Bitcoin Mining Pools: A Cooperative Game Theoretic Analysis

Yoad Lewenberg[1]
yoadlew@cs.huji.ac.il

Yoram Bachrach[2]
yobach@microsoft.com

Yonatan Sompolinsky[1]
yoni_sompo@cs.huji.ac.il

Aviv Zohar[1]
avivz@cs.huji.ac.il

Jeffrey S. Rosenschein[1]
jeff@cs.huji.ac.il

[1]School of Computer Science and Engineering, The Hebrew University of Jerusalem, Israel
[2]Microsoft Research, Cambridge, United Kingdom

ABSTRACT

Bitcoin is an innovative decentralized cryptocurrency whose core security relies on a "proof of work" procedure, which requires network participants to repeatedly compute hashes on inputs from a large search space. Finding one of the rare inputs that generates an extremely low hash value is considered a successful attempt, allowing miners to approve new transactions and, in return, to collect rewards in bitcoins.

This reward allocation, which provides the incentive for miners to participate, is a random process with a large variance. Miners who desire a steady income thus often participate in *mining pools* that divide among their members the earned rewards, and reduce this variance. Mining pools are slightly better at coordinating participants due to lower-latency communication, a fact which implies that they manage to collect slightly higher rewards.

We examine dynamics of pooled mining and the rewards that pools manage to collect, and use cooperative game theoretic tools to analyze how pool members may share these rewards. We show that for some network parameters, especially under high transaction loads, it is difficult or even impossible to distribute rewards in a stable way: some participants are always incentivized to switch between pools.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence — Multiagent Systems

General Terms

Economics

Keywords

Bitcoin; Mining Pool; Game Theory; Cooperative Game

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

1. INTRODUCTION

Bitcoin [30] is a digital currency created in 2009. Its main achievement is its ability to arrive at a consensus about the valid transaction history in a totally decentralized fashion.

Agents in the Bitcoin network contribute computational power in order to maintain, secure, and extend Bitcoin's public ledger, the *block chain*. In return for their resources, the agents are awarded some amount of bitcoins, in some proportion to the computational power they invested.

Bitcoin's security relies on a *proof of work* framework, where a money transfer is only considered valid once the system obtains proof that a sufficient amount of computational work has been exerted by authorizing nodes. To achieve this, the network of participants, called *miners*, constantly attempts to solve cryptographic puzzles in the form of a hash computation. Each block in the shared data structure, the block chain, contains a set of transactions. The process of adding a new block to the block chain is called *mining*. To mine a block, a miner must examine inputs in a huge possible space X, seeking an input $x \in X$ that, when hashed along with the block's contents using a cryptographic hash function h, yields a value below a certain threshold t, so that $h(b(x)) < t$ (where $b(x)$ denotes the block with value x inserted into it). To incentivize participants to search for such an input, when such a hash is found and a block is mined, the block is released to the network, and if the majority of miners (in terms of computational power) consider this block to be valid and build further blocks on top of it, the miner who mined that block is rewarded with bitcoins.

An input x selected at random from X has a very small probability of having a low value under the hash, denoted as $p_t = Pr_{x \in X}(h(b(x)) < t)$. Thus trying a random input from the space is a Bernoulli trial with a success probability p_t. Currently, Bitcoin is designed to set the threshold t so that a single block would be mined in the *entire network* in expectation once every 10 minutes. Hence, a miner with a state-of-the-art mining machine [19] will in expectation wait 687 days to mine a single block [44].

This results in a large variance in rewards across miners — even with a long time horizon of a few months, the majority of miners would get no rewards, while a few miners would get large rewards. Most miners seek to have a steady income stream and wish to reduce the variance in rewards. To do so, miners form teams, called *mining pools*, that share rewards

among the pool members. When a pool member finds a successful input (i.e., an input x where $h(b(x)) < t$) the block is mined and the rewards are distributed among the pool members. As pool participants may be unequal in their computational resources, the rewards are split among them in proportion to their contributed computational power.

Combining powers by forming a pool not only reduces the volatility in the rewards of its participants but also increases their total accumulated revenue. As the pool presents itself to the Bitcoin network as a single powerful node, it is able to gain an advantage over other agents, by prevailing in the case of natural conflicts in the network. Moreover, a recently proposed attack allows a pool that encompasses more than a third of the network's overall computational power to obtain more than its fair share of the rewards, by deviating from the Bitcoin protocol rules [24].

Our contribution: We consider how the existence of mining pools affects the reward allocation in the Bitcoin network. We analyze the effect of skewed rewards, and demonstrate the fact that a pool gains more rewards, in expectation, than its fair share (that which corresponds to the fraction of computational power held by the pool's participants). We show that the non-linear returns depend on the communication delay parameters of the network. This makes decisions regarding which mining pool to join a *strategic choice*, motivating the use of software agents that choose which pool to join so as to optimize payoffs.

We use tools from *cooperative game theory* to study which pools agents may wish to join, and how pool members are likely to share the monetary rewards. We show that due to the non-linear nature of returns, it may be difficult to distribute the pool's rewards among its participants in a stable way: any reward allocation creates an incentive for some miners to leave their pool and join other pools so as to increase their expected reward. Furthermore, we show that this instability becomes more severe as the network processes high transaction loads. We thus expect that as more people adopt Bitcoin, there will be a higher rate of miners switching pools, resulting in a larger overhead for using the system. Our analysis constitutes a practical application of cooperative game theory in the context of automated agents, which might be responsible for decision-making regarding which pool to join, at any moment, so as to maximize payoffs. It illustrates the use of game theoretic tools applied to a real-world software environment.

2. PRELIMINARIES

We start with a brief overview of Bitcoin.[1] Bitcoin is a decentralized crypto-currency. Two kinds of agents participate in the Bitcoin network: *clients*, who trade in the currency, and *miners*, who validate monetary transactions. We focus on interactions among miners. The entire transaction history of Bitcoin is stored on a shared data structure called the *block chain*. Every block in the block chain contains a set of the recent transactions it approves. In addition, its header contains, among other meta-data fields, a pointer to its predecessor in the block chain, a compressed representation of its transaction set, and a "nonce" field which acts as a proof of work. The blocks thus form a tree, rooted at the "genesis block" which was created at Bitcoin's inception,

[1]This is a partial description. A more detailed account can be found in [30] and the Bitcoin Wiki.

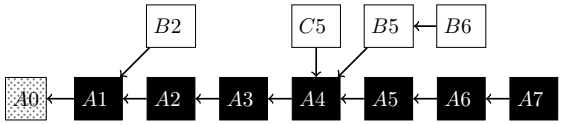

Figure 1: An illustration of a block tree. Blocks in the longest chain from the genesis block (dotted, $A0$) to one of the leaves are black. Blocks that are not in the longest chain are white. The next mined block should refer to block $A7$ as its previous block.

with each block being a child of the block it references in its header.

The process of adding a new block to the block chain is called "mining a block". For a block to be considered valid, the value of the hash of its header must be lower than a target threshold t. The nonce field can then be utilized to modify the hash's result to meet t. The only known method to find a suitable nonce is by random search.

A miner is an agent that continuously tries to mine blocks. The more hashes a miner can compute per time unit, the more likely she is to mine the next block. When a miner mines a valid block, she publishes it to the Bitcoin network; if the block is eventually extended and is part of the *longest chain*, its creator is rewarded with bitcoins.

We assume a fixed reward per mined block. The probability that a single hash based on a random nonce results in a valid block is very low, and the number of mined blocks per time unit of a miner can be well approximated as a Poisson process.

A chain in the block tree is a path from a leaf block to the genesis block. Under the Bitcoin protocol, the longest chain is the only valid chain, so transactions that are not recorded in a block that is contained in the longest chain are not considered valid. Moreover, miners are only rewarded for blocks in the longest chain, so they are incentivized to extend the longest chain of which they are aware. The longest chain rule is designed to allow miners to reach consensus on the state of the chain and to mitigate attacks on the protocol, such as double-spending [37]. We assume that miners are honest and follow the protocol. Figure 1 illustrates a block chain and the longest chain rule. Miners communicate over TCP [17], and *delays* in communication are inevitable. The delay between fellow miners might vary based on geographic location, physical connection, hardware, software and the size of the message [20]. For simplicity, we assume that the delay between every two miners is fixed.

