Rewarding miners: bankruptcy situations and pooling strategies

Marianna Belotti, Stefano Moretti, Paolo Zappalà

Rewarding miners: bankruptcy situations and pooling strategies

Marianna Belotti, Stefano Moretti, Paolo Zappalà

[1] CEDRIC, CNAM, 75003 Paris, France

Caisse des Dépôts, 75013 Paris, France,

marianna.belotti@caissedesdepots.fr

[2] LAMSADE, CNRS, Université Paris-Dauphine, Université PSL, 75016 Paris, France

stefano.moretti@dauphine.fr

[3] LIP6, CNRS, Sorbonne Université, 75005 Paris, France

Politecnico di Milano, 20133 Milano, Italy

paolo.zappala@lip6.fr

Abstract. In Proof-of-Work (PoW) based blockchains (e.g., Bitcoin), mining is the procedure through which miners can gain money on regular basis by finding solutions to mathematical crypto puzzles (*i.e.,* full solutions) which validate blockchain transactions. In order to reduce the uncertainty of the remuneration over time, miners cooperate and form pools. Each pool receives rewards which have to be split among pool's participants. The objective of this paper is to find an allocation method, for a mining pool, aimed at redistributing the rewards among cooperating miners and, at the same time, preventing some malicious behaviours of the miners.

Recently, Schrijvers et al. (2017) have proposed a rewarding mechanism that is *incentive compatible*, ensuring that miners have an advantage to immediately report full solutions to the pool. However, such a mechanism encourages a harmful inter-pool behaviour (*i.e.,* pool hopping) when the reward results insufficient to remunerate pool miners, determining a loss in terms of pool's computational power.

By reinterpreting the allocation rules as outcomes of *bankruptcy situations*, we define a new rewarding system based on the well-studied *Constrained Equal Losses* (CEL) rule that maintains the incentive compatible property while making pool hopping less advantageous.

Keywords: Blockchain, Mining, Mining Pool, Reward, Bankruptcy Situation.

1 Introduction

In the blockchain systems transactions are collected in blocks, validated and published on the distributed ledger. Nakamoto [5] proposed a Proof-of-Work system based on Back's Hashcash algorithm [1] that validates blocks and chains them one to another. The Proof-of-Work system requires finding an input of a predefined one-way function (*e.g.,* hash function) generating an

output that meets the difficulty target. More precisely, the goal for the block validators (*miners*) is to find a numerical value (*nonce*) that added to an input data string and "hashed" gives an output which is lower than the predefined threshold. A miner who finds a *full solution* (*i.e.,* a nonce meeting the difficulty target) broadcasts it across the network.

Miners compete to be the first to find a full solution in order to publish the block and gain a reward consisting in new minted crypto-currencies. Mining is a competitive crypto puzzle (a *mining race*) that participants try to solve as fast as possible. The difficulty D of the crypto puzzle limits the rate at which new transaction blocks are generated by the blockchain (*e.g.,* it takes approximately 10 minutes to find a full solution in the Bitcoin network). This difficulty value is adjusted periodically in order to meet the established validation rate. At the time of writing, in order to validate a block in the Bitcoin blockchain, miners needs to generate (on average) a number $D = 15,47\text{T}$ of hashes.

Mining is a procedure through which miners can gain a substantial amount of money. Nowadays, due to the high difficulty values, solo miners (*i.e.,* miners who work alone with a personal device) find a full solution with a time variance range of billions of years. Small miners survive in this new industry by joining mining pools. A mining pool is a cooperative approach in which multiple miners share their efforts (*i.e.,* their computational power) in order to validate blocks and gain rewards. Once a full solution is found, pool's reward is split among the miners. In this way small miners, instead of waiting for years to be rewarded, gain a fraction of the reward on a regular basis.

Miners' reward is based on their contribution in finding a full solution. In order to give proof of their work, miners submit to the pool partial solutions, *i.e.,* nonces that do not meet the original threshold, but a higher one. The solutions of this easier crypto puzzle are considered "near to valid" solutions and called *shares*. For those blockchains that adopt a SHA-256 function, every *hash value* (*i.e.,* output of the hash function) is a full solution with probability $\frac{1}{2^{32}D}$, and each hash has a probability of $\frac{1}{2^{32}}$ to be a share. Hence, a share is a full solution with probability $p := \frac{1}{D}$.

Miners are rewarded according to the number of shares that they provide. Whenever a share is also a full-solution a block is validated and the pool gains a reward that is split among pool participants according to the number of shares that they have reported.

Mining pools are managed by a pool manager that establishes the way in which miners should be rewarded. Each pool adopts its own rewarding system. There exist several rewarding approaches that can be more or less attractive to miners (see for instance [6]).

1.1 Mining pool attacks

An attack to a mining pool refers to any miner's behavior which differs from the default practice (the honest one) and that jeopardizes the collective welfare of the pool. Rosenfeld provided in [6] an overview of the possible malicious behaviours regarding pools whose profitability depends on their own rewarding mechanism. Miners may attack their pool at the time of reporting their Proof-

of-Work. More precisely they can (*i*) delay in reporting a share (*i.e., block withholding*) and/or (*ii*) report a share elsewhere (*i.e., pool-hopping*).

The former is a practice consisting in delaying in reporting shares and full solutions to a mining pool. This practice implies delaying a block validation and the consequent possession of the reward, that in some cases may be profitable for attackers. Pool-hopping consists in an attack where miners "hop" from a pool to another one according to pools' attractiveness.

