On the Bitcoin Limitations to Deliver Fairness to Users

Antonella del Pozzo, Sara Tucci-Piergiovanni, Önder Gürcan

On the Bitcoin Limitations to Deliver Fairness to Users

Önder Gürcan and Antonella Del Pozzo and Sara Tucci-Piergiovanni

CEA LIST
Point Courrier 174, Gif-sur-Yvette, F-91191 France
{onder.gurcan, antonella.delpozzo, sara.tucci}@cea.fr

Abstract. While current Bitcoin literature mainly focuses on miner behaviors, little has been done to analyze user participation. Because Bitcoins users do not benefit from any incentive, their participation in the system is conditional upon system ability to provide a transactional service at a reasonable cost and acceptable quality. A recent observed trend on a growing number of unconfirmed transactions seems, however, to substantiate that Bitcoin is facing service degradation. The objective of this paper is to shed some light on user participation in Bitcoin against a notion of system fairness, through a utility-based approach. We first introduce fairness to quantify the satisfaction degree of participants (both users and miners) with respect to their justified expectations over time. We then characterize user strategies, deriving the necessary condition for fairness, and we show Bitcoin limitations in delivering it. The utility-based model allows to finally draw conclusions on possible improvements for fairness to promote user participation.

Keywords: Bitcoin, Blockchain, Fairness, User Expectation.

1 Introduction

The blockchain protocol, introduced by Satoshi Nakamoto [13], is the core of the Bitcoin-like decentralized cryptocurrency systems. Participants following this protocol can create together a distributed economical, social and technical system where anyone can join (or leave) and perform transactions in-between without neither needing to trust each other nor having a trusted third party. It is a very attractive technology since it maintains a *public*, *immutable* and *ordered* log of transactions which guarantees an *auditable* ledger accessible by anyone.

Technically speaking, all participants of a blockchain system store unconfirmed transactions in their memory pools and confirmed transactions in their blockchains. Participants called users create transactions with a fee and then broadcast them across the blockchain network for being confirmed. After receiving a certain number of transactions, participants called miners try to confirm them as a block by solving a computational puzzle (a hash-based proof-of-work - PoW) of pre-defined difficulty by consuming a considerable amount electricity

power. Note that, each miner tries to confirm a different block, with its own set of transactions, put into its own order. The successful miner broadcasts its block to the network to be chained to the blockchain. This block contains the transactions, the unique hash value (the solution of the puzzle) and a block reward, which is composed of a static block reward plus the total fees of transactions, expressed as a transaction to the successful miner.

To analyze the security properties of Bitcoin-like blockchains [13], several formal studies have been conducted so far [8, 7, 18, 14]. Garay et al. [8] showed that the number of blocks created by honest miners is proportional to their fraction of computational powers if majority of the computational power belongs to them, assuming that there are honest and Byzantine miners. Eyal and Sirer [7] and Sapirstein et al. [18] showed that even if the majority of the miners are honest, a selfish miner having enough resource and good network connectivity can increase its proportion of blocks, assuming that there are honest and selfish miners. These studies conclude that Bitcoin-like blockchains are not promoting honest participation. In such context, Pass et al. [15] provided a first definition of *fair* protocol for miners: if, in any sufficiently long window of time, honest miners create blocks proportionally to their computational powers then the protocol is fair.

However, as well as miners, the participation of users is also important. Without users, miners will have no transactions to confirm[1]. In this sense, we claim that a comprehensive definition of *fairness* is crucial for improving overall participation. Both users and miners consider worthwhile to join and stay over time in the system only if they find it fair. For instance, miners find the system fair if they are able to create blocks as they expected, and users find the system fair if their transactions are confirmed as they expected. Hence, fairness can be defined as the satisfaction of expectations of participants to a certain degree.

The contributions of this paper are as follows:

- A detailed basic model for Bitcoin blockchain;
- An elaborated rational system model where all participants (users and miners) are modeled as rational agents;
- A formal definition of *fairness* with respect to the utilities of rational agents;
- A *necessary condition* to have user fairness;
- An analysis of the Bitcoin protocol limitations in delivering fairness to *users*.

The paper is organized as follows. Section 2 gives the related work about fairness in blockchain systems. Section 3 provides a formalization of the existing Bitcoin protocol. Section 4 defines the rational agent model and proposes a definition of *fairness* related to rational behaviors for each type of participant (users and miners). Moreover, a necessary condition for fairness provided to the users is presented. Section 5 presents an analysis of the agent behaviors and finally, Section 6 concludes the paper.

[1] Technically the miners can create empty blocks and get block rewards. But this is not the purpose of blockchain systems.

2 Related Work

In the rational agents context, two concepts play a key role: "*incentive compatibility*" and "*fairness*". Informally, a protocol is incentive compatible if rational nodes have incentives to follow it, while a protocol is fair if rational nodes are satisfied by executing the protocol actions, i.e., they have a profit acceptable for them.

Eyal et al. [7] show that the mining protocol itself is not incentive compatible: a miner that follows the protocol has a lower gain than a miner that does not. To such purpose they define a new attack strategy, the *selfish mining* in which colluding miners obtain a revenue larger than their fair share withholding new created blocks. Hereafter, other works analyzed scenarios where for miners it is more profitable to be selfish than to correctly follow the protocol (cf. [5],[18] and [9], just to cite a few). The blockchain protocol does not rely only on the mining task but also on the information (transactions, blocks, etc.) flooding. As discussed by Babaioff et al. [2], rational agents have not incentives to forward information.

Contrarily to incentive compatibility, less research has been done concerning fairness in the Bitcoin-like systems. In [10], the authors present an inclusive protocol and discuss about its fairness as related to proportion between the miner hashing power and their rewards (a similar concept appears in [6]). Finally, Pass and Shi [15] present a formal concept of "*fairness*" and a protocol that provides it. Informally they propose a protocol where "*honest players contributing a ϕ fraction of the computational resources get a ϕ fraction of the blocks (and thus rewards) in a sufficiently long window*".