While a new block is created in the whole Bitcoin network every 10 minutes, in expectation, a single agent (with limited resources) is likely to wait a very long period of time before creating a block, due to the size of the network. To achieve more steady and predictable returns on their investment in computational resources, miners collaborate and form teams called *pools*, managed by some *pool manager*. When one of the pool members succeeds in mining a block, the members share its rewards, in proportion to their computational contributions [36].

Internal communication within a pool is relatively efficient, as the pool manager sends to its miners only block headers, while they send back only suitable nonce fields. This is in contrast to communication outside the pool, where agents send entire blocks to one another.

Cooperative Games.

We analyze agent interactions in pools using cooperative game theory. A (transferable utility) *Coalitional Game* [29] is composed of a set of players I, $|I| = n$, and a *characteristic function* $v : 2^I \to \Re$ specifying the monetary value that any coalition (a subset of the agents working as a team) can achieve when cooperating. Intuitively, $v(C)$ is the total utility that the members in $C \subseteq I$ can gain by working together. The characteristic function describes the total payoff of every coalition, but it does not prescribe a way of *distributing* these payoffs among the agents in the coalition. Such a division is called an *imputation*. An imputation is a vector $x \in \Re^{|I|}$ that divides the gains of the *grand coalition* I among all the agents, where $\sum_{i \in I} x_i = v(I)$ and $x_i \geq 0$ is the payoff of player i. The most prominent solution concept that describes stability in coalitional games is the *core* [25]. An imputation x is *blocked* by some coalition $B \subseteq I$ if the members of B can abandon the grand coalition and achieve a higher utility, by working as a group of their own, than the share currently allocated to them; that is, $v(B) > \sum_{i \in B} x_i$. An imputation x is in the core if it is not blocked by any coalition, i.e., for any coalition $C \subset I$ we have that $x(C) \geq v(C)$ (where $x(C) = \sum_{i \in C} x_i$).

Cooperative Games with Coalition Structures.

In many domains *several* teams may form, creating a structure of coalitions. Cooperative games with *Coalition Structures* have been used by artificial intelligence researchers to model agent collaboration and team formation [21, 39, 40, 42]. A coalition structure is a partition of the agent set I into disjoint sets called teams. That is, $\mathcal{S} = \{C_1, \ldots, C_m\}$ is a coalition structure over I iff $\bigcup_{i=1}^{m} C_i = I$ and for all $i \neq j$, $C_i \cap C_j = \emptyset$. We denote by $\mathcal{CS}(I)$ the set of all possible coalition structures over I. In some settings, the value of a coalition depends on the structure of the other coalitions. A cooperative game with coalition structures is defined by a *partition function* [33] that takes as input a coalition structure $\mathcal{S} \in \mathcal{CS}(I)$ and a coalition $C \in \mathcal{S}$ and outputs a value $v(\mathcal{S}, C) = v_{\mathcal{S}}(C)$, i.e., it determines the utility of a coalition C under the partition of other agents, as given by $\mathcal{S}$. Similar to transferable utility coalitional games, agents need to divide the gains of each team in the coalition structure. An imputation associated with $\mathcal{S}$ is a vector $x \in \Re^{|I|}$ such that for all $i \in I$, $x_i \geq 0$, and for every $C \in \mathcal{S}$ it holds that $x(C) = v_{\mathcal{S}}(C)$. An imputation associated with $\mathcal{S}$ is in the *CS-core* of $\mathcal{S}$ if for every $C \subset I$ we have $x(C) \geq v_{\mathcal{S}_C}(C)$, where $\mathcal{S}_C = \{C\} \cup \{\{D \backslash C\} : D \in \mathcal{S}, D \backslash C \neq \emptyset\}$. Intuitively, an imputation is in the CS-core of $\mathcal{S}$ if there is no agent subset that can gain by leaving their teams and forming a new team.

$\mathbb{D}$-Stability.

Requiring that no agent subset can gain by leaving their teams and forming a new team is quite a strong demand. In some situations, not every agent subset can collaborate and form a new team. To address this, we define a *defection function* [2]. A defection function is a function, $\mathbb{D} : \mathcal{CS}(I) \to 2^I$ that associates with each coalition structure in $\mathcal{CS}(I)$ a set of coalitions. Intuitively, for a coalition structure $\mathcal{S}$, and a coalition $C \in I$, we say that the agents in C can defect from $\mathcal{S}$ only if $C \in \mathbb{D}(\mathcal{S})$. An imputation associated with $\mathcal{S}$ is $\mathbb{D}$-*stable* if for every $C \in \mathbb{D}(\mathcal{S})$ we have $x(C) \geq v_{\mathcal{S}_C}(C)$.

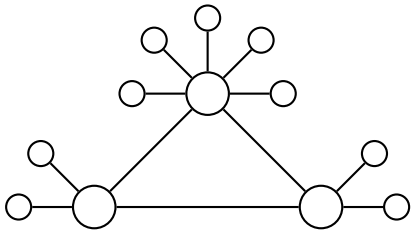

Figure 2: An example of a network with 9 miners and 3 pools. The small circles represent the miners and the large circles represent the pool managers; edges represent overlay connections between nodes.

In addition, we define the $\mathbb{D}$-*CS-core* of $\mathcal{S}$ as the set of all the $\mathbb{D}$-stable imputations that are associated with $\mathcal{S}$.

In this paper we focus on the defection function $\mathbb{D}_{MS}$ that allows one coalition to *merge* with a subset of another coalition, or for a subset of a coalition to *split* from its coalition.

For a coalition structure $\mathcal{S} \in \mathcal{CS}(I)$ we define $\mathbb{D}_{MS}(\mathcal{S}) := \mathbb{D}_M(\mathcal{S}) \cup \mathbb{D}_S(\mathcal{S})$, where $\mathbb{D}_M(\mathcal{S})$ is the set of all possible merges between a coalition and a (possibly trivial) subset of another, and $\mathbb{D}_S(\mathcal{S})$ is the set of all possible splits of a coalition into two (possibly trivial) subsets. Formally, we define $\mathbb{D}_M(\mathcal{S}) := \{C | D_1 \subset C \subset D_1 \cup D_2, D_1, D_2 \in \mathcal{S}\}$, and $\mathbb{D}_S(\mathcal{S}) := \{C | C \subset D, D \in \mathcal{S}\}$.

3. A NETWORK OF MINERS

We model the mining pool interactions as a *miner network*. A miner network is a tuple $\Gamma = \langle \mathcal{M}, \mathcal{S}, \mathcal{P}, D, d, \lambda \rangle$, where $\mathcal{M} = \{1, \ldots, n\}$ is the set of miners; $\mathcal{S}$ is the partition of some of the miners into pools (where each element in $\mathcal{S}$ is a team of miners constituting a single pool); $\mathcal{P} = \{p_1, \ldots, p_n\}$ is the distribution of the computational power among the miners — if p_i is agent i's fraction of computational power then $\forall i \in \mathcal{M}$, $p_i \in [0, 1]$, and $\sum_{i \in \mathcal{M}} p_i = 1$; $D > 0$ is the delay in communication, in seconds, between machines of *different* pools (delay *between pools*); $d > 0$ is the delay between machines in the *same* pool (delay *within a pool*); and λ is the expected number of blocks mined by the network, per second. We assume that every miner $i \in \mathcal{M}$ mines blocks according to a Poisson process with parameter $p_i \lambda$. A miner that does not participate in a pool is referred to as a *solo miner*. In our model, miners in a pool communicate only through the pool manager. Hence, if miner i in pool C_j mined block B at time t, then at time $t+d$ her pool manager knows B was mined. At this point, if B extends the longest chain then the pool manager publishes it to the rest of the network, and updates its participants about the header of the next block to mine. Upon which, at time $t+2d$ the other miners in pool C_j will learn of B, and at time $t + d + D$ the other pool managers and solo miners will learn of B. An illustration of a network with 9 miners and 3 pools is given in Figure 2.