1.2 Related works on rewarding mechanisms

The problem for a pool manager is to establish how to redistribute the rewards among pool participants in order to prevent malicious behaviours (as the ones listed above). In other words, the pool manager must choose an "appropriate" rewarding mechanism preventing (possibly, all) different types of attacks. Concerning the block withholding practice, *Schrivers et al.* [7] make use of non-cooperative game theory to propose a rewarding mechanism (denoted as *incentive compatible*) that prevents this attack. This specific rewarding system is robust against malicious actions operated inside a pool, however it does not behave as well in an inter-pool environment since it cannot prevent pool-hopping. In this case, Rosenfeld [6] shows that malicious miners can gain at the expenses of the honest ones, who receive a lower reward than the expected one. In [4] the authors use cooperative games to prove that pool-hopping is not preventable, thus mining pools are not stable coalitions.

Our contribution. Starting from the model in [7], the goal of this work is to propose an alternative incentive compatible rewarding mechanism discouraging the pool-hopping practice. By reinterpreting the reward function in [7] as an outcome of a bankruptcy situation, we construct, analyze and test a new rewarding mechanism adoptable by pools to remunerate contributing miners.

The paper is structured as follows. Section 2 presents the basic model for mining pool and some definitions about bankruptcy situations. In Section 3, we introduce a reward function from the literature, we compare it with a new one (based on a modified version of the CEL rule for bankruptcy situation) and we show that the two are equivalent with respect to incentives in reporting shares or full solutions. Then, in Section 4, we compare these two methods from a multi-pool perspective by showing (also with the aid of simulations) that the CEL-based reward function performs better than the one from the literature in discouraging miners to hop from a pool to another. Section 5 concludes the paper.

2 Preliminaries

2.1 The model

Let $N = \{1, \dots, n\}$ be a finite set of miners. Time is split into *rounds*, *i.e.,* the period it takes any of the miner in the pool to find a full solution. During a round miners participate in the mining

race and report their shares (and the full solution) to the pool manager. Once the full solution is submitted, the pool manager broadcasts the information to the network and receives the block reward B. Then, the pool manager redistributes the block reward B among the miners according to a pre-defined reward function. The round is then concluded and a new one starts. For the sake of simplicity we set $B = 1$.

The situation is represented by the vector $\mathbf{s} = (s_1, s_2, ..., s_n) \in \mathbb{N}^N$, defined as *history transcript*, that contains the number of shares s_i reported by each miner $i \in N$ in a round. Letting $S = \sum_{i \in N} s_i$ be the total number of reported shares, the reward function $R : \mathbb{N}^N \to [0,1]^n$, according to [7], is a function assigning to each history transcript $\mathbf{s}$ an allocation of the reward $(R_1(\mathbf{s}), \ldots, R_n(\mathbf{s}))$, where R_i denotes the fraction of reward gained by the single miner $i \in N$ and $\sum_{i \in N} R_i = B = 1$.

Following the approach in [7], under the assumption of rationality, miners want to maximize their individual revenues over time. Let K be the numbers of rounds that have been completed at time t and let $\mathbf{s}_j$ be the transcript history for any round $j \in K$. Given a reward function R, a miner $i \in N$ will adopt a strategy (*i.e.,* the number of reported shares at each round j) aimed at maximizing her total reward given by

$$\lim_{t \to +\infty} \sum_{j \in K} R_i(\mathbf{s}_j),$$

where a strategy affects both the number of completed rounds and the number of reported shares. In [7], a reward function R is said to be *incentive compatible* if each miner's best response strategy is to immediately report to the pool a share and a full solution. Assuming that (i) one single pool represents the total mining power (normalized to 1) of the network and that (ii) each miner $i \in N$ has a fraction α_i of the hashing power, then the probability for a miner i to find a full solution is α_i. Under this assumption, Schrijvers et al. [7] show that a miner $i \in N$ has an incentive to immediately report her shares if and only if the reward function R is monotonically increasing (*i.e.,* $R_i(\mathbf{s} + \mathbf{e}^i) > R_i(\mathbf{s})$ for all history transcripts $\mathbf{s}$, where $\mathbf{e}^i = (e^i_1, \ldots, e^i_n) \in \{0,1\}^N$ is a vector such that $e^i_j = 0$ for each $j \in N \setminus \{i\}$ and $e^i_i = 1$.). Moreover, they show that a miner $i \in N$, finding a full solution at time t, has an incentive to immediately report it if and only if the following condition holds:

$$\sum_{j=1}^{n} \alpha_j \cdot \big(R_i(\mathbf{s}^t + \mathbf{e}^j) - R_i(\mathbf{s}^t)\big) \leq \frac{\mathbb{E}_{\mathbf{s}}[R_i(\mathbf{s})]}{D} \tag{1}$$

for all vectors of mining powers $(\alpha_i)_{i=1}^n$ and all history transcripts $\mathbf{s}^t$, where $\mathbb{E}_{\mathbf{s}}[R_i(\mathbf{s})]$ is the expected reward for miner i over all possible history transcripts. Condition (1) results from the comparison of the withholding strategy – *i.e.,* $\sum_{\mathbf{s}:S=1} \mathbb{P}(\text{find } \mathbf{s}) \cdot \big(R_i(\mathbf{s}^t + \mathbf{s})\big)$ – with the honest one – *i.e.,* $R_i(\mathbf{s}^t) + \frac{\mathbb{E}_{\mathbf{s}}[R_i(\mathbf{s})]}{\mathbb{E}_{\mathbf{s}}[S]}$ – knowing the fact that the total number of submitted shares in a round, $S = \sum_{i \in N} s_i$, follows a geometrical distribution of parameter $p = \frac{1}{D}$ (*i.e.,* the probability for a share to be a full solution), where D is the difficulty of the crypto puzzle. Therefore, the value of the parameter D corresponds to the average number of submitted shares in a round.

2.2 Bankruptcy situations

We now provide some game-theoretical basic definitions. A *bankruptcy situation* arises whenever there are some agents claiming a certain amount of a divisible estate, and the sum of the claims is larger than the estate. Formally, a *bankruptcy situation* on the set N consists of a pair $(\mathbf{c}, E) \in \mathbb{R}^N \times \mathbb{R}$ with $c_i \geq 0\ \forall i \in N$ and $0 < E < \sum_{i \in N} c_i = C$. The vector $\mathbf{c}$ represents agents' demands (each agent $i \in N$ claims a quantity c_i) and E is the estate that has to be divided among them (and it is not sufficient to satisfy the total demand C).