All those works consider the strategies that rational miners can put in place with respect to the actual Bitcoin situation. Actually the Bitcoin protocol provides two incentives for a miner that solve the PoW, the static block reward and the transaction fees. Let us recall that the static block reward is periodically halving to eventually disappear. Carlsten et al. [3] analyze what would be more profitable for miners in a scenario where the transaction fees dominate the static block rewards. To do so they stated the decisions that miners can take at each time instant as: which block to extend, how many of the available transactions include in the block and for each unpublished block, whether or not to publish it. As a result, the Bitcoin will be an unstable system where selfish mining performs even better than in the current scenario advocating that block reward plays a key role for the stability of the system.

Carlsten et al. [3] is the closest to our contribution, not only for the scenarios considered, but also for the analysis on the decisions that miners can take. On our side we considered not only the miners' but also the users' actions in order to model the expectations and rewards of those participants and provide a definition of fairness general enough for both of them. At the best of our knowledge the only notion of fairness applied to the user side concerns the fair exchange in the e-commerce context [1] which is extended to the Bitcoin-Like scenario in [11] more in the sense that if there are two players performing an exchange then either both of them get what they want or none of them. All those fairness

definitions apply globally to the system, contrarily, in our work we provide a local definition of fairness looking at the perceptions that users have about the system fairness.

3 Basic Blockchain System Model

In this section we provide a high-level Bitcoin protocol description[2]. To this end, we first introduce a basic model for each element involved in the protocol and then we provide a high-level detailed pseudocode.

3.1 Network Model

We model the blockchain network as a dynamic directed graph $G = (N, E)$ where N denotes the dynamic node (vertex) set, E denotes dynamic directed link (edge) set. A node n can enter and leave G by using its $join(G)$ and $leave(G)$ actions respectively. Upon joining G, n discovers neighbor nodes to connect to[3]. A link $\langle n, m\rangle \in E$ represents a directed link $n \to m$ where $n, m \in N$, n is the owner of the link and n is the neighbor of m.

A node n can communicate with a set of recipient nodes R_n (where $\forall m \in R_n | \{n, m\} \in E$) by exchanging messages of the form $\langle n, msg, d\rangle$ where n is the sender, msg is the type and d is the data contained.

3.2 Node Model

Each node $n \in N$ has a list of its neighbors N_n where $N_n \subseteq N$ and $\forall m \in N_n | \langle n, m\rangle \in E$. A node n adds and removes another node m as its neighbor using its $addNeighbour(m)$ and $removeNeighbour(m)$ actions respectively. Each neighbor $m \in N_n$ is represented as a 3-tuple $\langle m, \times_m, t_m\rangle$ where $\times_m$ is the ban score and t_m is the last communication time. t_m is updated at each message receipt and if m has not communicated for more than some time, m is removed from N_n.

Each node n has a memory pool Θ_n in which it keeps unconfirmed transactions that have input transactions, an orphan pool $\bar{\Theta}_n$ in which they keep unconfirmed transactions that have one or more missing input transactions (orphan transactions) and a blockchain ledger B_n in which they keep confirmed transactions where $\Theta_n \cap \bar{\Theta}_n = \emptyset$, $\Theta_n \cap B_n = \emptyset$ and $\bar{\Theta}_n \cap B_n = \emptyset$ always hold.

[2] This description is based on the Nakamoto paper [13], the Bitcoin source code (`https://github.com/bitcoin/bitcoin`), bitcoin.org (`https://bitcoin.org/`) and the Bitcoin StackExchange forum (`https://bitcoin.stackexchange.com/`).

[3] `https://bitcoin.stackexchange.com/questions/53938/how-does-one-node-connect-to-other-nodes`, last access 30 May 2017.

3.3 Miner Model

A (user) node n can turn to be a miner node if it chooses to create blocks for confirming the transactions (mining) in its memory pool Θ_m. It can start and stop mining using the actions of the form $startMining()$ and $stopMining()$ respectively, and n is said to be a miner node if it started mining but has not stopped yet. The set of miner nodes is then denoted by M where $M \subseteq N$. In order to be able to mine, $n \in M$ has to solve a cryptographic puzzle (i.e. Proof of Work) using its hashing power[4] q_n where $q_n > 0$. The first successful miner is awarded by a fix amount of reward plus the total fee of the transactions.

Algorithm 1 The actions of a miner node m.

```
1:  action startMining()
2:    ω ← true
3:    createBlock()
4:
5:  action stopMining()
6:    ω ← false
7:
8:  action createBlock()
9:    while (ω) do
10:     i ← |B*_n| + 1
11:     θ_n ← selectTransactions(Θ_n)
12:     createBlock(i, θ_n)
13:   endwhile
14:
15: action createBlock(i, θ_n)
16:   Σf ← calculateTotalFee(θ_n)
17:   tx_c ← createTransaction(n, R + Σf)
18:   Ψ_i ← createMerkleTree(tx_c, θ_n)
19:   t_{b_i} ← getTime()
20:   h_i = ⟨v_n, H(h_{i−1}), H(h_i), t_{b_i}, η_i, ψ_i⟩
21:   try
22:     find η_i such that H(h_i) < μ holds
23:     b_i = {h_i, Ψ_i}
24:     processBlock(∅, b_i)
25:   catch (i = |B*_n| ∨ ω = false)
26:
```

The actions performed by miners are reported in details in the Algorithm 1.

3.4 Blockchain Model

We model the blockchain ledger of a node n as a dynamic append-only tree $B_n = \{b_0 \xleftarrow{r_0} b_1 \xleftarrow{r_1} ... \xleftarrow{r_{h-1}} b_h\}$ where each block b_i $(0 < i \leq h)$ contains a cryptographic reference r_{i-1} to its previous block b_{i-1}, $h = |B_n|$ is the depth of B_n, b_0 is the root block which is also called the *genesis block* and b_h is the furthest block from the genesis block which is referred to as the *blockchain head*.