We follow the notations of Sompolinsky and Zohar [43]; for a given miner network, Γ, we define $\beta = \beta(\Gamma)$ to be the rate of block addition to the *longest chain* per second.

For every miner i, we denote by $\gamma_i = \gamma(\Gamma)_i \in [0, 1]$ the probability that a block belonging to the longest chain was mined by miner i. For every pool $C_j \in \mathcal{S}$ we define $\gamma_{C_j} = \sum_{i \in C_j} \gamma_i$. As miners are only rewarded for blocks in the longest chain, a miner i has an incentive to increase γ_i.

LEMMA 1. *Let $\Gamma = \langle \mathcal{M}, \mathcal{S}, \mathcal{P}, D, d, \lambda \rangle$ be a miner network with $|\mathcal{M}| > 1$ miners, $D > 0$ and $d > 0$. For every solo miner i, it holds that $\lambda \geq \beta(\Gamma) > p_i \lambda$.*

PROOF. The left-hand side inequality holds because in the best case, every block is in the longest chain. As for the right-hand side, any sequence of blocks created by the same agent is necessarily of increasing height, hence the growth rate of the longest chain is lower bounded by any solo miner's block creation rate. □

LEMMA 2. *Let $\Gamma = \langle \mathcal{M}, \{\mathcal{M}\}, \mathcal{P}, D, d, \lambda \rangle$, be a miner network with one pool and no solo miners. Then $\beta(\Gamma) = \frac{\lambda}{1+2d\lambda}$.*

PROOF. Under this miner network the pool manager's longest chain coincides with the network's. Let block B be the last block that was created and accepted by the manager as extending the longest chain. Upon its acceptance, it takes d seconds for the manager to update the miners (including B's creator) with the new block-header to mine; any block created during this interval will be ignored by the manager as outdated. After λ^{-1} seconds, in expectation, a new block C is created, which points at B as its predecessor, and it takes an additional d seconds for the manager to learn about C, during which, again, any additional created block will be wasted. Thus the expected time lag between consecutive lengthening of the longest chain is $2d + \lambda^{-1}$ seconds, which implies that the longest chain's growth rate is $\left(2d + \lambda^{-1}\right)^{-1} = \frac{\lambda}{1+2d\lambda}$. □

Retargeting Only transactions on the longest chain are considered valid, so the volume of transactions the Bitcoin network can process is determined by the rate of block addition *to the longest chain*, rather than the total number of blocks mined. Given a target rate β for longest chain growth in a miner network Γ, the protocol parameters are set so that $\beta(\Gamma) = \beta$.[2] This is achieved by determining the threshold t below which the value should be, after the hash. Requiring a larger run of zeros in the value under the hash (i.e., a lower threshold t) makes fewer possible nonces from the space be successful ones, raising the computational burden on the miners. Adjusting the threshold to get a desired value of $\beta(\Gamma) = \beta$ is referred to as *retargeting* [16].

4. THE TWO MINER CASE

We extend the analysis of miner networks with two miners (and no pools) from Sompolinsky and Zohar [43].

THEOREM 3 (SOMPOLINSKY AND ZOHAR). *Let Γ be a miner network with two solo miners. Then*

$$\beta = \beta(\Gamma) = \frac{(\lambda p_1)^2 e^{2Dp_1\lambda} - (\lambda p_2)^2 e^{2Dp_2\lambda}}{\lambda p_1 e^{2Dp_1\lambda} - \lambda p_2 e^{2Dp_2\lambda}} \quad (1)$$

and when $p_1 = p_2 = \frac{1}{2}$ it holds that $\beta(\Gamma) = \lambda \frac{2+D\lambda}{2+2D\lambda}$.

THEOREM 4. *Let Γ be a miner network with two solo miners. Then*

$$\gamma(\Gamma)_1 = \frac{p_1^2 e^{2D\lambda p_1} - p_1 p_2 \left(2 \frac{e^{2D\lambda}-1}{e^{2D\lambda p_1}+e^{2D\lambda p_2}-2} - 1\right)}{p_1^2 e^{2D\lambda p_1} - p_2^2 e^{2D\lambda p_2}} \quad (2)$$

[2]We abuse the notation and use β to denote both the parameter and the function.

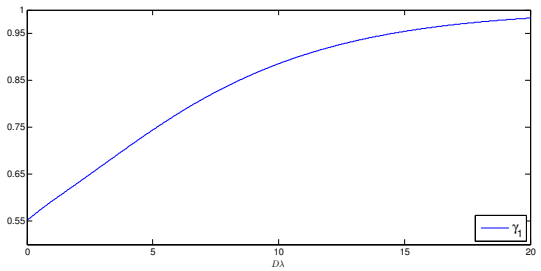

Figure 3: γ_1 as a function of $D\lambda$, for $p_1 = 0.55$, $p_2 = 0.45$.

The theorem implies that $\gamma(\Gamma)_2 = 1 - \gamma(\Gamma)_1$. When $p_1 = p_2 = \frac{1}{2}$, using L'Hopital's rule we get $\gamma_1 = \gamma_2 = \frac{1}{2}$. The proof uses similar techniques to Sompolinsky and Zohar [43], and is omitted due to space constraints.

Corollary 1. Let Γ be a miner network with two solo miners. If $p_i > \frac{1}{2}$ then $\gamma_i \geq p_i$ and if $D\lambda > 0$ then $\gamma_i > p_i$.

Example 1. Consider a miner network with two solo miners, the first with 55% of the hash rate (computational power), and the second with 45%. γ_1 as a function of $D\lambda$ is given in Figure 3. This shows that even when the hash rate is relatively similar among the miners, as $D\lambda$ grows the bigger miner's share in the longest chain, hence in revenues, grows eventually to 100%. Note that if the network suffers no delays ($D\lambda = 0$) the share of every miner is precisely the proportion of computational power held by her.

5. MINING IN COALITION STRUCTURES

We propose a model of the miners and pool interactions as a cooperative game with coalition structures.

Definition 1. A Miner Coalitional Game with Coalition Structures is defined by the tuple $\mathcal{C} = \langle \mathcal{M}, \mathcal{P}, D, d, \beta \rangle$, where $\mathcal{M}$ is the set of miners (players); $\mathcal{P}$ is the distribution of computational power among the miners; D is the delay between pools and d is the delay within a pool; β is the desired rate of longest chain growth. For a given coalition structure $\mathcal{S} \in \mathcal{CS}(\mathcal{M})$, let $\Gamma = \langle \mathcal{M}, \mathcal{S}, \mathcal{P}, D, d, \lambda \rangle$ be a miner network such that $\beta(\Gamma) = \beta$. For every $C \in \mathcal{S}$ we set $v_{\mathcal{S}}(C) = \gamma(\Gamma)_C$. $v_{\mathcal{S}}(C)$ is the pool's share in the longest chain.

We now present our main insight into games with coalition structure: under some quite general assumptions on the partition function v, there is no stable way for coalitions to divide their revenues among their agents. The instability here amounts to the $\mathbb{D}_{MS}$-CS-core of the game being empty.

Definition 2. $w \in \Re^{|I|}$ is a *weight vector* associated with a set of players I, if $w_i \geq 0$ for all $i \in I$ and $\sum_{i \in I} w_i = 1$. For every $C \in I$, we denote $w(C) = \sum_{i \in C} w_i$.

Definition 3. We say that v is:

- **Constant-Sum**, if there exists $c \in \Re^+$ such that for every $\mathcal{S} \in \mathcal{CS}(I)$ it holds that $\sum_{C \in \mathcal{S}} v_{\mathcal{S}}(C) = c$.
- **Nonlinear** with respect to a weight vector $w \in \Re^{|I|}$, if for every $\mathcal{S} \in \mathcal{CS}(I)$ such that $\max_{C \in \mathcal{S}} w(C) > \min_{C \in \mathcal{S}} w(C)$, it holds that for $C_i \in \arg\max_{C \in \mathcal{S}} w(C)$ we have: $v_{\mathcal{S}}(C_i) > w(C_i) \cdot \sum_{C \in \mathcal{S}} v_{\mathcal{S}}(C)$.