We denote by $\mathbb{B}^N$ the class of all bankruptcy situations $(\mathbf{c}, E) \in \mathbb{R}^N \times \mathbb{R}$ with $0 < E < \sum_{i \in N} c_i$ A *solution* (also called *allocation rule* or *allocation method*) for bankruptcy situations on N is a map $f : \mathbb{B}^N \to \mathbb{R}^N$ assigning to each bankruptcy situation in $\mathbb{B}^N$ an *allocation vector* in $\mathbb{R}^N$, which specifies the amount $f_i(\mathbf{c}, E) \in \mathbb{R}$ of the estate E that each agent $i \in N$ receives in situation $(\mathbf{c}, E)$.

A well-known allocation rule in the literature is the *Constrained Equal Losses* (CEL) rule, which is defined in the following definition (see, for instance, [3,8] for more details on bankruptcy situations and the CEL rule).

Definition 1 (Constrained equal losses rule (CEL)). *For each bankruptcy situation* $(\mathbf{c}, E) \in \mathbb{B}^N$*, the constrained equal losses rule is defined as* $CEL_i(\mathbf{c}, E) = \max(c_i - \lambda, 0)$ *where the parameter* λ *is such that* $\sum_{i \in N} \max(c_i - \lambda, 0) = E$.

3 Incentive Compatible Reward Functions

Schrijvers et al. [7] introduce a reward mechanism that fulfills the property of incentive compatibility using the identity of the full solution discoverer w. Given a vector $\mathbf{e}^w = (e_1^w, \ldots, e_n^w) \in \{0,1\}^N$ such that $e_i^w = 0$ for each $i \in N \setminus \{w\}$ and $e_w^w = 1$, the incentive compatible reward function R is the following:

$$R_i(\mathbf{s}; w) = \begin{cases} \frac{s_i}{D} + e_i^w\left(1 - \frac{S}{D}\right), & \text{if } S < D \\ \frac{s_i}{S}, & \text{if } S \geq D \end{cases} \qquad \forall i \in N, \tag{2}$$

where s_i is the number of shares reported by miner i, S is the total number of reported shares in a round and D is the crypto puzzle difficulty. This function rewards miner i proportionally to the submitted shares in the case $S \geq D$. On the other hand, in the case $S < D$, each miner receives a fixed reward-per-share equal to $\frac{1}{D}$ and the discoverer w of the full solution receives, in addition, all the remaining amount $1 - \frac{S}{D}$. So, in both cases, $\sum_{i \in N} R_i(\mathbf{s}; w) = B = 1$. Roughly speaking, the reward function R is the combination of two distinct allocation methods. In a *short round*, *i.e.*, when the total amount of reported shares is smaller than the difficulty D of the original problem, the reward function allocates a fixed amount-per-share to all agents equal to $\frac{1}{D}$, but the agent w

who finds a solution is rewarded with an extra prize. Instead, in a *long round*, *i.e.*, when the total amount of reported shares exceeds the difficulty of the problem, the reward function allocates the reward proportionally to the individual shares.

Remunerating miners in a per-share fashion, for long rounds, would lead pool going bankrupt since the reward B results insufficient to pay out all the reported shares. For long rounds, the rewarding mechanism proposed in [7] is nothing more than a solution to a bankruptcy situation. Therefore, it is possible to create new reward functions by simply substituting in long rounds (*i.e.,* in bankruptcy situations) different bankruptcy solutions.

Let us now create a new rewarding mechanism based on the CEL rule defined in Section 2.2 and let us compare the properties of the two allocation methods in long rounds. In order to preserve incentive compatibility we define a CEL-based reward function.

Definition 2. *Given the identity of the full solution discoverer w, the CEL-based reward function $\widehat{R}$ is defined as follows:*

$$\widehat{R}_i(\mathbf{s};w) = \begin{cases} \frac{s_i}{D} + e_i^w\Big(1 - \frac{S}{D}\Big), & \text{if } S < D \\ \frac{e_i^w}{D} + \max\Big(\frac{s_i}{D} - \lambda, 0\Big),\ \lambda : \sum_i \max\Big(\frac{s_i}{D} - \lambda, 0\Big) = 1 - \frac{1}{D}, & \text{if } S \geq D \end{cases} \qquad \forall i \in N,$$

where $\mathbf{e}^w = (e_1^w, \ldots, e_n^w) \in \{0,1\}^N$ is a vector such that $e_w^w = 1$ and $e_i^w = 0\ \forall i \in N \setminus \{w\}$, s_i is the number of shares reported by miner i, $S = \sum_{i \in N} s_i$ is the total number of reported shares in a round and D is the crypto puzzle difficulty.

We assign to agent w, who finds the solution during a long round, an extra prize of $\frac{1}{D}$ to add to the allocation established by the classical CEL rule for the bankruptcy situation $(\mathbf{c}, E) = \big(\frac{1}{D} \cdot \mathbf{s}, 1 - \frac{e_i^w}{D}\big)$, with the estate reduced by $\frac{1}{D}$. More precisely, in long rounds $\widehat{R}_i(\mathbf{s};w) = \frac{e_i^w}{D} + CEL_i\big(\frac{1}{D} \cdot \mathbf{s}, 1 - \frac{e_i^w}{D}\big)$. In other words, it means that 1 is added to the count of the shares s_i reported by the full solution discoverer w. If the value $\frac{s_i}{D} - \lambda$ is negative, by default, the agent w is receiving $\frac{1}{D}$. This incentive is sufficient to make the reward function incentive compatible.

Before proving this statement, let us compare the allocations provided by the classical CEL rule and $\widehat{R}$ through an example with $n = 3$, $D = 10$ and $E = 1$.