A block b_{i-1} can have multiple children blocks, which causes the situation called a $fork$. The *main branch* is then defined as the longest path h from any block to b_0 and is denoted as $B^\star_n$ where $|B^\star_n| = h$ and $B^\star_n \subseteq B$ such that $|B^x_n| < |B^\star_n|$ for all branches $B^x_n \subset B_n$ where $B^x_n \neq B^\star_n$. All branches other than the main branch are called *side branches*. If at any time, there exists more than 1

[4] Hashing power is proportional to computational power and nodes may change this power by time.

longest path with a depth h (i.e. there are multiple heads), the blockchain ledger B_n is said to be *inconsistent* and thus $B_n^\star = \emptyset$. This situation disappears when a new block extends one of these side branches and creates $B_n^\star$. The blocks on the other branches are discarded and referred as *stale blocks*.

3.5 Block Model

We denote a block as $b_i = \langle \mathtt{h}_i, \Psi_i \rangle$ where $\mathtt{h}_i$ is the block header and Ψ_i is the block data. The block data Ψ_i contains all the transactions organized as a Merkle tree [12]. Basically, the copies of each transaction are hashed, and the hashes are then paired, hashed, paired again, and hashed again until a single hash remains, the merkle root of a merkle tree. We denote a merkle tree and its root as Ψ_i and ψ_i respectively. A merkle tree Ψ_i is created using the action of the form $createMerkelTree(tx_c, \theta_m)$ where tx_c is the coinbase transaction[5] that rewards the miner node $m \in M$ with the block reward $R = \mathtt{F} + \Sigma f$ for its work[6] (where $\mathtt{F}$ is the static block reward, and Σf is the total fees of the transactions included in this block), $\theta_m \subseteq \Theta_m$ is the set of candidate transactions chosen for this block. Here it is important to note that, blocks have limited sizes[7] and thus the size of θ_m can not exceed this limit[8]. The set of candidate transactions θ_m are selected using the action of the form $selectTransactions(\Theta_m) : \theta_m$ where Θ_m is the memory pool of the miner. The total fee Σf is then calculated by using the action of the form $calculateTotalFee(\theta_m)$.

The block header is denoted as $\mathtt{h}_i = \{v_n, \mathcal{H}(\mathtt{h}_{i-1}), \mathcal{H}(\mathtt{h}_i), t_{b_i}, \eta_i, \psi_i\}$ where v_n is the version number of the protocol used by n, $\mathcal{H}(\cdot)$ is the cryptographic hash function, $\mathcal{H}(\mathtt{h}_{i-1})$ is the cryptographic hash code of the header of the previous block b_{i-1} $(i > 0)$, $\mathcal{H}(\mathtt{h}_i)$ is the cryptographic hash code of $\mathtt{h}_i$ generated by m, t_{b_i} is the current time stamp, η_i is an integer nonce value $(\eta_i \geq 0)$ to be found by the miner in order to generate the right $\mathcal{H}(\mathtt{h}_i)$ conforming to the difficulty level μ defined in the protocol version v (the cryptographic puzzle mentioned in Section 3.3) and ψ_i is the root of the merkle tree.

3.6 Transaction Model

We model a transaction as $tx = \langle I, O \rangle$ where I is a list of inputs $(I \neq \emptyset)$ and O is a list of outputs $(O \neq \emptyset)$. Each input $i \in I$ references to a previous unspent output (for spending it). Each output then waits as an Unspent Transaction Output (UTXO) until an input spends it. If an output has already been spent

[5] Any transaction fees collected by the miner are also sent in this transaction.

[6] Note to remember is that the coins in a coinbase transaction cannot be spent until they have received 100 confirmations in the blockchain. All things being equal, 100 confirmations should equate to roughly 16 hours and 40 minutes.

[7] The current maximum block size in Bitcoin is 1 MB. See `https://bitcoin.org/en/glossary/block-size-limit`, last access on 18 July 2017.

[8] Average block size for Bitcoin is given in `https://blockchain.info/charts/avg-block-size`, last access on 18 July 2017.

by an input, it cannot be spent again by another input (no double spending). We model the outputs as $o_i = \langle m, ¢_{o_i} \rangle$ where $m \in N$ is the receiver of the coins $¢_{o_i}$ ($¢_{o_i} \geq 0$). All inputs of a transaction have to be spent in that transaction and the total input coins $¢_I$ has to be greater than or equal to the total output coins $¢_O$. The fee f_{tx} of a transaction tx is then modeled as $f_{tx} = ¢_I - ¢_O$. The fees of transactions are not fixed and are estimated by using the action of the form $estimateFee(I, O)$ where I is the set of inputs and O is the set of outputs. Depending on the fee to be paid, if there are still some coins left to be spent, the sender can add an output that pays this remainder to itself.

Algorithm 2 The actions of a user node n.

```
1: action makeTransaction(m,¢)
2:   tx = createTransaction(m,¢)
3:   processTransaction(n, tx)
4:
5: action createTransaction(m,¢)
6:   I ← selectUnspentTransactionOutputs(B_n, ¢)
7:   o_1 ← ⟨m,¢⟩
8:   f ← estimateFee(I, {o_1})
9:   ¢_r ← ¢_I − ¢ − f
10:  o_2 ← ⟨n,¢_r⟩
11:  return tx = ⟨I, {o_1, o_2}⟩
12:
13:
14: action processTransaction(s, tx)
15:   × ← validateTransaction(tx, Θ_n)
16:   if (× = 0) then
17:     U ← getUnspentTransactionOutputs()
18:     if ({∀i ∈ I_tx ∧ ∃o ∈ U | o ≺ i} = ∅) then
19:       Θ̄_n ← Θ̄_n ⋃ {tx}
20:     else acceptTransaction(tx) endif
21:   else updateBanscore(s, ×) endif
22:
23: action acceptTransaction(tx)
24:   Θ_n ← Θ_n ⋃ {tx}
25:   sendMessage(⟨n,"inv",H(tx)⟩,N_n)
26:   processTransaction(∅, ∀tx' ∈ Θ̄_n | tx' ≺ tx)
```

The coinbase transaction tx_c (see Section 3.5) is special transaction that collects and spends any transaction fees paid by transactions included in a block and exceptionally it does not have any input set ($I = \emptyset$). It is the first transaction in a block and can only be created by a miner.

