	$\approx$ 18 seconds
$T = 6$	$\approx$ 23 minutes
$T = 7$	$\approx$ 39 hours

Table XII – The time it took to verify that the equilibrium path in the $n = 2$ player game never results in forks.

We proved computationally that all games with $T \leq 7$ stages result in a single chain without any forks. This contrasts the findings of Biais et al. [8] who prove that there exist equilibria that exhibit forks (in their specific model).

If Ewerhart's model can exhibit equilibria with forks for games with $T > 7$ stages remains to be seen. The aid of a more powerful computer or a faster programming language is unlikely to help. It may enable us to consider two or three additional stages, but that is not useful.

Please refer to Appendix A.2 for more details about our code. It is freely available on GitHub, the project's link can be found in the Appendix.

6 Conjectures

We tried, but failed to construct formal proofs of the conjectures in this section. However, as we will see, we verified all conjectures to be true for finite blockchain games with $T \leq 7$ stages and $n = 2$ miners. We will expand on our findings in section 6.6.

6.1 Subgame-Perfect Equilibrium

Conjecture 1 The strategy profile (b_{t-1}, b_{t-1}) is played in every stage in the subgame-perfect equilibrium of a finite blockchain game with any number of stages T and $n = 2$ miners. Consequently, equilibria in randomized actions need not be considered along the subgame-perfect equilibrium path, and the subgame-perfect equilibrium path is unique.

Conjecture 1 is True if and only if there are no forks along the subgame-perfect equilibrium path, and vice versa. To see this, please consider Lemma 1.

Lemma 1 If the strategy profile (b_{t-1}, b_{t-1}) is played in every stage of a finite blockchain game with T stages and $n = 2$ miners, then the resulting blockchain $\mathbb{B}_T$ at the end of the game does not exhibit any forks. If the blockchain $\mathbb{B}_T$ does not exhibit any forks, then there cannot have been any stage where a strategy profile different from (b_{t-1}, b_{t-1}) was played.

Proof. We begin by proving the first part of Lemma 1. We assume that players play (b_{t-1}, b_{t-1}) in every stage t. After the first stage we will have a sequence of blocks $B_1 = \{b_0, b_1\}$ with $b_0 \Leftarrow b_1$, because (b_0, b_0) was played in stage $t = 1$. After the second stage we will have a sequence of blocks $B_2 = \{b_0, b_1, b_2\}$ with $b_0 \Leftarrow b_1$ and $b_1 \Leftarrow b_2$, because (b_1, b_1) was played in the stage $t = 2$. We can repeat this argument until we reach stage $t = T$, the end of the game. Therefore, this pattern must continue for all remaining stages $t \in \{3, \ldots, T\}$.
Next we prove the second part of Lemma 1. We assume that $\mathbb{B}_T$ does not exhibit any forks. This means that every block except block b_T has exactly one child block and that no child was ever appended to a parent that already had a child. In the final stage $t = T$ the strategy profile (b_{T-1}, b_{T-1}) must have been played, because b_{t-1} is the only block without a child in stage $t = T$. Furthermore, in stage $t = T - 1$ the strategy profile (b_{T-2}, b_{T-2}) must have been played, because b_{T-2} was the only block without a child in stage $t = T - 1$. We can repeat this argument until we reach stage $t = 1$, the beginning of the game. Therefore, this

pattern must continue for all other stages $t \in \{1, \ldots, T-2\}$. This concludes the proof. □

6.2 Monotonicity

Monotonicity as defined by Kroll et al. [9] is used to describe the behaviour of blockchain mining strategies. Consider an arbitrary situation where miner i's strategy in stage t is to mine on block b. Miner i's strategy is monotonic if and only if miner i always mines on b' in the next stage $t+1$ as long as $b \Leftarrow b'$. In other words, if a monotonic strategy tries to extend the blockchain at some block b, then the addition of a new block b' on b will cause it to mine on b' instead.

Conjecture 2 The payoff-maximizing strategy in a finite blockchain game with $n = 2$ payoff-maximizing miners is monotonic for any horizon T.

Proving conjecture 2 would be very useful because it is a strong result. We argue that some valuable insights follow from it. We know that conjecture 2 implies conjecture 1. To see this, please consider Lemma 2.