- **Monotonic**, if for every $\mathcal{S} \in \mathcal{CS}(I)$, such that $\mathcal{S} = \{C_1, \ldots, C_m\}$ $(m > 2)$ and $v_{\mathcal{S}}(C_1) \leq \cdots \leq v_{\mathcal{S}}(C_m)$: $v_{\mathcal{S}}(C_1) + v_{\mathcal{S}}(C_2) < v_{\mathcal{S}_C}(C)$ for $C = C_1 \cup C_2$.

THEOREM 5. *Let v be a partition function of a coalitional game with coalition structures. If v is constant-sum and nonlinear with respect to a weight vector w satisfying $\forall i \in I : w_i < \frac{1}{2}$, then for every coalition structure the CS-core is empty.*

PROOF. Let v be a partition function as in the conditions of the theorem, let $\mathcal{S} \in \mathcal{CS}(I)$ be a coalition structure of I and let x be an imputation associated with $\mathcal{S}$; we will show that there exists $C \subset I$ such that $v_{\mathcal{S}_C}(C) > x(C)$. Assume without loss of generality that $I = \{1, \ldots, n\}$, and that for every $\mathcal{S} \in \mathcal{SC}(I)$, $\sum_{S \in \mathcal{S}} v_{\mathcal{S}}(S) = 1$.

For every i let $\varepsilon_i = x_i - w_i$ and without loss of generality assume that $\varepsilon_1 \leq \varepsilon_2 \leq \cdots \leq \varepsilon_n$. Note that $\sum_{i \in I} \varepsilon_i = \sum_{i \in I} x_i - \sum_{i \in I} w_i = 1 - 1 = 0$. Let $C = I \backslash \{n\}$. $\varepsilon_n \geq 0$, hence $\sum_{i \in C} \varepsilon_i \leq 0$ and $x(C) \leq w(C)$. $\sum_{i \in C} w_i > \frac{1}{2}$, as $w_n < \frac{1}{2}$. In addition, $v_{\mathcal{S}_C}(C) > w(C)$ because v is nonlinear, therefore $v_{\mathcal{S}_C}(C) > x(C)$. □

The following examples show that all conditions in Theorem 5 are required.

Example 2. Let $w \in \Re^{|I|}$ be a weight vector and consider the game with the partition function: $v_{\mathcal{S}}(C) = 2$ if $\mathcal{S} = \{I\}$ and $v_{\mathcal{S}}(C) = \frac{e^{w(C)}}{\sum_{D \in \mathcal{S}} e^{w(D)}}$ otherwise. v is nonlinear in w but v is not constant-sum. The imputation $x_i = 2w_i$ associated with the coalition structure $\mathcal{S} = \{I\}$ is stable.

Example 3. Consider a game with $|I| = n$ players with the partition function $v_{\mathcal{S}}(C) = \frac{|C|}{n}$ for every $\mathcal{S} \in \mathcal{CS}(I)$ and $C \in \mathcal{S}$. v is constant-sum, however v is not nonlinear as for every weight vector there must be some $C \subset I$, such that $|C| = \lceil \frac{n+1}{2} \rceil$ and $w(C) \geq \frac{|C|}{n}$, yet $v_{\mathcal{S}}(C) \leq w(C)$ for every $\mathcal{S} \in \mathcal{CS}(I)$ such that $C \in \mathcal{S}$. The imputation $x_i = \frac{1}{n}$ associated with every coalition structure is stable.

Example 4. Consider a game with $|I| = n$ players described by the partition function $v_{\mathcal{S}}(C) = 1$ if $1 \in C$, and $v_{\mathcal{S}}(C) = 0$ otherwise. v is constant-sum and nonlinear with respect to the weight vector $w_1 = 0.6$ and $w_i = \frac{0.4}{n-1}$ for $i \neq 1$. The imputation $x_1 = 1$ and $x_i = 0$ for $i \neq 1$, associated with every coalition structure is stable.

While Theorem 5 provides sufficient conditions for the emptiness of the CS-core, it does not imply the emptiness of the $\mathbb{D}_{MS}$-CS-core. Indeed, in some cases there might be an imputation associated with a coalition structure that is not in the CS-core, but is $\mathbb{D}_{MS}$-stable.

Example 5. Consider the game described in Table 1. In the game there are 8 players of three types. The payoff of every coalition in the right column is according to the order of the coalition structure in the left column. The game is constant-sum, because for every $\mathcal{S} \in \mathcal{CS}(I)$, $\sum_{C \in \mathcal{S}} v_{\mathcal{S}}(C) = 100$, and it is nonlinear with respect to, for example, the weight vector $w_a = \frac{1}{20}$ and $w_b = \frac{1}{10}$, $w_b = \frac{3}{10}$. The imputation $(3, 3, 3, 3, 10, 34, 10, 34)$ associated with $\mathcal{S}_1$ is $\mathbb{D}_{MS}$-stable — any merge of two coalitions or a split of a coalition does not result in a better outcome for the deviators.

$\mathcal{S}_1 = \{\{a\}, \{a\}, \{a\}, \{a\}, \{b, c\}, \{b, c\}\}$	$(3, 3, 3, 3, 44, 44)$
$\mathcal{S}_2 = \{\{a\}, \{a\}, \{a\}, \{a\}, \{b, b, c, c\}\}$	$(4, 4, 4, 4, 84)$
$\mathcal{S}_3 = \{\{a\}, \{a\}, \{a\}, \{a\}, \{c\}, \{b, b, c\}\}$	$(4, 4, 4, 4, 30, 54)$
$\mathcal{S}_3 = \{\{a\}, \{a\}, \{a\}, \{a\}, \{b\}, \{b, c, c\}\}$	$(4, 4, 4, 4, 10, 74)$
$\mathcal{S}_4 = \{\{a\}, \{a\}, \{a\}, \{a\}, \{b\}, \{c\}\, \{b, c\}\}$	$(3, 3, 3, 3, 10, 34, 44)$
$\mathcal{S}_5 = \{\{a\}, \{a\}, \{a\}, \{b, c\}, \{a, b, c\}\}$	$(4, 4, 4, 41, 47)$
$\mathcal{S}_6 = \{\{a\}, \{a\}, \{a, a\}, \{b, c\}, \{b, c\}\}$	$(2, 2, 6, 45, 45)$

Table 1: The $\mathbb{D}_{MS}$-CS-core of $\mathcal{S}_1$ is not empty as $(3, 3, 3, 3, 10, 34, 10, 34)$ is $\mathbb{D}_{MS}$-stable

The next theorem provides sufficient conditions for the emptiness of the $\mathbb{D}_{MS}$-CS-core.

THEOREM 6. *Let v be a partition function of a coalitional game with coalition structures. If v is constant-sum, monotonic and nonlinear with respect to a weight vector such that for every $i \in I$: $w_i < \frac{1}{2}$, then for every coalition structure the $\mathbb{D}_{MS}$-CS-core is empty.*

PROOF. Let v be a partition function as in the conditions of the theorem, let $\mathcal{S} \in \mathcal{CS}(I)$ be a coalition structure of I and let x be an imputation associated with $\mathcal{S}$; we will show that there exists $C \in \mathbb{D}_{MS}(\mathcal{S})$ such that $v_{\mathcal{S}_C}(C) > x(C)$. Assume without loss of generality that $I = \{1, \ldots, n\}$, and that for every $\mathcal{S} \in \mathcal{SC}(I)$, $\sum_{S \in \mathcal{S}} v_{\mathcal{S}}(S) = 1$.

If $|\mathcal{S}| \geq 3$, write $\mathcal{S} = \{C_1, \ldots, C_m\}$ such that $v_{\mathcal{S}}(C_1) \leq \cdots \leq v_{\mathcal{S}}(C_m)$. Let $C = C_1 \cup C_2$. The imputation x satisfies $x(C) = x(C_1) + x(C_2) = v_S(C_1) + v_S(C_2)$. As C is a merge of two coalitions in $\mathcal{S}$ we have that $C \in \mathbb{D}_{MS}(\mathcal{S})$. Finally, v is monotonic, thus $v_{\mathcal{S}_C}(C) > v_{\mathcal{S}}(C_1) + v_{\mathcal{S}}(C_2)$, therefore $v_{\mathcal{S}_C}(C) > x(C)$.