Example 1. Given the following bankruptcy situation: $\mathbf{s} = (2, 7, 8)$, miner 1 finds the full solution $(w = 1)$ and $(\mathbf{c}, E) = ((0.2, 0.7, 0.8), 1)$. By Definition 1, it is easy to check that $\lambda = 0.25$, hence:

$$CEL(\mathbf{c}, E) = (0, 0.45, 0.55).$$

Now, consider the new CEL-based rule $\widehat{R}$, a prize of $\frac{1}{D} = 0.1$ is allocated to miner 1, and a new bankruptcy situation $(\mathbf{c}, E')$ arises where the estate is reduced by 0.1; $(\mathbf{c}, E') = ((0.2, 0.7, 0.8), 0.9)$. By Definition 2, now $\lambda = 0.3$ and we have that:

$$\widehat{R}((2, 7, 8); 1) = (0.1, 0, 0) + CEL((0.2, 0.7, 0.8), 0.9) = (0.1, 0, 0) + (0, 0.4, 0.5) = (0.1, 0.4, 0.5).$$

In order to prove that the CEL-based rule is incentive compatible we need to present some preliminary results. More precisely, to express Condition (1) for this new reward function we need to focus on the parameter λ of the definition. This parameter depends on miners' demands and it changes value from round to round. It is important to analyze how the parameter varies if an additional share is found. Let us denote by:

(i) λ_1 the value of the parameter λ when miner i finds the full solution and immediately reports it to the pool and,

(ii) λ_2 the value of the parameter after delaying in reporting the full solution by one additional share. By convention, if miner i finds the additional share the parameter is denoted as λ_2^1, while if any other miner finds it we have λ_2^2.

By analyzing the different values of the parameter λ it is possible to derive the following result:

Proposition 1. *Let us consider* $CEL_i(\mathbf{c}, E) = \max(c_i - \lambda_1, 0)$ *and* $CEL_i(\mathbf{c} + \mathbf{e}_j, E) = \max(c_i^{'} - \lambda_2, 0)$. *For each* $(\mathbf{c}, E) \in \mathbb{B}^N, i, j \in N$ *we have that* $\lambda_1 \leq \lambda_2$.

Proof. Let us report the efficiency condition for the two allocations:

$$\max(c_j - \lambda_1, 0) + \sum_{i \in N \setminus \{j\}} \max(c_i - \lambda_1, 0) = \max(c_j + 1 - \lambda_2, 0) + \sum_{i \in N \setminus \{j\}} \max(c_i - \lambda_2, 0).$$

If $c_j \leq \lambda_1$, efficiency condition implies that $\sum_{i \in N \setminus \{j\}} \max(c_i - \lambda_1, 0) \geq \sum_{i \in N \setminus \{j\}} \max(c_i - \lambda_2, 0)$. Hence, $\lambda_1 \leq \lambda_2$. For $c_j > \lambda_1$ let us assume, by contradiction, that $\lambda_1 > \lambda_2$. The assumption implies that $\sum_{i \in N \setminus \{j\}} \max(c_i - \lambda_1, 0) \leq \sum_{i \in N \setminus \{j\}} \max(c_i - \lambda_2, 0)$. However, $\max(c_j - \lambda_1, 0) = c_j - \lambda_1 < c_j - \lambda_2 < c_j + 1 - \lambda_2 = \max(c_j + 1 - \lambda_2, 0)$ and this leads to contradiction.

Corollary 1. *Given the situation of Proposition 1 we have that:* $\lambda_2 - \frac{1}{D} \leq \lambda_1 \leq \lambda_2$.

Now, we are ready to prove the incentive compatibility of the new reward function based on the CEL rule.

Proposition 2. *The CEL-based reward function* $\widehat{R}$ *of Definition 2 satisfies the property of incentive compatibility.*

Proof. Let us write down Condition (1) for $\widehat{R}$:

$$\alpha_i \left(\frac{1}{D} + \max\left(\frac{s_i + 1}{D} - \lambda_2^1, 0 \right) - \frac{1}{D} - \max\left(\frac{s_i}{D} - \lambda_1, 0 \right) \right) + (1 - \alpha_i) \left(\max\left(\frac{s_i}{D} - \lambda_2^2, 0 \right) - \frac{1}{D} - \max\left(\frac{s_i}{D} - \lambda_1, 0 \right) \right) \leq \frac{\mathbb{E}_s[\widehat{R}_i(\mathbf{s}; w)]}{D}.$$

Since the average reward $\mathbb{E}_s[\widehat{R}_i(\mathbf{s}; w)]$ is positive, the right hand side is positive. Therefore, the condition is fulfilled if the left hand side is not positive.

Due to Proposition 1 and Corollary 1 we have that: $\frac{s_i}{D} - \lambda_2^2 \leq \frac{s_i}{D} - \lambda_1 \leq \frac{s_i + 1}{D} - \lambda_2^1$.

If all the terms in the form $\max(\cdot, 0)$ are positive, then the condition is fulfilled:

$$\alpha_i \left(\frac{1}{D} - \lambda_2^1 + \lambda_1 \right) + (1 - \alpha_i) \left(-\lambda_2^2 - \frac{1}{D} + \lambda_1 \right) \leq \max(\alpha_i, 1 - \alpha_i)(-\lambda_2^1 - \lambda_2^2 + 2\lambda_1) \leq 0.$$

If $\frac{s_i}{D} - \lambda_2^2 \leq 0 \leq \frac{s_i}{D} - \lambda_1$ we get:

$$\alpha_i \left(\frac{1}{D} - \lambda_2^1 + \lambda_1 \right) + (1 - \alpha_i) \left(-\frac{s_i}{D} - \frac{1}{D} + \lambda_1 \right) \leq \max(\alpha_i, 1 - \alpha_i)(-\lambda_2^1 - \frac{s_i}{D} + 2\lambda_1) \leq 0.$$

If $\frac{s_i}{D} - \lambda_1 \leq 0 \leq \frac{s_i+1}{D} - \lambda_2^1$ we get:

$$\alpha_i \left(\frac{s_i + 1}{D} - \lambda_2^1 \right) + (1 - \alpha_i) \left(-\frac{1}{D} \right) \leq \max(\alpha_i, 1 - \alpha_i) \left(\frac{s_i}{D} - \lambda_2^1 \right) \leq \max(\alpha_i, 1 - \alpha_i) \left(\frac{s_i}{D} - \lambda_1 \right) \leq 0.$$

In the end, if all the terms in the form $\max(\cdot, 0)$ are equal to 0, then the condition is fulfilled, since the left hand side is negative.

