Journal of Construction Project Management and Innovation, 10 (1): 20-32, 2020
ISSN 2223-7852
© Sustainable Human Settlement and Construction Research Centre
DOI: https://doi.org/10.36615/jcpmi.v10i1.351

NONLINEAR PROOF-OF-WORK: IMPROVING THE ENERGY EFFICIENCY OF BITCOIN MINING

Riaan BEZUIDENHOUT[1], Wynand NEL[2], and Andries J. BURGER[3]

[1,2,3]Computer Science and Informatics, University of the Free State, 205 Nelson Mandela Drive, Bloemfontein, South Africa

Email: rbez@mweb.co.za[1]; nelw@ufs.ac.za[2]; andries2410@gmail.com[3]

ABSTRACT
Bitcoin is probably the most well-known blockchain system in existence. It employs the proof-of-work (PoW) consensus algorithm to add transactions to the blockchain. This process is better known as Bitcoin mining. PoW requires miners to compete in solving a cryptographic puzzle before being allowed to add a block of transactions to the blockchain. This mining process is energy-intensive and results in high energy wastage. The underlying cause of this energy inefficiency is the result of the current implementation of the PoW algorithm. PoW assigns the same cryptographic puzzle to all miners, creating a linear probability of success between the miner's computational power as a proportion of the total computational power of the network. To address this energy inefficiency of the PoW mining process, the researchers investigated whether a nonlinear probability of success, between the miner's computation power and its probability of success, will result in better energy usage. A nonlinear proof-of-work (nlPoW) algorithm was constructed by using a design science approach to derive the requirements for and structure of the algorithm. The Bitcoin mining process was tested through statistical simulation, comparing the performance of nlPoW with PoW. Preliminary results, simulating a network of 1000 miners with identical computational power, indicate that nlPoW reduce the number of hash computations, and therefore the energy consumption, required by Bitcoin mining. The findings are significant because nlPoW does not reduce the degree of decentralised consensus, or trade energy usage for some other resource as is the case with many other attempts to address the energy consumption problem in PoW.

Keywords: Bitcoin, bitcoin mining, consensus algorithm, nonlinear proof-of-work, proof-of-work

1. INTRODUCTION
Blockchain systems such as Bitcoin is a public, indelible, transactional data structure shared on a distributed computer network without a central authority (Mulár, 2018). The nodes of the distributed, decentralised computer network follow a pre-determined set of rules (consensus algorithm) to add transactions and reach consensus on the single correct version of the transaction history in the blockchain. This entire process is called Bitcoin mining (Nakamoto, 2008).

In order for blockchain systems to work as distributed systems, they require consensus algorithms to function under the assumption that a limited number of the components may be faulty. It is critical that distributed systems are able to agree on a piece of data (the transaction record in this case) even if part of the system is unreliable (Fischer, 1983). Section 3.4 deals with different consensus algorithms, and Section 5 places these into context with the algorithm that is proposed in this research.

Different blockchain systems all share certain essential components, although they differ in their intended application and in the architecture used to achieve their aims and objectives. Section 2 describes the basic blockchain components as introduced by Nakamoto for what was to become Bitcoin (Nakamoto, 2008) and elaborated upon by Zheng *et al.* (2017).

The Bitcoin mining process is extremely energy-intensive and results in high energy wastage. The underlying cause of this energy inefficiency is the result of the original implementation of the proof-of-work (PoW) algorithm, as discussed in Section 3. For this reason, the researchers studied the effect of a nonlinear proof-of-work (nlPoW) algorithm on the energy usage of the Bitcoin network.

2. BITCOIN BLOCKCHAIN DATA STRUCTURE

The blockchain data structure consists of a sequential series of data blocks that are cryptographically linked. Each block consists of three main components or sub-blocks, namely the transaction block, block header and block hash (BitcoinProject, 2020). These components will be discussed in Sections 2.1, 2.2 and 2.3, respectively.

2.1 Transaction Block

The transaction sub-block contains a set of transactions to be added to the blockchain. In Bitcoin, this set of transactions is structured as a Merkle tree (

Figure 1) where each parent hash contains a hash of its children. This process culminates in a root hash that contains a signature unique to the data contained in all the transactions and which would fail to be reproducible if any of the information in any of transactions were to be changed (BitcoinProject, 2020; Nakamoto, 2008).