If $|\mathcal{S}| \leq 2$, let $\varepsilon_i = x_i - w_i$ (as in the proof of Theorem 5), and assume that $\varepsilon_1 \leq \varepsilon_2 \leq \cdots \leq \varepsilon_n$. Then for $C = I \backslash \{n\}$ we have, $v_{\mathcal{S}_C}(C) > x(C)$. If $|\mathcal{S}| = 1$ then C is the result of a split from the grand coalition, and if $|\mathcal{S}| = 2$ then C is the result of a merge between the two coalitions, hence in both cases $C \in \mathbb{D}_{MS}(\mathcal{S})$. □

LEMMA 7. *Let $\mathcal{C} = \langle \mathcal{M}, \mathcal{P}, D, d, \beta \rangle$ be a Miner Coalitional Game with Coalition Structures for which $D, \beta > 0$, and $\forall i$: $p_i \in (0, 1)$. Assume that d is smaller than some d_1 (a bound that depends on $\mathcal{P}, D$, and β). Then for all $\mathcal{S} \in \mathcal{CS}(\mathcal{M})$ and for all $C_i, C_j \in \mathcal{S}$, if $p(C_i) > p(C_j)$ then $\frac{v_{\mathcal{S}}(C_i)}{p(C_i)} > \frac{v_{\mathcal{S}}(C_j)}{p(C_j)}$.*

PROOF SKETCH. Fix $\mathcal{M}$, $\mathcal{P}$, D and β. We will prove the result for the case $d = 0$, and will then define d_1 to be the maximal positive d which satisfies the required property.

Indeed, let $\mathcal{S} \in \mathcal{CS}(\mathcal{M})$ be a coalition structure, and let $\lambda > 0$ such that $\beta(\Gamma) = \beta$ for the miner network $\Gamma = \langle \mathcal{M}, \mathcal{S}, \mathcal{P}, D, d, \lambda \rangle$. For every $C \in \mathcal{S}$, we denote by $\alpha_{\mathcal{S}}(C)$ the probability that a block that was mined by one of the members of C will end up in the longest chain. Note that for every pool it holds that $\beta \cdot \gamma(\Gamma)_C = \lambda \cdot p(C) \cdot \alpha_{\mathcal{S}}(C)$ as both sides of the equation are the expected number of blocks that were mined by the pool C that are in the longest chain, in every second. Therefore, we need to show that if $p(C_i) > p(C_j)$ then $\alpha_{\mathcal{S}}(C_i) > \alpha_{\mathcal{S}}(C_j)$.

When $d = 0$ there is no delay within the pools and every pool acts as one miner. In this case, if $p(C_i) > p(C_j)$ then $\alpha_{\mathcal{S}}(C_i) > \alpha_{\mathcal{S}}(C_j)$, because in a case of a conflict between the pools, the next block that will be created is more likely to belong to the stronger pool C_i; and because $D > 0$ the probability of a conflict is positive. We can thus define d_1

as the maximal delay within the pools such that for every coalition structure $\mathcal{S} \in \mathcal{CS}(\mathcal{M})$ and for every $C_i, C_j \in \mathcal{S}$, $p(C_i) > p(C_j)$ implies $\alpha_{\mathcal{S}}(C_i) > \alpha_{\mathcal{S}}(C_j)$. □

LEMMA 8. *Let $\Gamma_k = \langle \mathcal{M}, \mathcal{S}, \mathcal{P}, D, 0, \lambda_k \rangle$ $(k = 1, 2)$ be two miner networks, with $\lambda_1 > \lambda_2$, $\mathcal{S} = \{C_1, \ldots, C_m\}$ $(m > 2)$, $p(C_1) \leq \cdots \leq p(C_m)$, and $p(C_1) < p(C_m)$ Then, $\gamma(\Gamma_1)_{C_1} + \gamma(\Gamma_1)_{C_2} < \gamma(\Gamma_2)_{C_1} + \gamma(\Gamma_2)_{C_2}$.*

We omit the quite involved proof due to lack of space.

LEMMA 9. *Let $\mathcal{C} = \langle \mathcal{M}, \mathcal{P}, D, d, \beta \rangle$ be as in Lemma 7. Assume that d is smaller than some d_2 (a bound that depends on $\mathcal{P}, D$, and β). Then for every $\mathcal{S} \in \mathcal{CS}(\mathcal{M})$ with $\mathcal{S} = \{C_1, \ldots, C_m\}$ $(m > 2)$ and $p(C_1) \leq \cdots \leq p(C_m)$, $v_{\mathcal{S}}(C_1) + v_{\mathcal{S}}(C_2) < v_{\mathcal{S}_C}(C)$ for $C = C_1 \cup C_2$.*

PROOF SKETCH. We use a similar technique to the proof of Lemma 7. Fix $\mathcal{M}$, $\mathcal{P}$, D and β, and let $\mathcal{S} = \{C_1, \ldots, C_m\}$ be a coalition structure as in the conditions of the lemma. Let $C = C_1 \cup C_2$ and $\mathcal{S}' = \mathcal{S}_C$. If $p(C_1) = p(C_m)$ then Lemma 7 implies that $v_{\mathcal{S}}(C_1) + v_{\mathcal{S}}(C_2) < v_{\mathcal{S}'}(C)$. Assume therefore that $p(C_1) < p(C_m)$. Let $\lambda_1, \lambda_2 > 0$ such that for the miner networks $\Gamma_{\mathcal{S},1} = \langle \mathcal{M}, \mathcal{S}, \mathcal{P}, D, d, \lambda_1 \rangle$ and $\Gamma_{\mathcal{S}',2} = \langle \mathcal{M}, \mathcal{S}', \mathcal{P}, D, d, \lambda_2 \rangle$, it holds that $\beta(\Gamma_{\mathcal{S},1}) = \beta$ and $\beta(\Gamma_{\mathcal{S}',2}) = \beta$. In addition, consider the miner network $\Gamma_{\mathcal{S},2} = \langle \mathcal{M}, \mathcal{S}, \mathcal{P}, D, d, \lambda_2 \rangle$. Note that $\lambda_1 > \lambda_2$ as $\Gamma_{\mathcal{S}',2}$ is a more concentrated network than $\Gamma_{\mathcal{S},1}$. Using the previous lemma we have: $v_{\mathcal{S}}(C_1) + v_{\mathcal{S}}(C_2) = \gamma(\Gamma_{\mathcal{S},1})_{C_1} + \gamma(\Gamma_{\mathcal{S},1})_{C_2} < \gamma(\Gamma_{\mathcal{S},2})_{C_1} + \gamma(\Gamma_{\mathcal{S},2})_{C_2}$.

We claim that $\gamma(\Gamma_{\mathcal{S},2})_{C_1} + \gamma(\Gamma_{\mathcal{S},2})_{C_1} < \gamma(\Gamma_{\mathcal{S}',2})_C = v_{\mathcal{S}'}(C)$. Indeed, the merge of the two pools eliminates the conflicts between the pools, and in a case of a conflict with another pool, the two pools now have more hash power. We can thus conclude that $v_{\mathcal{S}}(C_1) + v_{\mathcal{S}}(C_2) < v_{\mathcal{S}'}(C)$.