4 A multi-pool analysis

Pool-hopping consists of a practice in which miners leave a pool to join another one that is considered more attractive in terms of remuneration. More precisely, during a round a miner performing pool hopping (*i.e.,* a *hopper*) stops submitting shares to the pool she was working with at the beginning of the round and starts submitting shares to a different one. A hopper leaves, during a mining race, a pool entering (or already in) a long round for a pool that is currently in a short round. The hopping miner receives an increasing reward from the brand new pool (in short round) and a decreasing reward from the pool left (facing a bankruptcy situation where the resource B is insufficient to remunerate the working miners).

In a multi-pool framework, the total mining power of the network is represented by different mining pools each with its own computational power. Differently from Section 2.1, each miner $i \in N$ is characterized by α_i that now represents a fraction of the pool hashing rate. Indeed, in the single-pool framework, we denote with α_i the fraction of the total hashing power.

Hopping affects the actual rewards of a pool. If a miner performs pool hopping the pool loses computational power and so on average the full solution is found later, *i.e.,* the rounds become longer.

In our multi-pool analysis, we assume that pool hopping is performed at the very beginning of a long round and that miners hop between two pools adopting the same rewarding mechanism. Every mean denoted as $\mathbb{E}[\cdot]$ is considered conditioned to the fact that the miner is in a long round: $\mathbb{E}_{\mathbf{s}}[\cdot | S > D]$. From now on, we mark with an asterisk (*) every variable defining the reward of miners once pool hopping is performed.

4.1 Hopping Analysis on Schrijver's Rewarding Function

When miner i is remunerated with reward function R her incentive to perform pool hopping can be measured as the difference between (i) the average reward when hopping $\mathbb{E}[R_i^*]$ and (ii) the average reward $\mathbb{E}[R_i]$ when working for the pool:

$$\delta_{hop} := \mathbb{E}[R_i^*] - \mathbb{E}[R_i].$$

Proposition 3. *The reward function R proposed by Schrijvers et. al. always gives miners a positive incentive $\delta_{hop} > 0$ to perform pool hopping.*

Proof. As shown in [7], the average reward of an honest miner $i \in N$, *i.e.,* not hopping, is:

$$\mathbb{E}[R_i] = \alpha_i.$$

A hopper (hopping at time t) receives an increasing reward from the new pool in a short round and a decreasing one from the pool left. The sum of the two represents the total reward. On average at the end of a short round ($S = D$) a miner has found $\alpha_i \cdot D$ shares. The round finishes after $D + t$ shares are found, with $t \in [0, +\infty)$, hence the reward for the miner who performs pool hopping is the following:

$$\mathbb{E}[R_i^*] = \sum_{t=0}^{\infty} \left(\frac{\alpha_i t}{D} + \frac{\alpha_i D}{D+t} \right) p'(1-p')^t > \frac{\alpha_i}{1-\alpha_i} > \alpha_i,$$

where $p' = \frac{1-\alpha_i}{D}$ is the probability that a share found by an honest miner is a full solution, t is the time taken by an honest miner (working for the old pool) to find a new share and R_i^* is the reward obtained by a miner who hops from a pool rewarding with R to another pool using the same reward function. Hence, the incentive to perform pool hopping is always positive:

$$\delta_{\text{hop}} = \mathbb{E}[R_i^*] - \mathbb{E}[R_i] > \alpha_i - \alpha_i = 0.$$

A second result deriving from Proposition 3 is the fact that, on average, the hopping miners gain more than their hashing ratio α_i. This has been empirically verified on the Bitcoin network in [2]. The average reward for a hopper, between pools adopting R, can be analytically computed according to the following result.

Proposition 4. *The average reward of miner $i \in N$ hopping between two different pools remunerating miners according to the reward function R is the following:*

$$\mathbb{E}[R_i^*] = \frac{\alpha_i}{1-\alpha_i} + \alpha_i^*,$$

where $\alpha_i^* = \alpha_i(1-\alpha_i)e^{1-\alpha_i}(-Ei(\alpha_i - 1))$ with $Ei(x) = \int_{-\infty}^{x} \frac{e^t}{t}$ denoting the exponential integral.

Proof. On average, a miner with computational power α_i who performs pool hopping receives:

$$\mathbb{E}[R_i^*] = \sum_{t=0}^{\infty} \frac{\alpha_i t}{D} p'(1-p')^t + \sum_{t=0}^{\infty} \frac{\alpha_i D}{D+t} p'(1-p')^t.$$

The first term represents the reward received by the new pool in short round that can be easily computed as follows:

$$\frac{\alpha_i}{D} \sum_{t=0}^{\infty} t \cdot p'(1-p')^t = \frac{\alpha_i}{D} \cdot \frac{D}{1-\alpha_i} = \frac{\alpha_i}{1-\alpha_i}.$$