Lemma 2 Conjecture 2 implies conjecture 1. In other words, if all players behave consistent with monotonicity, then the strategy profile (b_{t-1}, b_{t-1}) will be played in every stage t.

Proof. This follows directly from the definition of monotonicity. Assume that all players behave monotonically. We know that the strategy profile (b_0, b_0) is always played in the stage $t = 1$, regardless of player's strategies. Now we have a situation where both miners sought to extend the blockchain at b_0 in stage $t = 1$, and b_1 was appended to b_0 in that stage, i.e. $b_0 \Leftarrow b_1$. Because all miners behave monotonically, their strategy is now to mine on b_1, i.e. the strategy profile (b_1, b_1) is played in stage $t = 2$. This argument can be repeated for all remaining stage $t \in \{3, \ldots, T\}$. Therefore, the strategy profile (b_{t-1}, b_{t-1}) is played in every stage. □

Conjecture 2 sounds intuitive, but writing a formal proof is hard. The complexity stems from the fact that strategies in the subgame-perfect equilibrium are interdependent. Writing a proof for monotonicity thus requires showing that behaving monotonically is mutually optimal for all miners in any situation. We suspect that we do not know enough about the game to prove that yet.

6.3 A Miner's Behavior after Winning their First Block

Conjecture 3 Consider a finite blockchain game with T stages and $n = 2$ payoff-maximizing miners. Assume that miner i has not won any blocks before stage $t \in \{1, 2, \ldots, T-1\}$. Under the condition that miner i *maximizes their payoff and wins their first block in stage* t it is optimal for miner i to continue mining on the same branch for the rest of the game.

Note that conjecture 3 does not apply if miner i does not maximize their payoff in stage t. One can construct an example where miner i wins their first block in stage t, in which it is not optimal for them to keep mining on the same branch in the subsequent stage $t+1$ when they do not maximize their payoffs in stage t. Additionally, conjecture 3 may follow from conjecture 2 (monotonicity), but we were unable to construct a formal proof.

6.4 Switching to Shorter Chains

Conjecture 4 Consider a finite blockchain game with T stages and $n = 2$ payoff-maximizing miners. In any intermediate stage $t \in \{1, 2, \ldots, T-1\}$ miner i mines on some branch C, seeking to extend it. If miner i switches to a different branch C' in the next stage $t+1$, then that branch will never be shorter than the one they are switching from, i.e. C' will never be shorter than C.

6.5 Coordination

Conjecture 5 In a finite blockchain game with T stages and $n = 2$ payoff-maximizing miners, coordination between the miners is never necessary to play a Nash equilibrium in any stage (on and off the subgame-perfect equilibrium path).

		Miner 2	
		b_0	b_1
Miner 1	b_0	$(\pi_1,\ \pi_2)$	$\pi_1',\ \pi_2'$
	b_1	$\pi_1',\ \pi_2'$	$(\pi_1,\ \pi_2)$

Table XIII – A payoff-matrix where coordination between the players is necessary to play one of the Nash equilibria. A situation like this can never happen as long as conjecture 5 holds.

Consider Table XIII and assume that $\pi_1 > \pi_1'$ and $\pi_2 > \pi_2'$. This represents a situation akin to the classic game known as *Battle of the Sexes*, which was introduced by Luce and Raiffa in 1957 [11]. There are two Nash equilibria, (b_0, b_0) and (b_1, b_1), which can only be played if the players coordinate their actions (consciously or by random chance). Ewerhart's model assumes coordination to be impossible because the miners move simultaneously. This can lead to a problem: What if the players play a strategy profile that is not a Nash equilibrium? To avoid this problem, we always assumed that players only ever play a Nash equilibrium in any given stage, but this assumption is unnecessary as long as conjecture 5 holds. Indeed, in practice we never found any situation where coordination was necessary to play a Nash equilibrium. When we encounter a situation with multiple Nash equilibria, the miners are always indifferent between each equilibrium. An example of such a situation is shown in Table V. All equilibria in that payoff-matrix yield identical payoffs for both miners. The game is well-behaved in this way.

