A Formalized Approach to Adaptive Proof of Work Complexity in Decentralized Systems

Lorenzo Abati

June 2021

Abstract

The robustness of decentralized systems, such as the Bitcoin protocol, hinges on their ability to adaptively manage transaction verification times based on available computational power. This paper presents a rigorous mathematical formulation that delineates the adjustment of proof of work complexity in response to the dynamic computational capacities of participating nodes. Drawing from foundational probability axioms, we formally demonstrate the efficacy of our model. While the underlying principles echo mechanisms in existing protocols, this formalized approach offers a deeper analytical insight, potentially guiding enhancements in decentralized transaction verification processes.

Contents

1 Axioms

1.1 Concept of Probability

In this paper, we adhere to the classical or frequentist interpretation of probability. The axioms of probability, as introduced by the Russian mathematician Andrey Kolmogorov, provide the foundation for our discussions. These axioms are:

1. For any event A, the probability $P(A)$ is a non-negative real number:

$$P(A) \geq 0$$

2. The probability of the sample space S is 1:

$$P(S) = 1$$

3. For any countable sequence of mutually exclusive events $A_1, A_2, \ldots$, the probability that at least one of the events happens is the sum of their individual probabilities:

$$P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i)$$

These axioms ensure that the probabilities we assign to events are consistent and meaningful. Throughout this paper, when we refer to the probability of an event, we are invoking these foundational principles.

2 Initial assumptions

2.1 Variables Definition

φ = number of zeros at the beginning of the hash
α = base of the number system
η = number of miners participating in the network
κ = individual miner participating in the network
t = block production interval
λ = computational power of a node

2.2 Event Probability

Consider an imaginary hashing function that generates a digest composed of 2 bits in binary. Let the set of all possible digests be represented by Ω. The probability P_1 represents the likelihood of obtaining a digest with certain predefined characteristics.

Given:

$$\Omega = \{00, 01, 10, 11\}$$

If we decide that our digest should start with 0 (it could also be 1), then the set of acceptable digests, A, is:

$$A_1 = \{00, 01\}$$

Thus:

$$P_1 = \frac{|A_1|}{|\Omega|} = \frac{2}{4} = \frac{1}{2}$$

If any digest from Ω is acceptable, then:

$$A_2 = \Omega$$

In this case:

$$P_1 = \frac{|A_2|}{|\Omega|} = \frac{4}{4} = 1$$

The probability P_1 can be generalized in terms of the number of leading zeros, represented by φ, in the digest as:

$$P_1 = \frac{1}{\alpha^{\varphi}} \tag{1}$$

2.3 Computational Probability

This represents the likelihood that a given computational power will find a hash. For instance, if the computational power or hashrate λ is

$$\frac{1}{10}$$

the computational power will be lower than

$$\frac{1}{5}$$

This is because it took one attempt to find a hash but in half the time, so the higher it is, like

$$\frac{1000}{2000}$$

which is:

$$\frac{1}{2}$$

We want the defined computational power to increase, and the probability should decrease. Thus, the probability P_2 is inversely proportional to the computational power:

$$P_2 = \frac{1}{\lambda}$$

3 Proposed Solution

3.1 Equation

$$\varphi = \log_\alpha \left[\left(\frac{\sum_{\kappa=1}^{\eta} \lambda_\kappa}{\eta} \right) t \right] \tag{2}$$

3.2 Mathematical Demonstration

To have a probability of finding the digest that varies based on the computational power of the network, we need a mathematical system that links a probability P_1 to a probability P_2, where P_1 and P_2 are:

$$P_1 = \frac{1}{\alpha^\varphi} \qquad P_2 = \frac{1}{\lambda} \tag{3}$$

The next step is to equate them so that as the computational power (λ) increases, P_1 decreases.

$$\frac{1}{\alpha^\varphi} = \frac{1}{\lambda} \tag{4}$$

Thus:

$$\alpha^\varphi = \lambda \tag{5}$$

To determine φ:

$$\varphi = \log_\alpha \lambda \tag{6}$$

To modify the time from 1s to ns, it's necessary to multiply λ by a quantity t. In this way, with a computational power of λ, the time to find a hash with complexity φ will be t.

$$\varphi = \log_{\alpha}(\lambda\ t\,) \tag{7}$$

The rest of the equation involves replacing λ with the average computational power of the network (γ), where the summation of the network's computational power is divided by the number of miners to obtain the average computational power.

$$\gamma = \frac{\sum_{\kappa=1}^{\eta} \lambda_{\kappa}}{\eta} \tag{8}$$

The final form is thus:

$$\varphi = \log_{\alpha}\left[\left(\frac{\sum_{\kappa=1}^{\eta} \lambda_{\kappa}}{\eta}\right)\ t\right] \qquad or \qquad \varphi = \log_{\alpha}(\gamma\ t\,) \tag{9}$$

4 Literature Review

A comprehensive review of existing literature reveals that adjusting proof of work complexity based on computational power is a topic of growing interest, especially with the rise of decentralized systems. Several methodologies have been proposed, but a universally accepted solution remains elusive. This research contributes to this ongoing discourse by offering a novel mathematical approach.

5 Conclusions

The derived equation offers a tangible solution for adjusting the complexity of proof of work based on computational power. While its primary application pertains to digest characteristics, the underlying principles can be abstracted for broader applications. Further research is encouraged to refine and expand upon the presented premises.

6 Future Work

Potential avenues for future research include exploring the real-world applicability of the proposed model in diverse computational environments and integrating machine learning techniques to enhance adaptability.