The second term (denoted as α_i^*) corresponds to the reward assigned by the pool left by the hopping miner. In order to compute this term we need to consider an approximation for $D \to \infty$:

$$\alpha_i^* = \sum_{t=0}^{\infty} \frac{\alpha_i D}{D+t} \frac{1-\alpha_i}{D} \left(1 - \frac{1-\alpha_i}{D}\right)^t \approx \alpha_i(1-\alpha_i)e^{1-\alpha_i} \sum_{t=D}^{\infty} \frac{1}{t} e^{-\frac{1-\alpha_i}{D}t}.$$

The computations can be solved by defining:

$$f(x) := \lim_{D\to\infty} f_D(x) = \lim_{D\to\infty} \sum_{t=D}^{\infty} \frac{1}{t} e^{-\frac{tx}{D}}.$$

Using Lebesgue's theorem and given the constraint $\lim_{x\to\infty} f(x) = 0$ we get:

$$f(x) = -Ei(-x).$$

Hence:

$$\alpha_i^* \approx \alpha_i(1-\alpha_i)e^{1-\alpha_i} f(1-\alpha_i) = \alpha_i(1-\alpha_i)e^{1-\alpha_i}(-Ei(\alpha_i - 1)).$$

Thanks to the result provided by Proposition 4 we note that the shares submitted in the pool left by the hopping miner (α_i^*) represent an important part of her average reward. More precisely, for values of $\alpha_i < 0.39$, α_i^* is more than the 50% of the average reward a miner would have got by not leaving the pool (*i.e.,* her computational power α_i). For instance, if a miner has $\alpha_i = 0.2$ as computational power, she will get $\alpha_i^* \approx 0.11$.

4.2 Hopping Analysis on CEL-based Rewarding Function

Following similar arguments, we can analyze the incentive to perform pool hopping when adopting the CEL-based rule $\hat{R}$. We can, then, compare the results obtained for the reward function R with the ones provided by $\widehat{R}$. We denote as β_i the average reward of function $\widehat{R}$ (corresponding to α_i for function R) that can be computed as follows since the probability for a miner i to find a full solution and to receive the extra prize $\frac{1}{D}$ is α_i:

$$\beta_i = \mathbb{E}[\widehat{R}_i] = \frac{\alpha_i}{D} + \mathbb{E}\left[\max\left(\frac{s_i}{D} - \lambda, 0\right)\right] \quad \lambda : \sum_i \max\left(\frac{s_i}{D} - \lambda, 0\right) = 1 - \frac{1}{D}.$$

Like in Section 4.1, let us define the incentive to perform pool hopping $\widehat{\delta}_{hop} := \mathbb{E}[\widehat{R}_i^*] - \mathbb{E}[\widehat{R}_i]$ and let us compute the average reward received by a hopper:

$$\mathbb{E}[\widehat{R}_i^*] = \sum_{t=0}^{\infty} \left(\frac{\alpha_i t}{D} + \max\left(\frac{\alpha_i D}{D} - \lambda, 0\right)\right) p'(1-p')^t = \frac{\alpha_i}{1-\alpha_i} + \mathbb{E}\left[\max(\alpha_i - \lambda, 0)\right],$$

where $p' = \frac{1-\alpha_i}{D}$, t is the time taken by an honest miner to find a new share and $\widehat{R}_i^*$ is the reward obtained by a hopping miner. Analogously to α_i^* for function R, we denote by β_i^* the reward given by the pool the hopper left:

$$\beta_i^* = \mathbb{E}\left[\max(\alpha_i - \lambda, 0)\right].$$

Hence we have that:

$$\widehat{\delta}_{\text{hop}} = \mathbb{E}[\widehat{R}_i^*] - \mathbb{E}[\widehat{R}_i] = \frac{\alpha_i}{1-\alpha_i} + \beta_i^* - \beta_i.$$

4.3 Comparison of the two Rewarding Functions in a multi-pool framework

We have, now, the metrics to compare the performance of the reward functions R and $\widehat{R}$ in hopping situations. Both rewarding systems present an incentive to hop in long rounds, however the miner rewarded with the CEL-based reward function are less incentivized. It is possible to compare the incentives $\delta_{\text{hop}}, \hat{\delta}_{\text{hop}}$ given by the two functions $R, \widehat{R}$ through the variables introduced in Section 4.2 since:

$$\hat{\delta}_{\text{hop}} \leq \delta_{\text{hop}} \Leftrightarrow \beta_i - \beta_i^* \geq \alpha_i - \alpha_i^*.$$

In order to show that the hopping incentive for the CEL-based reward function is lower with respect to the incentive given by R it is sufficient to prove that $\beta_i - \beta_i^* \geq \alpha_i - \alpha_i^*$.

Proposition 5. *Let N be the ordered set of miners: $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_n$, let us define $\alpha_{>i} := \sum_{j>i} \alpha_j$, as the global computational powers of the miners that are more powerful than α_i. Then, $\beta_i - \beta_i^* \geq \alpha_i - \alpha_i^*$ if $(1-\alpha_i)(\alpha_{>i} - \alpha_i)e^{-\alpha_i + (\alpha_{>i} - \alpha_i)^{-1}}(-Ei(\alpha_i - 1)) \geq 1$ where $Ei(\cdot)$ is the exponential integral function.*

Proof. Given the definitions of β_i and β_i^*:

$$\beta_i = \frac{\alpha_i}{D} + \sum_{t=0}^{\infty} \max\left(\alpha_i + \frac{\alpha_i t}{D} - \lambda, 0\right) p(1-p)^t \quad \text{and} \quad \beta_i^* = \sum_{t=0}^{\infty} \max\left(\alpha_i - \lambda, 0\right) p'(1-p')^t,$$

let us recall that $p = \frac{1}{D}$ is the probability for a share to be a full solution and that $p' = \frac{1-\alpha_i}{D}$ represents the probability for a share reported by an honest miner to be a full solution.