Upon which we define d_2 as the maximal positive d for which this property is satisfied, namely, $v_{\mathcal{S}}(C_1) + v_{\mathcal{S}}(C_1) < v_{\mathcal{S}_C}(C)$ for every coalition structure $\mathcal{S} = \{C_1, \ldots, C_m\}$. □

THEOREM 10. *Let $\mathcal{C} = \langle \mathcal{M}, \mathcal{P}, D, d, \beta \rangle$ be as in Lemma 7, and let d_1 and d_2 be the bounds obtained in Lemma 7 and Lemma 9, respectively. If $d < \min\{d_1, d_2\}$ then the partition function of $\mathcal{C}$ is constant-sum, monotonic and nonlinear with respect to $\mathcal{P}$.*

PROOF. The partition function is constant-sum because for every $\mathcal{S} \in \mathcal{CS}(\mathcal{M}) : \sum_{C \in \mathcal{S}} v_{\mathcal{S}}(C) = 1$. Lemma 9 implies that the partition function is monotonic, as $d < d_2$. Let $w \in \Re^{|I|}$ be the weight vector $w_i = p_i$; we show that the partition function is nonlinear with respect to w. Let $\mathcal{S} \in \mathcal{CS}(\mathcal{M})$ be a coalition structure with $\max_{C \in \mathcal{S}} w(C) > \min_{C \in \mathcal{S}} w(C)$, and let $C_i \in \arg\max_{C \in \mathcal{S}} w(C)$. We have $d < d_1$, and therefore by Lemma 7:

$$\begin{aligned} 1 = & \sum_{C \in \mathcal{S}} v_{\mathcal{S}}(C) = \sum_{C \in \mathcal{S}} p(C) \frac{v_{\mathcal{S}}(C)}{p(C)} \\ < & \sum_{C \in \mathcal{S}} p(C) \frac{v_{\mathcal{S}}(C_i)}{p(C_i)} = \frac{v_{\mathcal{S}}(C_i)}{p(C_i)} \sum_{C \in \mathcal{S}} p(C) = \frac{v_{\mathcal{S}}(C_i)}{p(C_i)} \end{aligned}$$

Consequently, $v_{\mathcal{S}}(C_i) > p(C_i) = w(C_i)$, and the partition function is nonlinear. □

We finally arrive at the main result of this section, the emptiness of the $\mathbb{D}_{MS}$-CS-Core. Combining the previous theorems we conclude:

Corollary 2. Let $\mathcal{C} = \langle \mathcal{M}, \mathcal{P}, D, d, \beta \rangle$ be as in the theorem. If for all i: $p_i \in \left(0, \frac{1}{2}\right)$, then for every coalition structure the $\mathbb{D}_{MS}$-CS-Core is empty.

6. MINING AS A COOPERATIVE GAME

In order to apply the result of the previous section to the Bitcoin world, we need to investigate the behavior of the partition function v. Unfortunately, the topology of the Bitcoin miner network is unknown and keeps changing. To avoid the intractable analysis of the general case, where any topology is possible, we deviate from the coalition structure setup, and model instead the miner and pool interactions as a transferable utility coalitional game. In these games, the value of a coalition depends solely on the members of that coalition, with no dependence on the other players. To adapt this model to the Bitcoin world, where inter-pool effects are possible, we allow only one coalition to form, and fix the topology of its environment as one which consists of solo miners only.

We then investigate the conditions on the network under which the core of the game is empty. The constriction to the simple notion of core means that only the stability of the grand coalition is taken into consideration.

The game's network is denoted by $\mathcal{C} = \langle \mathcal{M}, \mathcal{P}, D, d, \beta \rangle$, as above. For $|\mathcal{M}| > 2$ we make the following simplifying assumptions. First, we assume that all miners hold the same computational power: $p_i = \frac{1}{n}$ where $n = |\mathcal{M}|$. Secondly, we assume the following form of v: for any $C \subset \mathcal{M}$ with $|\mathcal{M}| = 1$, $v(C) = \frac{1}{n}$; $v(\mathcal{M}) = 1$. In order to represent C's value for the general case, we approximate its share of the longest chain, using Equation 2, and assign this approximation of γ_C to $v(C)$.

We apply Theorem 4 to a hypothetical two solo miner network, Γ_2 whose parameters as set as follows. Denote $\alpha_C = \frac{|C|\lambda}{n+2d|C|\lambda}$,and $\alpha_S = \frac{n-|C|}{n}\lambda\frac{n+d\lambda}{n+(n-|C|)d\lambda}$. The first miner in Γ_2 (representing the pool C in the original network) holds a fraction $p_1 = \frac{\alpha_C}{\alpha_C+\alpha_S}$ of the computational power, and the second one (corresponding to the rest of the miners) holds $p_2 = 1-p_2$. The free parameter λ is set so as to satisfy $\beta(\Gamma_2) = \beta$, where $\beta(\Gamma_2)$ is given by Equation 1. Finally, to complete the description of Γ_2, we set $\lambda_2 = \alpha_C + \alpha_S$.

The motivation for this setup is that it applies a divide and conquer method in order to estimate γ_C. We first split the miner network into two sub-networks, the first consisting of the pool, and the second consisting of the solo miners. The growth rate of the longest chain, in the sub-network of the pool, is α_C, as shown in Lemma 2. The sub-network representing the solo miners is a symmetric network with $m = |\mathcal{M} \backslash C|$ miners, and its longest chain's growth rate can be approximated by $\lambda\frac{m+D\lambda}{m+mD\lambda}$ (see Section 7 for further discussion). Putting $m = n - |C|$ we arrive at α_S. Combining these two sub-networks, and approximating their operations as following Poisson processes (with parameters α_C and α_S respectively), we can extract from Equation 2 an estimate for γ_C. Simulation results presented in the next section suggest that this approximation can be done safely, as it underestimates the pool's actual value, and the instability is reached even under weaker conditions.

We are now ready to show that under some constraints on d, for $n > 2$ identical miners, the core of the game is empty. Note that some constraints on d are indeed required to obtain such results. If d is arbitrarily high the pool is highly inefficient and the probability that a block in the longest chain belongs to the pool becomes arbitrarily low, which leaves no incentive for miners to work for the pool. We begin with the $|\mathcal{M}| = 3$ case. We precede the theorem

with a lemma, whose (rather immediate) proof we do not provide here:

LEMMA 11. *Let $\Gamma = \langle \{1,2\}, \emptyset, \{p, 1-p\}, D, \lambda \rangle$ be a miner network. The function $f_{\frac{2}{3}} : \Re^+ \to \left(\frac{1}{2}, \frac{2}{3}\right)$, defined by $f_{\frac{2}{3}}(D\lambda) = p \iff \gamma(\Gamma)_1 = \frac{2}{3}$, is monotonically decreasing.*

THEOREM 12. *Let $\mathcal{C} = \langle \mathcal{M}, \mathcal{P}, D, d, \beta \rangle$ be a Miner Game with $|\mathcal{M}| = 3$ miners. If $d \leq \frac{2-3f_{\frac{2}{3}}(D\beta)}{4\beta f_{\frac{2}{3}}(D\beta)}$, then the core of $\mathcal{C}$ is empty.*

PROOF. Falsely assume that the core is not empty, and let x be an imputation in the core. Without loss of generality we can assume: $x(C) \leq \frac{2}{3}$ for $C = \{1,2\}$. We shall prove that if $d \leq \frac{2-3f_{\frac{2}{3}}(D\beta)}{4\beta f_{\frac{2}{3}}(D\beta)}$ then $v(C) > \frac{2}{3} \geq x(C)$.

Let $\Gamma_2 = \langle \{C, S\}, \{p_C, p_S\}, D, \lambda_2 \rangle$ be the two solo miner network used for the approximation of $v(C)$. Let $\alpha_C = \frac{2\lambda}{3+4d\lambda}$ and $\alpha_S = \frac{\lambda}{3}$ be the rates at which the miners in Γ_2 publish their blocks; let $p_C \propto \alpha_C$, $p_S \propto \alpha_S$ and λ satisfy $\alpha_C + \alpha_S = \lambda_2$, and $\lambda(\Gamma_2) = \beta$.