6.6 Insights from Computer Simulations

Using computer simulations we were able to verify that all conjectures mentioned in this report (conjectures 1, 2, 3, 4 and 5) are True for finite blockchain games with $T \leq 7$ stages.

On an average desktop computer with an 8th generation Intel i7 processor it took on the order of 48 hours to run each test, but they can all be run simultaneously because each runs on one core only. Significant speed-ups are possible, but unlikely to be of much use. Our Python implementation makes use of recursion, which is slow, but elegant. We require a speed-up of approximately 70 to 100 times to analyze an additional stage (this requirement gets larger with every additional stage). This means that if we were somehow able to achieve a speed-up of 100'000 times (by using more powerful computers or a faster programming language), we would only be able to look at two or three additional stages, i.e. to look at games with $T \leq 9$ stages. More details about our Python code can be found in Appendix A.2.

7 Conclusion

We limited our analysis to finite blockchain games with $n = 2$ miners and investigated five conjectures, which proved to be True for games with $T \leq 7$ stages. In short, we found that the subgame-perfect equilibrium both players play (b_{t-1}, b_{t-1}) in every stage, which leads to only one chain without forks at the end of the game. This symmetric equilibrium is stable (i.e. no player has an incentive to deviate at any stage). It even represents an equilibrium in strictly dominant strategies and is therefore unique. Additionally, we found that the payoff-maximizing strategy is monotonic, that a miner will continue mining on the same branch they won their first block on for the rest of the game, that a miner will never switch to a shorter chain than the one they are already mining

on, and that coordination between the miners is never necessary to play a Nash equilibrium in any given stage. We were only able to confirm these conjectures for games with less than seven stages because of the increased complexity every additional stage adds. To quantify this increase in complexity, we introduced the functions $\vartheta_T(n)$ and $\xi_T(n)$, which count the number of intermediate nodes in the game tree and the number of possible blockchains respectively.

Even tough we are looking at a complex multistage game, the subgame-perfect equilibrium strategies for both players are straight forward: In the subgame-perfect equilibrium they always mine on the most recent block. In this way, Ewerhart's model can show how a small set of rules can generate stable consensus in blockchains. The equilibria also make sense and are welfare-maximizing. No payoffs are lost because of orphaned blocks or branches.

Given that the computer simulations confirm all our conjectures for $T \leq 7$, it might be worth investigating them formally, i.e. to try to come up with formal proofs. The value of our computer based analysis lies in the fact that we now know that there are no "simple" counter-examples for the conjectures to be found. The conjectures are not easily disproven.

8 References

[1] C. Ewerhart, "Finite blockchain games," *Economics Letters*, vol. 197, p. 109 614, 2020.

[2] A. T. Sherman, F. Javani, H. Zhang, and E. Golaszewski, "On the origins and variations of blockchain technologies," *IEEE Security & Privacy*, vol. 17, no. 1, pp. 72–77, 2019.

[3] D. L. Chaum, *Computer Systems established, maintained and trusted by mutually suspicious groups*. Electronics Research Laboratory, University of California, 1979.

[4] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” *Decentralized business review*, p. 21 260, 2008.

[5] J. F. Nash Jr, “Equilibrium points in n-person games,” *Proceedings of the national academy of sciences*, vol. 36, no. 1, pp. 48–49, 1950.

[6] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan, “On the instability of bitcoin without the block reward,” in *Proceedings of the 2016 acm sigsac conference on computer and communications security*, 2016, pp. 154–167.

[7] I. Eyal and E. G. Sirer, “Majority is not enough: Bitcoin mining is vulnerable,” *Communications of the ACM*, vol. 61, no. 7, pp. 95–102, 2018.

[8] B. Biais, C. Bisiere, M. Bouvard, and C. Casamatta, “The blockchain folk theorem,” *The Review of Financial Studies*, vol. 32, no. 5, pp. 1662–1715, 2019.

[9] J. A. Kroll, I. C. Davey, and E. W. Felten, “The economics of bitcoin mining, or bitcoin in the presence of adversaries,” in *Proceedings of WEIS*, Citeseer, vol. 2013, 2013.