For $t \to \infty$ (*i.e.,* for very long rounds) the function $\max(\cdot, 0)$ either tends to 0 or to 1 since in long rounds eventually the most powerful miner is receiving all the reward ($\max(\cdot, 0) \to 1$) and the other miners are receiving none of it ($\max(\cdot, 0) \to 0$). The limit value 0 is reached for $t = \gamma_i \cdot D$, where $\gamma_i \in \mathbb{R} \cup \{+\infty\}$ is defined as follows; $\gamma_i := \text{argmin}_\gamma\{\frac{\alpha_i t}{D} - \lambda < 0,\ \forall t \geq \gamma D\}$. Roughly speaking, $\gamma_i \cdot D$ represents the number of shares after which miner i is not rewarded.
Hence, it is possible to rewrite β_i and β_i^* in this form:

$$\beta_i = \frac{\alpha_i}{D} + \sum_{t=0}^{\gamma_i D} \left(\alpha_i + \frac{\alpha_i t}{D} - \lambda\right) p(1-p)^t \quad \text{and} \quad \beta_i^* = \sum_{t=0}^{\gamma_i D} \left(\alpha_i - \lambda\right) p'(1-p')^t.$$

The value of γ_i might change if the miner is performing pool hopping, but for the sake of simplicity we approximate by considering the same γ_i in both cases.
Assuming that $\sum_t \lambda \cdot p(1-p)^t \approx \sum_t \lambda \cdot p'(1-p')^t$ we can approximate the difference between β_i and β_i^* as follows:

$$\beta_i - \beta_i^* \approx \frac{\alpha_i}{D} + \sum_{t=0}^{\gamma_i D} \frac{\alpha_i t}{D} p(1-p)^t.$$

Due to the value of the difficulty D, we can consider the limit for $D \to \infty$, then:

$$\beta_i - \beta_i^* \approx \frac{\alpha_i}{D} + \sum_{k=0}^{\gamma_i D} \frac{\alpha_i k}{D} \frac{1}{D} e^{-k/D} = \frac{\alpha_i}{D} + \alpha_i \frac{1}{D^2} \sum_{k=0}^{\gamma_i D} k e^{-k/D} \to \alpha_i(1 - e^{-\gamma_i}(1 + \gamma_i)).$$

Let us now compute explicitly γ_i. If $i = N$ (*i.e.,* $\alpha_i = \text{argmax}_j\{\alpha_j\}$) then $\gamma_i = \infty$. Otherwise at time $t = \gamma_i \cdot D$ the miners who receive a positive reward are all the $j \in N : \alpha_j > \alpha_i$, *i.e.,* all the ones having larger computational power than i. According to the CEL rule definition we get the following balance equation:

$$\sum_{j>i} \left(\alpha_j \left(1 + \frac{t}{D} \right) - \lambda \right) = 1 - \frac{1}{D} \approx 1.$$

Since the time $t = \gamma_i \cdot D$ is the moment when the value $\max(\alpha_i(1 + \frac{t}{D}) - \lambda, 0)$ turns from positive to null, we can say that $\alpha_i(1 + \frac{t}{D}) - \lambda \approx 0$. Therefore we have that:

$$\sum_{j>i} \left(\alpha_j \left(1 + \frac{t}{D} \right) - \alpha_i \left(1 + \frac{t}{D} \right) \right) = 1.$$

Replacing the value of t with $\gamma_i \cdot D$ we get $(\alpha_{>i} - (N - i)\alpha_i)(1 + \gamma_i) = 1$, then:

$$\gamma_i = ((\alpha_{>i} - (N - i)\alpha_i)^{-1} - 1).$$

Now we can find a lower bound for γ_i (since $N - i \geq 1$) and so for $\beta_i - \beta_i^*$:

$$\gamma_i \geq \bar{\gamma}_i := ((\alpha_{>i} - \alpha_i)^{-1} - 1) \Longrightarrow \beta_i - \beta_i^* \geq \alpha_i(1 - e^{-\bar{\gamma}_i}(1 + \bar{\gamma}_i)).$$

The sufficient condition for $\beta_i - \beta_i^* \geq \alpha_i - \alpha_i^*$ is:

$$\alpha_i(1 - e^{-\bar{\gamma}_i}(1 + \bar{\gamma}_i) \geq \alpha_i - \alpha_i^*.$$

We get the statement of the proposition by replacing $\bar{\gamma}_i$ and α_i^* with their explicit formulas.

Thanks to Proposition 5, given miner i's hashing ratio (*i.e.,* α_i) and the power of the miners who are stronger than i (*i.e.,* $\alpha_{>i}$), we can check whether $\widehat{R}$ is giving a lower hopping incentive than the one given by R (*i.e.,* check whether $\hat{\delta}_{\text{hop}} \leq \delta_{\text{hop}}$) by simply applying the sufficient condition introduced above that we denote as $f(\alpha_i, \alpha_{>i})$:

$$f(\alpha_i, \alpha_{>i}) := (1 - \alpha_i)(\alpha_{>i} - \alpha_i)e^{-\alpha_i + (\alpha_{>i} - \alpha_i)^{-1}}(-Ei(\alpha_i - 1)) \geq 1.$$

Let us analyze the hopping performance of R and $\widehat{R}$ in the following example.

Example 2. Given 5 miners ordered according to their hash rates: $\alpha_1 = 0.10$ $\alpha_2 = 0.15$, $\alpha_3 = 0.20$, $\alpha_4 = 0.25$, $\alpha_5 = 0.30$, using the condition provided by Proposition 5 we get:

- $f(\alpha_1, \alpha_{>1}) = f(0.10, 0.90) = 0.59 < 1$, miner 1 has a greater incentive to perform pool hopping if rewarded with $\widehat{R}$ rather than with R;
- $f(\alpha_2, \alpha_{>2}) = f(0.15, 0.75) = 0.66 < 1$, miner 2 has a greater incentive to perform pool hopping if rewarded with $\widehat{R}$ rather than with R;
- $f(\alpha_3, \alpha_{>3}) = f(0.20, 0.55) = 1.24 > 1$, miner 3 has a greater incentive to hop if rewarded with R rather than with $\widehat{R}$;

- $f(\alpha_4, \alpha_{>4}) = f(0.25, 0.30) > 10^6 > 1$, miner 4 has a greater incentive to hop if rewarded with R rather than with $\widehat{R}$;
- $f(\alpha_5, \alpha_{>5}) = f(0.30, 0) \to \infty > 1$, miner 5 has a really low incentive to perform pool hopping if rewarded with $\widehat{R}$.