Now, $p_C = \frac{\frac{2}{3+4d\lambda}}{\lambda_2} = \frac{6}{9+4d\lambda} > \frac{2}{3+4d\beta}$, the last inequality holding due to $\beta > \frac{\lambda}{3}$. As $d \leq \frac{2-3f_{\frac{2}{3}}(D\beta)}{4\beta f_{\frac{2}{3}}(D\beta)}$ it holds that $f_{\frac{2}{3}}(D\beta) \leq \frac{2}{3+4d\beta}$. $f_{\frac{2}{3}}$ is monotonic decreasing, and $\beta \leq \lambda_2$, thus it holds that $f_{\frac{2}{3}}(D\beta) \leq f_{\frac{2}{3}}(D\lambda_2)$. We thus conclude that $p_C > f_{\frac{2}{3}}(D\lambda_2)$. By the definition of $f_{\frac{2}{3}}$ it holds that $\gamma(\Gamma_2)_C = \gamma_C > \frac{2}{3}$. That is, $v(C) > x(C)$, and we arrive at a contradiction to x's belonging to the core. □

The following theorem provides a similar result for the case where $|\mathcal{M}| > 3$:

THEOREM 13. *Let $\mathcal{C}$ be a Miner Game with $|\mathcal{M}| = n > 3$ miners. If $d \leq \frac{D(n-3)}{2(n+1)(1+D\beta)}$ then the core of $\mathcal{C}$ is empty.*

PROOF. The proof is similar to that of the previous theorem. We begin by assuming (in negation) that x is an imputation in the core, and without loss of generality $x(C) \leq \frac{m}{n}$ where $m = \lceil \frac{n}{2} \rceil$ and $C = \{1, \ldots, m\}$. We want to prove that if d is under the specified bound then $v(C) > x(C)$. We now use Γ_2 with α_C and α_S obtaining the values $\frac{|C|\lambda}{n+2d|C|\lambda}$ and $\frac{n-m}{n}\lambda\frac{n+D\lambda}{n+(n-m)D\lambda}$, respectively.

As $\beta > \frac{\lambda}{n}$ and $m \leq \frac{n+1}{2}$, the bound on d implies that $d < \frac{Dn(n-m-1)}{2m(n+D\lambda)}$, and therefore $p_C > \frac{m}{n}$. From Corollary 1 we have: $\gamma(\Gamma_2)_C = \gamma_C > p_C$, hence $v(C) > x(C)$, which implies that x cannot be in the core. □

In contrast to the case where there were at least 3 miners, it turns out that the core of the game when $|\mathcal{M}| = 2$ is not empty, and, in fact, has precisely one element.

THEOREM 14. *Let $\mathcal{C} = \langle \mathcal{M}, \mathcal{P}, D, d, \beta \rangle$ be a Miner Game with $|\mathcal{M}| = 2$ miners. The core of C has precisely one element.*

PROOF. Let $\Gamma = \langle \{1,2\}, \emptyset, \mathcal{P}, D, d, \lambda \rangle$ be the miner network with the two solo miners such that $\beta(\Gamma) = \beta$. Let $\gamma = (\gamma_1, \gamma_2)$ be the proportional shares of the miners in the longest chain. The only possible deviation of the miners from the grand coalition is if they work solo, yielding $v(\{i\}) = \gamma_i$. For every imputation in the core $x = (x_1, x_2)$ we must have $x_i \geq 0$, $x_1 + x_2 = 1$ and $x_i \geq \gamma_i$. Hence, the only solution is $x_i = \gamma_i$ and therefore the core has exactly one element. □

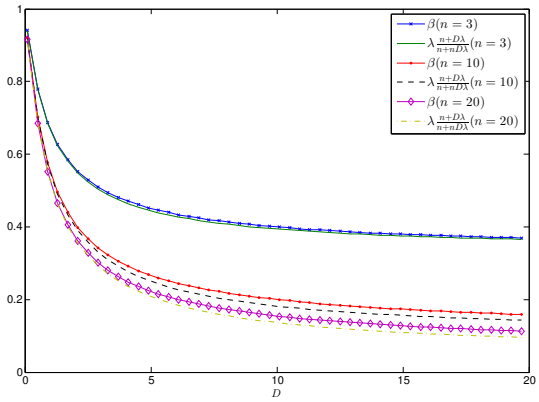

Figure 4: β as observed from simulations, on a symmetric miner network with $n = 3, 10, 20$ miners, where $\lambda = 1$. Compared with $\lambda\frac{n+D\lambda}{n+nD\lambda}$.

7. SIMULATIONS

We ran two experiments simulating mining in an artificial network. In each experiment we fixed the number of miners, the structure of the pools, λ, and the delay inside and outside the pools. In the first set of experiments, we examined β, the growth rate of the longest chain, in symmetric networks with 3, 10, and 20 miners. We fixed $\lambda = 1$ and compared the results to the expression $\lambda\frac{n+D\lambda}{n+nD\lambda}$. As Figure 4 demonstrates, this expression is a good approximation of the longest chain's growth rate in such networks.

In our second experiment we simulated miner networks with $n = 3, 10, 20$ miners, and one pool C with $|C| = m = \lceil \frac{n}{2} \rceil$ miners. We fixed $\beta = \frac{1}{2}$, and λ was set accordingly; d was given the highest possible values, as implied by Theorems 12 ($n = 3$) and 13 ($n = 10, 20$). We examined the pools' *share of the longest chain*, and compared it to $v(C)$, the value we used to estimate γ_C.

Both γ_C and $v(C)$ are presented in Figure 5. In practice, $v(C)$ was found to be a lower bound of γ_C. Recall that in order to prove the emptiness of the core we showed that $v(C) > \frac{m}{n}$. Thus when using the exact value of γ_C instead of the approximation, the core remains empty (under the same constraints on d). This suggests that our results apply more generally, even to networks that our model fails to describe accurately.

8. DISCUSSION

We analyzed block mining in the Bitcoin network, modeling mining pool interactions as a cooperative game. We focused on ways in which miners in pools can share mining revenues, using the core as our game theoretic solution concept. Our results show that a non-linearity of the reward function of the pool is inherent in the protocol.

We showed that in both the model with and without coalition structures, the game is likely to have an empty core. This indicates that no matter how the revenue is shared, some miners would be incentivized to switch to a different pool, as this switch would increase their expected revenue. While our theoretical model makes some simplifications and approximations, simulations corroborate these results.

We note that when the block creation rate is low, the reward non-linearity is small. Currently the Bitcoin network

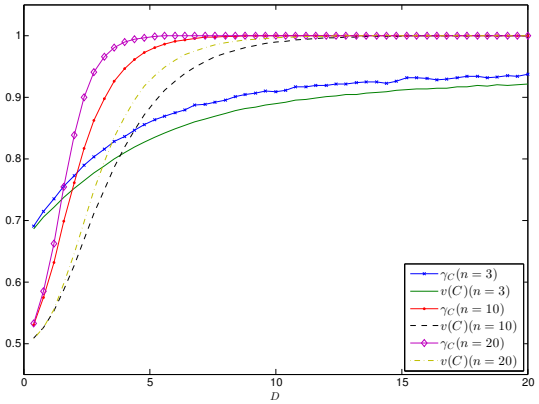

Figure 5: γ_C as observed from simulations, on a miner network with $n = 3, 10, 20$ miners and one pool, C, with $\lceil \frac{n}{2} \rceil$ miners, where $\beta = \frac{1}{2}$. Compared with $v(C)$ — the approximate value of γ_C.

can process up to 3.3 transactions per second, while VISA, for example, can process 47,000 transactions per second [45]. As Bitcoin grows and more transactions are processed, the load on the system would result in a change in parameters increasing the non-linearity effect, making the choice of a mining pool more important. We thus expect more miners to exhibit pool switching behavior, and use software implementations of automated pool choosing strategies.

9. RELATED WORK

Since its 2009 launch, Bitcoin has received attention in the research community. Rosenfeld [37] improved the original security analysis done in Bitcoin's white paper [30]. Babaioff et al. [5] looked at incentive issues related to transaction propagation. Eyal and Sirer [24] showed an attack in which large pools can gain more than their "fair" reward share. Privacy aspects of Bitcoin were the focus of other work [1, 34]. Ron and Shamir [35] analyzed the transaction graph, and tried to identify which accounts belong to the same entity. Sompolinsky and Zohar [43] analyzed the effect of network delays on the growth rate of the blockchain, and suggested a protocol modification that bypasses the trade-off between high block rate and loss of security against double-spending. Lewenberg et al. [26] suggested a modification to Bitcoin's data structure, in the form of directed acyclic graphs, and have analyzed the game theoretic aspects of their proposal.

Closest in spirit to this paper is the work of Niyato, Vasilakos and Kun [31], which models as a cooperative game cloud providers that can cooperate. They show that the solution of the core can be found using linear programming, while in our model the core is empty.