Creating a transaction tx by a node n is modeled as the action of the form $createTransaction(m,¢)$ where ¢ is the amount of coins ($¢ > 0$) paid to m ($n, m \in N$). Selecting the right inputs to be able spend ¢ is modeled using the action of the form $selectUnspentTransactionOutputs(B_n,¢)$ where B_n is the blockchain and ¢ is the coin to be spent. All those actions are detailed in Algorithm 2.

In the next section, we define a rational system model by augmenting the basic system model for better capturing its properties formally and defining the concept of *fairness*.

4 Rational Model

In this section we augment the basic blockchain system model given in Section 3 by the rational agent concept. Informally speaking, such agent chooses its actions/behavior with respect to its perceptions in order to maximize its utility. In such context we introduce a concept of fairness dependent on the agent behaviors and the corresponding utilities. To this aim, the remaining part of the section characterizes both agent and minor behaviors.

4.1 Rational Agent Model

A rational agent behaves according to its local perceptions and local knowledge, models uncertainty via expected values of variables or actions, and always chooses to perform the actions with the optimal expected outcome (among all feasible actions) for maximizing its utility [17]. We model all nodes in the blockchain network as rational agents and denote the rational agent set as N. Each rational agent $n \in N$ has a set of actions A_n and a utility function $\mathcal{U}_n$. Using A_n and $\mathcal{U}_n$, n uses a decision process where it identifies the possible sequences of actions to execute. We call these sequences as rational behaviors of n and denote as β. The objective of n is to choose the behaviors that selfishly keep $\mathcal{U}_n$ as high as possible.

We model the utility function of a rational agent $n \in N$ as $\mathcal{U}_n = u_0 + \sum_{i=1}^{k} \mathcal{U}(\beta_i)$ where u_0 is the initial utility value, $k \geq 0$ is the number of behaviors executed so far and $\mathcal{U}(\beta_i)$ is the utility value of the behavior β_i. A utility value $\mathcal{U}(\beta_i)$ can also be interpreted as the *degree of satisfaction* experienced by the realization of β_i. The utility value $\mathcal{U}(\beta_i)$ is calculated as $\mathcal{R}(\beta_i) - \mathcal{C}(\beta_i)$ where $\mathcal{R}(\beta_i)$ is the overall reward gained and $\mathcal{C}(\beta_i)$ is the overall cost spent for the execution of β_i.

When an agent needs to choose a behavior for execution, it needs to calculate its expected value. The expected value $\mathcal{E}(\beta_i)$ depends on the probabilities of the possible outcomes of the execution of β_i. We model the expected value as $\mathcal{E}(\beta_i) = \sum_{j=1}^{m} p_j \cdot \mathcal{U}(\beta_i^j)$ where $m > 0$ is the number of possible outcomes, $\mathcal{U}(\beta_i^j)$ is the utility value of the possible jth outcome β_i^j and p_j is the probability of this outcome such that $\sum_{j=1}^{m} p_j = 1$. Since behaviors are chosen based on their expected values, it can be said that a utility value $\mathcal{U}(\beta_i)$ represents the *satisfaction of expectation* about β_i.

> **Fairness:** A rational agent $n \in N$ finds a system (i.e. the blockchain network) G *fair*, if the total satisfaction of its expectations $\mathcal{U}_n$ is above a certain degree τ_n where $\tau_n < u_0$.

If at any time, an agent n finds G *unfair* ($\mathcal{U}_n \leq \tau_n$), it may decide to leave G if from its points of view it will not be possible to increase its overall utility above τ_n by calculating the expected values of its possible future behaviors. In other words, n may decide to leave G if $\mathcal{U}_n + \sum_{j=k}^{m} \mathcal{E}(\beta_j) \leq \tau_n$ where β_k, ..., β_m are sufficiently enough desired future behaviors of n.

Based on this model, to be able to decide if Bitcoin-like blockchains are fair, we propose behavioral models for rational user and miner agents in the next subsection.

4.2 Rational Behavior Model

The rational behaviors given in this section define the possible strategies of agents by using and improving the basic actions given in Section 3.

To formalize the behaviors, we model a round based approach (like Garay et al. [8]) in which miner agents start creating a new block with at the beginning of

the round and a round ends when a new block is successfully created by one of the miners. Both user and miner agents make their decisions on a roundly basis. This round-based model implicitly assumes that the block sent at at the end of the round is immediately delivered by all participants, i.e. communication delay is negligible with respect to block generation time.

User Agent Behaviors The user agent in the system makes transactions. To make a transaction, she has to determine a fee to assign to the transaction. This decision depends on the probability that the assigned fee has to lead the transaction confirmed as soon as possible and the interest of the agent on the transaction. All the duration the user agent spends waiting for the transaction confirmation is said to be the waiting cost. *Such a cost is unavoidable*, the user agent has to wait as long as the transaction is confirmed since the basic protocol does not allow revoking transactions, even if they are not confirmed after a considerable time.

We model making a transaction with a specific fee f as with the action of the form $makeTransaction(m,¢)$ where $m \in N$ is the receiver and ¢ is the amount of coins. For simplicity, it is assumed that a users agent has an ordered set of fees $\{f_1, f_2, \ldots, f_k\}$ to use. It is also assumed that $f_i < f_{i+1}$ where $0 < i < k$ and $0 \leq P(f_i) \leq P(f_{i+1}) \leq 1$ where $P(f)$ is the probability of a transaction with a fee f to be confirmed.

We model the interest the user agent has with respect to the transaction as I. It is assumed that the initial interest does not decrease round after round. We denote the waiting cost as $C(f)$, which is proportional to the fee f applied to the transaction. Intuitively, waiting for the confirmation of a transaction with a high fee assigned is more costly than waiting for the confirmation of a transaction with a lower fee assigned.

Along with the interest and the waiting cost, an important role is played by the fee assigned to the transaction. The fee is paid only when the transaction is confirmed in a block in the main chain. Thus, when a transaction is confirmed at round r, the expected value is given by the net difference between the interest I and the assigned fee f plus the waiting cost $C(f)$ for all the $r-1$ rounds while the user agent waited.