We can see that miners representing the 75% of the pool's computational power have a lower incentive to perform pool hopping when the CEL-based rewarding mechanism is adopted.

By analyzing function $f(\cdot, \alpha_{>i})$ – *i.e.,* fixing $\alpha_{>i}$ – we can identify the cases in which $\hat{\delta}_{\text{hop}} \leq \delta_{\text{hop}}$ (where $\widehat{R}$ performs better than R). For instance, $f(\cdot, \alpha_{>i}) > 1$ for every $\alpha_{>i} < 0.4$, means that with $\widehat{R}$ not only the miners representing the most powerful 40% of the pool have a lower incentive to perform pool hopping, but also the miner i who just follows in the ranking.

To compare the two reward functions, it is necessary to estimate the percentage of miners who have a lower incentive to perform pool hopping. In Example 2 this percentage is $p(\alpha = \{0.1, 0.15, 0.2, 0.25, 0.3\}) = 75\%$. Formally we would like to estimate:

$$p(\alpha) := \sum_i \alpha_i \cdot \mathbf{1}_{\{\beta_i - \beta_i^* \geq \alpha_i - \alpha_i^*\}}.$$

We know that $p(\alpha) > 40\%$ thanks to the analysis of function f. In order to get a better idea of the range of the value of function p we perform a simulation.

Simulation. Due to the unpredictability of α, we assume that it comes from a random distribution. More precisely, given $X_i \sim U[0, 1]$, α_i is defined as follows: $\alpha_i := \frac{X_i}{\sum_j X_j}$.
We run a simulation with 100 different samples of α for n miners, with $n \in \{3, 10, 20, 30, 50\}$, and estimate the CDF of $p_n(\alpha)$ for every n. We compute explicitly β_i and β_i^*, without using the approximation above introduced.

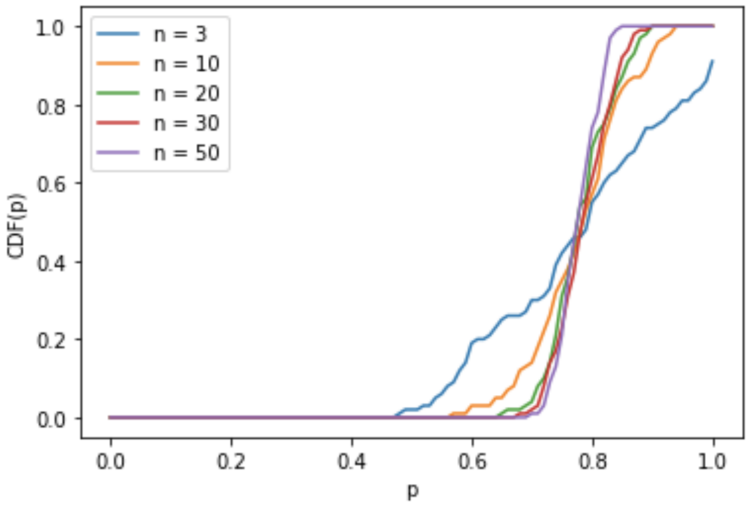

Fig. 1. CDF of every $p_n(\alpha)$, with $n \in \{3, 10, 20, 30, 50\}$.

The functions $p_n(\alpha)$ have *almost always* values over 0.5 (*i.e.,* in just two cases out of 100 with $n = 3$, $p_3(\alpha)$ achieves value between 0.47 and 0.5).
This means that in most of the cases the majority of the miners have a lower incentive to perform pool-hopping with $\widehat{R}$ rather than R.

5 Conclusion

The paper analyzes the robustness of two different rewarding mechanisms in both intra-pool and inter-pool environments. Schrijvers et al. introduce in [7] a reward function R that is incentive compatible. However, this rule gives miners an incentive to leave pools in long rounds to join pools in short rounds that adopt the same rewarding system (*i.e.,* pool-hopping).

By reinterpreting R, in long rounds, as an allocation rule for a bankruptcy situation, we create a new rewarding function $\widehat{R}$ inspired to the well-known Constrained Equal Loss (CEL) rule.

We show that this CEL-based rule is incentive compatible as R but it provides to most of the miners a lower incentive to perform pool hopping in long rounds. In conclusion, if a pool wants to tackle this issue, the proposed rewarding function $\widehat{R}$ is the one to be recommended.

References

1. Back, A.: Hashcash - a denial of service counter-measure (2002)
2. Belotti, M., Kirati, S., Secci, S.: Bitcoin pool-hopping detection. In: 2018 IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI). pp. 1–6. IEEE (2018)
3. Herrero, C., Villar, A.: The three musketeers: four classical solutions to bankruptcy problems. Mathematical Social Sciences **42**(3), 307–328 (2001)
4. Lewenberg, Y., et al.: Bitcoin mining pools: A cooperative game theoretic analysis. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. pp. 919–927. Citeseer (2015)
5. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
6. Rosenfeld, M.: Analysis of bitcoin pooled mining reward systems. arXiv preprint arXiv:1112.4980 (2011)
7. Schrijvers, O., et al.: Incentive compatibility of bitcoin mining pool reward functions. In: International Conference on Financial Cryptography and Data Security. pp. 477–498. Springer (2016)
8. Thomson, W.: Axiomatic and game-theoretic analysis of bankruptcy and taxation problems: an update. Mathematical Social Sciences **74**, 41–59 (2015)