[10] R. Selten, “Spieltheoretische behandlung eines oligopolmodells mit nachfragetraegheit. teil ii: Eigenschaften des dynamischen preisgleichgewichts,” *Zeitschrift für die gesamte Staatswissenschaft/Journal of Institutional and Theoretical Economics*, no. H. 4, pp. 667–689, 1965.

[11] R. D. Luce and H. Raiffa, “Games and decisions: Introduction and critical survey,” 1957.

A Appendix

A.1 More About the Four-Stage Game

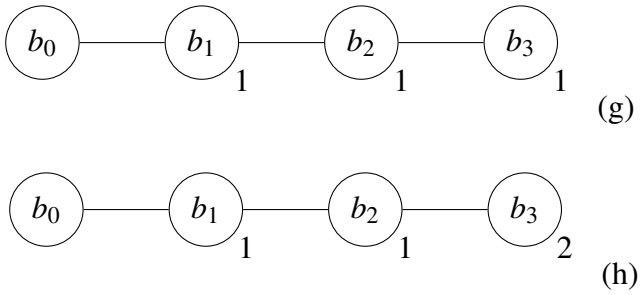

Figure A.1 – Six possible situations after the third stage, when case (a) materialized after the second stage.

		Miner 2		
		b_0	b_1	b_2
Miner 1	b_0	2.38, 0.50	2.25, 0.75	2.50, 1.00
	b_1	2.38, 0.50	2.25, 0.75	2.50, 1.00
	b_2	2.88, 0.50	2.75, 0.75	(3.00, 1.00)

Table A.1 – Payoff-matrix for blockchain (a) in the four-stage game.

		Miner 2			
		b_0	b_1	b_2	b_3
Miner 1	b_0	3.00, 0.00	3.00, 0.00	2.75, 0.25	3.00, 0.50
	b_1	3.00, 0.00	3.00, 0.00	2.75, 0.25	3.00, 0.50
	b_2	3.00, 0.00	3.00, 0.00	2.75, 0.25	3.00, 0.50
	b_3	3.50, 0.00	3.50, 0.00	3.25, 0.25	(3.50, 0.50)

Table A.2 – Payoff-matrix for blockchain (g) in the four-stage game.

		Miner 2			
		b_0	b_1	b_2	b_3
Miner 1	b_0	2.00, 1.00	2.00, 1.00	2.00, 1.00	2.00, 1.50
	b_1	2.00, 1.00	2.00, 1.00	2.00, 1.00	2.00, 1.50
	b_2	2.25, 0.75	2.25, 0.75	2.25, 0.75	2.25, 1.25
	b_3	2.50, 1.00	2.50, 1.00	2.50, 1.00	(2.50, 1.50)

Table A.3 – Payoff-matrix for blockchain (h) in the four-stage game.

A.2 Codebase

As mentioned before, we used Python to conduct some of our analysis. The code was used to verify the non-existence of forks in the subgame-perfect equilibrium in finite blockchain games with $T \leq 7$ stages and $n = 2$ miners. It was also used to test the conjectures in section 6 and to generate some payoff-matrices throughout this document. Jupyter Notebooks enabled us to add further explanations to our code, in an effort to make it more comprehensible and accessible to all readers. The notebooks are freely available on GitHub

at `https://github.com/koeztu/blockchain-game-analysis`.

A.3 Payoff Matrices in the Second Stage

For the particularly interested reader we include a collection of payoff-matrices in the second stage. The time it took to generate each matrix is roughly equal to the entries in Table XII (same order of magnitude).

		Miner 2	
		b_0	b_1
Miner 1	b_0	2.19, 1.69	2.50, 2.00
	b_1	2.69, 1.69	(3.00, 2.00)

Table A.4 – Payoff-matrix in intermediate stage $t = 2$ in the five-stage game.

		Miner 2	
		b_0	b_1
Miner 1	b_0	2.75, 2.25	3.00, 2.50
	b_1	3.25, 2.25	(3.50, 2.50)

Table A.5 – Payoff-matrix in intermediate stage $t = 2$ in the six-stage game.

		Miner 2	
		b_0	b_1
Miner 1	b_0	3.25, 2.75	3.50, 3.00
	b_1	3.75, 2.75	(4.00, 3.00)

Table A.6 – Payoff-matrix in intermediate stage $t = 2$ in the seven-stage game.