Let us consider a user agent that issues a transaction tx and chooses a fee f, has an interest I on tx and a waiting cost $C(f)$. The confirmation probability of tx at the first round is $P(f)$, thus if tx is confirmed during the first round the user agent gains: $(I-f) \cdot P(f)$ and no waiting cost. Let $\overline{P(f)} = 1 - P(f)$ be the unconfirmation probability of tx during a round. If tx is confirmed during the second round she gains: $\overline{P(f)} \cdot P(f) \cdot (I-f) - \overline{P(f)} \cdot C(f)$, and if confirmed during the third round she gains: $\overline{P(f)}^2 \cdot P(f) \cdot (I-f) - \overline{P(f)}^2 \cdot C(f)^2$. Generalizing for an infinite number of rounds r, the expected value $\mathcal{E}$ is:

$$\mathcal{E}(\beta) = \sum_{r=1}^{\infty} \overline{P(f)}^{r-1} \cdot P(f) \cdot (I-f) - \sum_{r=1}^{\infty} \overline{P(f)}^{r-1} \cdot C(f)^{r-1} \tag{1}$$

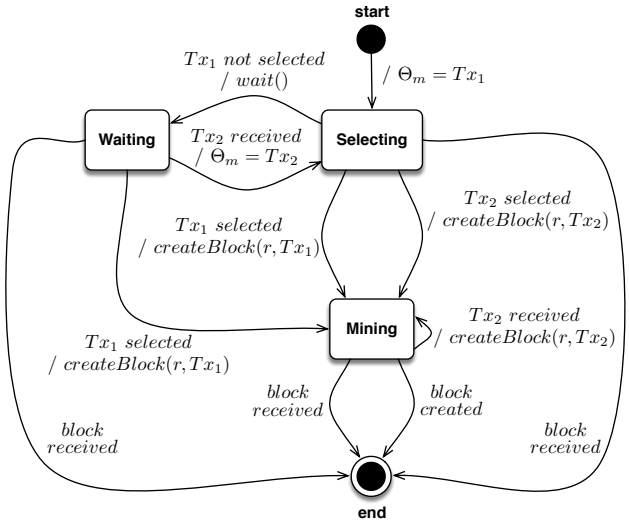


Fig. 1: The state machine of the rational mining behavior of a miner m for a round r.

where $\beta = \{makeTransaction(m,¢, f)\}$. An analysis of Equation 1, along with necessary condition for fairness, is given in Section 5.1.

Miner Agent Behaviors In the decision model for the user agent, we considered a probability P(f) constant over rounds[9]. On the other hand, in this section we want to shed some light on the miner behavior, in order to have insights on how the probability changes over rounds.

To this end, we consider the miners at the beginning of a round and focus on understanding how miners choose the transactions. In particular, since the size of the block is limited, *miners could decide to deliberately exclude an unconfirmed transaction that has already been received with a lower fee*. This behavior would obviously delay the confirmation time of that transaction and affects its confirmation probability. More in detail, we consider that at the beginning of a round a miner can decide if to start immediately creating a block with the transactions it already has in its pool (Tx_1)[10] or to wait for a set of transactions with better fees (Tx_2) to arrive. If such transaction arrives a miner decides if to continue to mine Tx_1 or to start mining with Tx_2.This decision process is depicted in the state machine given in Figure 1.

Let $\{0,1\}^k$ be the range of the hash function run to solve the Proof-of-Work (PoW) and k the security parameter and let $\mathcal{D}$ be difficulty level. Let q be the

[9] Technically speaking this means to consider $P(f)$ modeled as a stationary process, and the blockchain system as an ergodic dynamical system [4].

[10] For simplicity, it is assumed that the sizes of both Tx_1 and Tx_2 are equal to the maximum block size.

upper bound on the number of times the hash function can be invoked during a round and each invocation is independent from the other (as independent Bernoulli trials). The probability that an attempt solves the PoW is $p = \frac{\mathcal{D}}{2^k}$ thus the upper bound on the number of solutions that a miner can found per round is $P = p \cdot q$. If the miner solves the PoW, it gets a reward $R = \mathtt{F} + f_{Tx}$, where f_{Tx} is the total fee associated to the transactions in the set Tx. In the following, we model how P and R changes over the time, to take into account the miner decision to wait or not before starting creating a block.

Along with the miner rewards, we consider also the power consumption cost. We consider that each attempt implies a cost that we call C. Thus, the miner gain is net between the reward and the costs multiplied by the probability to win the PoW which depends on how many attempts are performed. The reward is $R = \mathtt{F} + f_{Tx_2}$ if the expected Tx_2 arrives and $R = \mathtt{F} + f_{Tx_1}$ otherwise. At each round r, miners calculate the expected values of each different behavior and choose one of them.

Algorithm 3 Rational *createBlock*() action of a rational miner agent m.

```
1:  action createBlock()
2:    while (w) do
3:      i ← |B*_n| + 1
4:      E ← calculate expected values
5:      Tx_1 ← selectTransactions(Θ_n)
6:      depending on E either:
7:        wait until Σf_{Tx_2} >> Σf_{Tx_1} such that Tx_2 ← selectTransactions(Θ_n)
8:          if Tx_2 arrives either createBlock(i, Tx_2) or createBlock(i, Tx_1)
9:          if Tx_2 not arrives either createBlock(i, Tx_1) or continue waiting
10:     or:
11:       start createBlock(i, Tx_1)
12:         if Tx_2 arrives either createBlock(i, Tx_2) or continue
13:         if Tx_2 not arrives continue
14:   endwhile
```

Based on the state machine given in Figure 1, the rational behaviors of miner agents can be implemented by improving the *createBlock()* action given in the Basic System Model (Algorithm 1) as shown in Algorithm 3. In particular, we consider three behaviors: (β_1) the miner waits the arrival of Tx_2 ignoring Tx_1; (β_2) the miner starts mining with Tx_1 ignoring Tx_2 (either if it arrives or not); (β_3) the miner starts mining with Tx_1 and when Tx_2 arrives she starts mining with Tx_2. Following the same reasoning as in Section 4.2, for each behavior we compute the expected values:

$$\mathcal{E}(\beta_1) = \sum_{q'=1}^{q} 1/2 \cdot (p \cdot (1-p)^{q'-1} \cdot (\mathtt{F} + f_{Tx_2}) - C) \tag{2}$$

$$\mathcal{E}(\beta_2) = \sum_{q'=1}^{q} (p \cdot (1-p)^{q'-1} \cdot (\mathtt{F} + f_{Tx_1}) - C) \tag{3}$$

$$\mathcal{E}(\beta_3) = \sum_{q'=1}^{q} (1/2 \cdot p \cdot (1-p)^{q'-1} \cdot (\mathtt{F} + f_{Tx_1}) + 1/2 \cdot p \cdot (1-p)^{q'-1} \cdot (\mathtt{F} + f_{Tx_2}) - C) \tag{4}$$