Cooperation among agents has been widely studied in the artificial intelligence literature. Relatively early work, such as that of Sandholm and Lesser [38], analyzed coalitions among self-interested agents that need to solve combinatorial optimization problems to operate efficiently in the world; and Shehory and Kraus [42] considered task allocations via agent coalition formation. We proposed a cooperative game model for analyzing Bitcoin mining pools. Cooperative game models have been used for many applications, including voting [22, 23, 10, 46, 47, 15, 41], auctions [6], negotiation [8], team formation [42, 4, 13, 11], network analysis [28, 7, 14, 12] and even pricing cloud services [18].

Computational aspects of cooperative games have been the focus of other work, such as the work of Elkind et al. [22] that showed that many stability-related solution concepts in weighted voting games are hard to compute, and the work of Aziz and De Keijzer [4] who proposed an algorithm for finding an optimal coalition structure for games with few player types.

Cooperative games with coalition structures were introduced by Aumann and Dreze [3]. In the common practice of cooperative games with coalition structures, so-called characteristic function games, the value of each coalition is independent of nonmembers' actions [39, 32, 21, 27, 9]. Our model is similar to one in Ray and Vohra [33], in which the value of a coalition depends on the coalition structure.

10. CONCLUSIONS

We have examined mining pools in Bitcoin, using a game theoretic model for team formation and reward sharing. We showed that the Bitcoin protocol results in a pool's reward being a non-linear function of the pool's computational power. This results in some inherent instability of the teams, where any reward allocation scheme in a pool is likely to result in some miners wishing to switch pools. Our analysis of relative rewards for agents in various pools constitutes a practical application of cooperative game theory in the context of automated agents, who might make decisions about what pool to join in order to maximize their payoffs.

There are two main avenues for expanding our understanding of collaboration in Bitcoin's miner network. Our analysis made some simplifying assumptions regarding the delays in communication. It would be interesting to see how the results change when the communication delays are not fixed or when the delays are not deterministic. Second, we assumed that for a given coalition structure, all the defections that resulted from merges or splits are allowed. Further work could examine weaker (or more relaxed) stability notions for cooperative games with coalition structures [2].

Acknowledgments

This research was supported in part by Israel Science Foundation grant #1227/12, Israel Ministry of Science and Technology grant #3-6797, and by Microsoft Research through its PhD Scholarship Programme. Aviv Zohar and Yonatan Sompolinsky are supported in part by the Israel Science Foundation (Grants 616/13, and 1773/13), and by the Israel Smart Grid (ISG) Consortium.

REFERENCES

[1] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun. Evaluating user privacy in Bitcoin. In *Financial Cryptography and Data Security*. 2013.

[2] K. R. Apt and T. Radzik. Stable partitions in coalitional games. *CoRR*, 2006.

[3] R. J. Aumann and J. H. Dreze. Cooperative games with coalition structures. *IJGT*, 1974.

[4] H. Aziz and B. De Keijzer. Complexity of coalition structure generation. In *AAMAS*, 2011.

[5] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar. On Bitcoin and red balloons. In *EC*, 2012.

[6] Y. Bachrach. Honor among thieves: collusion in multi-unit auctions. In *AAMAS*, 2010.

[7] Y. Bachrach. The least-core of threshold network flow games. *MFCS*, 2011.

[8] Y. Bachrach, P. Kohli, and T. Graepel. Rip-off: playing the cooperative negotiation game. In *AAMAS*, 2011.

[9] Y. Bachrach, P. Kohli, V. Kolmogorov, and M. Zadimoghaddam. Optimal coalition structure generation in cooperative graph games. In *AAAI*, 2013.

[10] Y. Bachrach, R. Meir, M. Feldman, and M. Tennenholtz. Solving cooperative reliability games. In *UAI*, 2011.

[11] Y. Bachrach, R. Meir, K. Jung, and P. Kohli. Coalitional structure generation in skill games. In *AAAI*, 2010.

[12] Y. Bachrach, E. Porat, and J. S. Rosenschein. Sharing rewards in cooperative connectivity games. *JAIR*, 2013.

[13] Y. Bachrach and J. S. Rosenschein. Coalitional skill games. In *AAMAS*, 2008.

[14] Y. Bachrach and J. S. Rosenschein. Power in threshold network flow games. *JAAMAS*, 2009.

[15] Y. Bachrach and N. Shah. Reliability weighted voting games. In *SAGT*. 2013.

[16] Bitcoin Wiki. Target, November 2012. https://en.bitcoin.it/wiki/Target.

[17] Bitcoin Wiki. Network, June 2014. https://en.bitcoin.it/wiki/Network.

[18] G. Blocq, Y. Bachrach, and P. Key. The shared assignment game and applications to pricing in cloud computing. In *AAMAS*, 2014.

[19] CoinTerra Inc. Protected: TerraMiner IV - 2TH/s, Networked ASIC Miner (Early January Batch), 2014. http://cointerra.com/product/terraminer-iv-2ths-networked-miner-january-soldout.

[20] C. Decker and R. Wattenhofer. Information propagation in the Bitcoin network. In *P2P*, 2013.

[21] E. Elkind, G. Chalkiadakis, and N. Jennings. Coalition structures in weighted voting games. In *ECAI*, 2008.

[22] E. Elkind, L. A. Goldberg, P. Goldberg, and M. Wooldridge. Computational complexity of weighted threshold games. In *AAAI*, 2007.

[23] E. Elkind, L. A. Goldberg, P. W. Goldberg, and M. Wooldridge. On the dimensionality of voting games. In *AAAI*, 2008.

[24] I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Financial Cryptography and Data Security*, 2013.

[25] D. B. Gillies. *Some theorems on n-person games*. PhD thesis, Princeton University, 1953.

[26] Y. Lewenberg, Y. Sompolinsky, and A. Zohar. Inclusive block chain protocols. *Financial Cryptography and Data Security*, 2015.

[27] R. Meir, Y. Bachrach, and J. Rosenschein. Minimal subsidies in expense sharing games. *SAGT*, 2010.

[28] R. Meir, Y. Zick, E. Elkind, and J. S. Rosenschein. Bounding the cost of stability in games over interaction networks. In *AAAI*, 2013.

[29] R. B. Myerson. *Game theory*. Harvard University Press, 2013.

[30] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. *URL: http://www. bitcoin. org/bitcoin. pdf*, 2012.

[31] D. Niyato, A. V. Vasilakos, and Z. Kun. Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach. In *CCGRID*, 2011.

[32] H. Raiffa. *The art and science of negotiation*. Harvard University Press, 1982.

[33] D. Ray and R. Vohra. A theory of endogenous coalition structures. *Games and Economic Behavior*, 26(2):286–336, 1999.

[34] F. Reid and M. Harrigan. *An analysis of anonymity in the Bitcoin system*. Springer, 2013.

[35] D. Ron and A. Shamir. Quantitative analysis of the full Bitcoin transaction graph. In *Financial Cryptography and Data Security*. Springer, 2013.

[36] M. Rosenfeld. Analysis of Bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.

[37] M. Rosenfeld. Analysis of hashrate-based double spending. *arXiv preprint arXiv:1402.2009*, 2014.

[38] T. W. Sandhlom and V. R. Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1-2):99 – 137, 1997. Economic Principles of Multi-Agent Systems.

[39] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 1999.

[40] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Allocating tasks in extreme teams. In *AAMAS*, pages 727–734. ACM, 2005.

[41] A. See, Y. Bachrach, and P. Kohli. The cost of principles: analyzing power in compatibility weighted voting games. In *AAMAS*, 2014.

[42] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165 – 200, 1998.

[43] Y. Sompolinsky and A. Zohar. Secure high-rate transaction processing in Bitcoin. *Financial Cryptography and Data Security*, 2015.

[44] E. Swanson. Bitcoin mining calculator, 2014. https://alloscomp.com/bitcoin/calculator.

[45] Visa Inc. Stress Test Prepares visaNet for the Most Wonderful Time of the Year, 2013.

[46] Y. Zick, A. Skopalik, and E. Elkind. The Shapley value as a function of the quota in weighted voting games. In *IJCAI*, 2011.

[47] M. Zuckerman, P. Faliszewski, Y. Bachrach, and E. Elkind. Manipulating the quota in weighted voting games. *Artificial Intelligence*, 2012.