These expected values depend on the transaction the miner is considering to mine (Tx_1) and the probability that a juicy transaction (Tx_2) arrives. Having a probability distribution over such an event is far from being trivial. We consider that the user has no information about the exact distribution probability, i.e., she takes a decision under ignorance. By applying the principle of insufficient reason[11] we then assign the same probability, to the two events: "Tx_2 *arrives*"; "Tx_2 *does not arrive*".

We show the results associated with a scenario in which the two events are considered as equally as possible in Section 5.2.

5 Analyses and Results

We analyzed the rational behaviors of each agent proposed in Section 4.2 using Matlab R2017a. It is assumed that the initial utility values u_0 of all agents are the same and high enough from the threshold τ. In the following, we provide results of these analyses considering both the users and the miners employing synthetic data.

5.1 User Agent

Recalling Equation (1), it is clear that depending on the values of $\overline{P(f)}$ and $C(f)$ the expected value $\mathcal{E}$ may or may not converge to $-\infty$.

Let us first give a graphical intuition of the series behavior (Figure 2). For simplicity, we denote the first part of Equation (1) as $\alpha_{gain} = \sum_{r=1}^{\infty} \overline{P(f)}^{r-1} \cdot P(f) \cdot (I - f)$ and the second part as $\alpha_{cost} = \sum_{r=1}^{\infty} \overline{P(f)}^{r-1} \cdot C(f)^{r-1}$. Consequently, $\mathcal{E} = \alpha_{gain} - \alpha_{cost}$. As can be seen, if $C(f) \geq 1/\overline{P(f)}$ then α_{cost} goes to ∞, thus $\mathcal{E}$ goes to $-\infty$ (cf. Fig 2a). Otherwise, if $C(f) < 1/\overline{P(f)}$ the cost converge to a bounded quantity and $\mathcal{E}$ as well (cf. Fig 2b). In the first case, it can be said that the user agent is facing an unfair situation, if she chooses the fee f her expected value $\mathcal{E}$ is $-\infty$ due to the fact she would pay an infinite waiting cost[12] with no possibility to change the strategy.

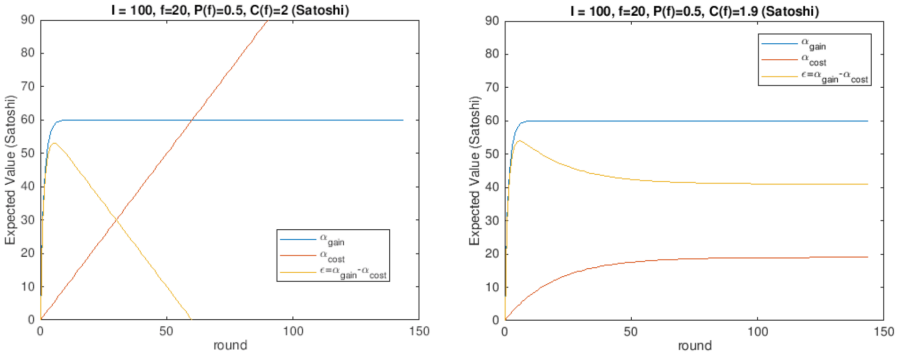
Based on this observation we state that

[11] The principle of insufficient reason prescribes that if one has no reason to think that one state of the world is more probable than another, then all state should be assigned equal probability [16] .

[12] This is the same situation as in the Saint Petersburg Paradox [19].

$$C(f) < 1/\overline{P(f)} \tag{5}$$

is a *necessary condition* for *fairness* since its violation leads the waiting cost to ∞ and the $\mathcal{E}$ for the transaction to issue to $-\infty$. That is, in such scenario is never profitable to try.

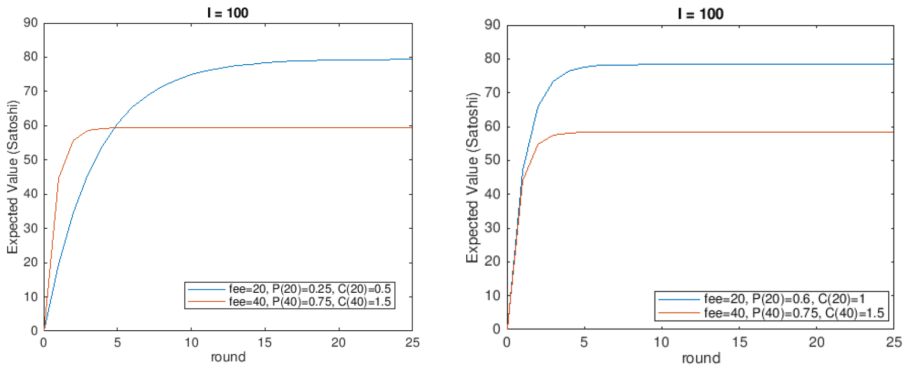


(a) (b)

Fig. 2: Expected values during a period of 144 rounds (one day), depending on the relationship between $C(f)$ and $\overline{P(f)}$, $\mathcal{E}$ may converge to a finite quantity or to $-\infty$. $\overline{P(f)}$ and $C(f)$ are chosen such that the different behavior of $\mathcal{E}$ can be seen when (a) Equation (5) holds, and (b) not holds.

In the following, we consider a scenario where the necessary condition given in Equation (5) is met and we show with a practical example why having $C(f) < 1/\overline{P(f)}$ is not sufficient to have fairness as defined in Section 4. We consider an user agent whose already issued some transactions, her $\mathcal{U}_n = -30Satoshi$ ($1Satoshi = 0.00000001BTC$) and her threshold is $\tau_n = 70Satoshi$. Since $\mathcal{U}_n < \tau_n$ the user finds the system unfair if the expected value of the next transaction does not allow to reach the threshold, in this case the user will leave. Let us consider that such user has a transaction to issue and can decide to play with two fees: a low and a high fee, 20 and 40 Satoshi respectively. In order to rise $\mathcal{U}_n$ above τ_n, the expected values have to be at least 100 $Satoshi$.

Figure 3 plots the expected values for two different scenarios with respect to rounds where in both cases the necessary condition (5) is met. In the first scenario (Figure 2a),the user agent can issue a transaction with a low fee of 20 Satoshi such that $P(20) = 0.25$ and $C(20) = 0.5$. If the user agent chooses a high fee of 40 Satoshi then such probability rise to 0.75 and the waiting cost to 1.5. In the second scenario (Figure 2b) nothing changes but the probability and the waiting cost for the low fee, $P(20) = 0.6$ and $C(20) = 1$ respectively. In both cases the user interest I is 100 Satoshi and is constant.



(a) (b)

Fig. 3: Expected values of user behaviors given in Section 4.2 with respect to rounds.

As can be seen, whatever fee the user decides to assign to the transaction the expected value is below 100 *Satoshi* so the user has no way to rise $\mathcal{U}_n$ above τ_n and since she will find the system unfair with no possibility to change such situation, she will leave the system. It is important to underline that *the perception of fairness is strictly local to the user since it depends to the interest she puts in the transactions issue and the waiting cost it perceives.*

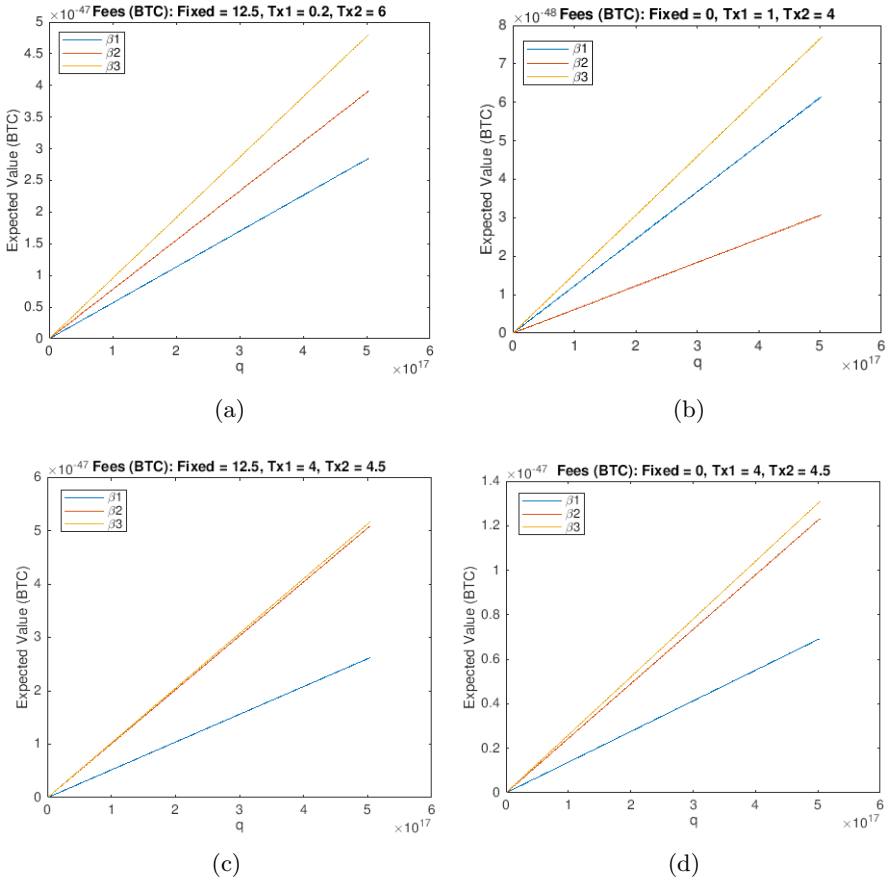
5.2 Miner Agent

Figure 4 depicts miner expected values as defined in Section 4.2. We consider different scenarios depending on the reward that a miner gets solving the PoW: a fixed part and the sum of the transactions fee included in the block. For simplicity, it is assumed that each miner can insert only one transaction in a block[13] and that at the beginning of the round the miner has in its transaction pool a transaction Tx_1 and it is expecting to deliver a second transaction Tx_2 such that $Tx_2 > Tx_1$. We consider that the miner has a AntMiner S9 device[14], thus miner has $q = 504e15$ attempts to solve the PoW during a round and the probability to solve it is $p = 6.1201e - 66$ for each attempt. In this analysis we are comparing the different strategies in one single round and the power cost is common and constant but a way greater than the possible reward in one round, for this reason we do not consider the cost in the plots. In Figure 4 on the y-axis, we have the Expected Value in Bitcoins (BTC) and on the x-axis the q attempts.

As can be seen, whenever Tx_2 arrives it is convenient to swap the two transactions and start creating a new block with Tx_2. How convenient it is depends

[13] In the Bitcoin protocol blocks have fixed size (see Section 3.5).

[14] Such device hash rate is 14TH/s, which means $14e12 * 36000$ attempts to solve the PoW in a round (10 minutes in average).



(a) (b) (c) (d)

Fig. 4: Expected values of miner behaviors with respect to rounds. In cases (a) and (c) the fixed reward is 12.5 BTC, in cases (b) and (d) fixed reward is zero.

on the difference between the two fees, *independently of the fixed part amount*. In Figure 4a and Figure 4b, $Tx_2 >> Tx_1$ and for the miner it is clearly convenient to swap from Tx_1 to Tx_2. This this is also true if the transaction fees are close to each other, Figure 4c and Figure 4d. Let us note that in the case of a bigger block size the miner would have started to mine a block with both transactions to increase her reward.

6 Discussion and Conclusion

To the best of our knowledge, this is the first study focusing on *users*. The rational system model given in Section 4 allows us to define fairness as the overall satisfaction of participants. However, there is still no mechanism which guarantees fairness for users. As shown in Section 5, the expected values for the users are always decreasing when their waiting costs are not taken into account by the miners. Moreover, we showed that the miners as rational agents have no incentive to keep the probability over rounds stationary.

Based on these findings, the main issues arisen by Bitcoin-like blockchain systems can be resumed as follows:

- There is no mechanism for a user to cancel an already issued transaction, i.e. once she decides to engage in the game, it can never abandon the game. This implies expected values going to minus infinity. In decision theory terms, this would mean assuming a user having an infinite interest on a transaction, which is in fact hard to assume in real settings.
- Given different fees with their associated probabilities, more flexibility to guarantee fairness would have been reached by allowing the user (experiencing a long waiting time) to resend the transaction with a lower fee. This flexibility, on the other hand, is not achieved in Bitcoin-like blockchain systems mainly due to two reasons: (1) there is no deterministic guarantee that the system chooses a given copy of the transaction, it is only guaranteed with high probability that only one copy will be inserted into the blockchain, and (2) the miner behavior favors the transaction with the highest fee to get included in a block.
- The block size is a fixed parameter today. Obviously, if the volume of transactions increases over time, the block size can become a scarce resource and miners will be more apt to deliberately delaying a transaction with a low fee. This will dramatically drop the probabilities for low fees and their corresponding expected values by time.
- Fees are decided by the users, leading to a possible race among users fees. Such a situation would make the probabilities difficult to predict, and expected values of users might drop even more quickly.
- The fairness is a locally perceived concept and it must indeed be tracked. This needs further detection mechanisms not included in the current systems.

In a nutshell, we can claim that Bitcoin-like blockchain systems do not envisage any specific mechanism to avoid unfairness for the users. Unfairness situations are not caused by the users and it is the users who has to sacrifice more.

Such situations may reduce the satisfaction of the users dramatically, and as a result the users may leave the system totally. Consequently, the rational system model should be augmented with mechanisms that allow agents to change their behaviors upon detecting unfair situations for balancing their overall satisfaction. Such a system will obviously encourage honest participants to stay in the blockchain system.

Besides, it is particularly difficult to devise such mechanisms while preserving the security properties of blockchain systems. This is especially true for those solutions envisaging to give rewards to users (this will change the computation of the expected value given in Section 4) for other protocol actions they do in the system, such as forwarding messages or validating transactions and blocks. Unfortunately, no proved solution exists on this line, to the best of our knowledge.

Pragmatically speaking, to avoid security issues and related complex analyses, we claim that a promising approach would be to stick as much as possible to the original Bitcoin protocol, introducing mechanisms to improve the degree of fairness of the system. For example, it would be interesting to consider how to give the possibility to the user to revoke a transaction, or to re-issue it again with a lower fee. It could also be interesting to allow the block size to be an adjustable parameter, as it is now for the difficulty parameter of the mathematical puzzle[15].

References

1. Asokan, N.: Fairness in electronic commerce (1998)
2. Babaioff, M., Dobzinski, S., Oren, S., Zohar, A.: On bitcoin and red balloons. In: Proceedings of the 13th ACM conference on electronic commerce. pp. 56–73. ACM (2012)
3. Carlsten, M., Kalodner, H., Weinberg, S.M., Narayanan, A.: On the instability of bitcoin without the block reward. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 154–167. ACM (2016)
4. Coudene, Y.: Ergodic Theory and Dynamical Systems. Universitext, Springer-Verlag London (2016), http://dx.doi.org/10.1007/978-1-4471-7287-1
5. Eyal, I.: The miner's dilemma. In: Security and Privacy (SP), 2015 IEEE Symposium on. pp. 89–103. IEEE (2015)
6. Eyal, I., Gencer, A.E., Sirer, E.G., Van Renesse, R.: Bitcoin-ng: A scalable blockchain protocol. In: NSDI. pp. 45–59 (2016)
7. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: International Conference on Financial Cryptography and Data Security. pp. 436–454. Springer (2014)
8. Garay, J., Kiayias, A., Leonardos, N.: The Bitcoin Backbone Protocol: Analysis and Applications, pp. 281–310. Springer Berlin Heidelberg, Berlin, Heidelberg (2015), http://dx.doi.org/10.1007/978-3-662-46803-6_10

[15] During the submission process of this paper (August 2017), the Bitcoin (BTC) protocol, which has 1 MB of block size, has been hard forked as the Bitcoin Cash (BCC) protocol, which has 8 MB of block size. However, it is early to conclude if BCC is better than BTC for the moment.

9. Göbel, J., Keeler, H.P., Krzesinski, A.E., Taylor, P.G.: Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. Performance Evaluation 104, 23–41 (2016)
10. Lewenberg, Y., Sompolinsky, Y., Zohar, A.: Inclusive block chain protocols. In: International Conference on Financial Cryptography and Data Security. pp. 528–547. Springer (2015)
11. Liu, J., Li, W., Karame, G.O., Asokan, N.: Towards fairness of cryptocurrency payments. arXiv preprint arXiv:1609.07256 (2016)
12. Merkle, R.C.: A Digital Signature Based on a Conventional Encryption Function, pp. 369–378. Springer Berlin Heidelberg, Berlin, Heidelberg (1988), http://dx.doi.org/10.1007/3-540-48184-2_32
13. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2018), https://bitcoin.org/bitcoin.pdf
14. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. IACR Cryptology ePrint Archive 2016, 454 (2016)
15. Pass, R., Shi, E.: Fruitchains: A fair blockchain. Cryptology ePrint Archive, Report 2016/916 (2016), http://eprint.iacr.org/2016/916.pdf
16. Resnik, M.D.: Choices: An introduction to decision theory. U of Minnesota Press (1987)
17. Russell, S.J., Norvig, P.: Artificial Intelligence - A Modern Approach (3. internat. ed.). Pearson Education (2010)
18. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal selfish mining strategies in bitcoin. In: International Conference on Financial Cryptography and Data Security. pp. 515–532. Springer (2016)
19. Weiss, M.: Conceptual Foundations of Risk Theory. Technical bulletin (United States. Department of Agriculture), U.S. Department of Agriculture, Economic Research Service (1987)