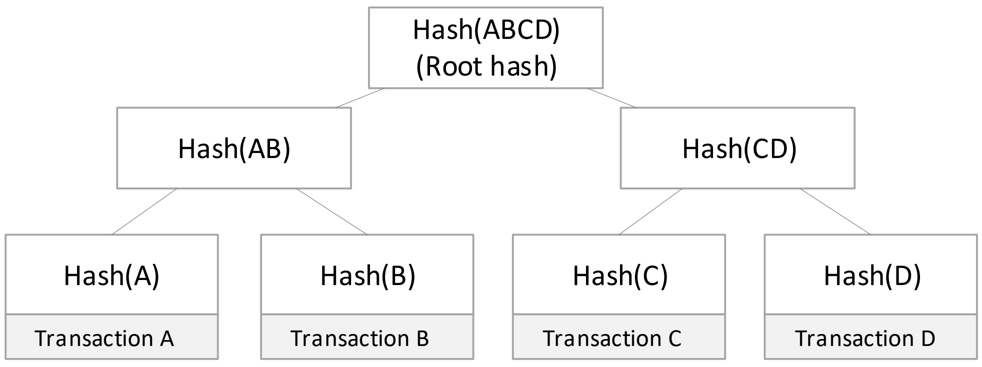

Figure 1: Transaction Merkle tree (Nakamoto, 2008)

In Bitcoin, the first transaction in the transaction Merkle tree is called the coinbase transaction. This transaction furthermore has the specific function of creating new Bitcoins and paying them to the mining node as a mining reward (Antonopoulos, 2014; BitcoinProject, 2020; Narayanan *et al.*, 2016).

2.2 Block Header

In its simplest form, a blockchain is a tamper-evident log of transactions. This means that each block must contain the transactions, a hash pointer to the previous block and a timestamp (Mulár, 2018). Nakamoto (2008) showed that it is not necessary to calculate the hash of the entire block, including all the transactions. The block header can, instead, be constructed to contain only the root hash of the transaction Merkle tree including the previous block hash and timestamp. Depending on the specific blockchain, some other block data may also be included in the block header. Note that the block hash is computed only from the block header. The Bitcoin block header contains the fields, as indicated in Table 1.

Table 1: Bitcoin block header fields (BitcoinProject, 2020)

Field	**Contents**
Previous block hash	Block hash calculated during previous block round.
Transaction Merkle tree root hash	The root hash of the transaction Merkle tree.
Timestamp	Timestamp when the miner started mining the block – according to the miner's clock.
Consensus algorithm software version	Specifies the set of validation rules used to create the block.
Number used only once (nonce)	A number, typically starting at zero and increased by one for every hash calculation. This produces a new hash while keeping all the other header fields unchanged.
nBits field / Static target (difficulty parameter) that all miners must solve	A binary number that indicates the maximum value that the hash may be. The miner must keep on changing the nonce and re-hashing the block header until the hash is strictly smaller than the target (difficulty parameter).

2.3 Block Hash

The block hash is the hash of the block header data and must conform to the rules of the consensus algorithm (Section 0). These rules include a calculation difficulty (in the form of a target) that the mining node must prove it reached during the construction of the block. Although it is challenging for the mining node to compute a suitable hash value, it is straightforward for any other node to inspect the blockchain and confirm that the mining node did indeed do the required work (Zheng *et al.*, 2017).

Figure 2 depicts the components of a Bitcoin block and shows how the cryptographic link between blocks are formed by including the hash of the previous block header in the hash of each new block.

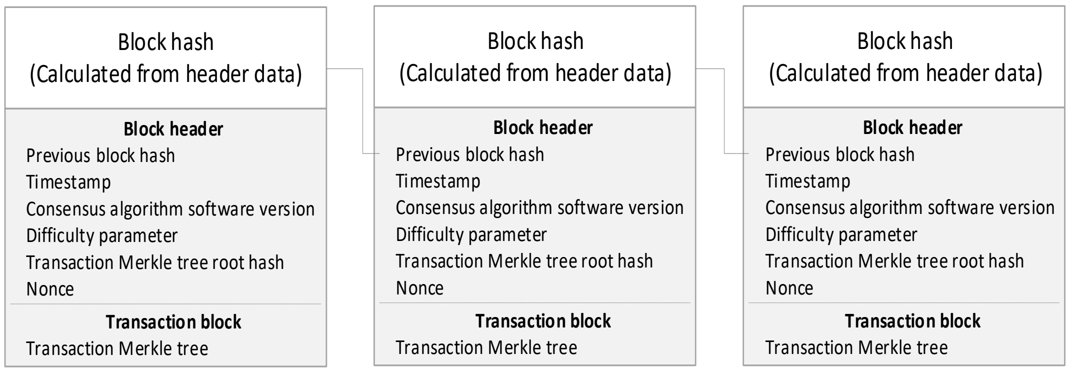

Figure 2: Structure of the Bitcoin blockchain (Zheng *et al.*, 2017)

3. BITCOIN CONSENSUS ALGORITHM

A blockchain consensus algorithm is a detailed set of rules that miners follow in order to mine new blocks and to agree amongst themselves which version of the blockchain is considered the valid one (Zheng *et al.*, 2017). The focus of this paper is on the Bitcoin PoW consensus algorithm, which is discussed in Section 3.1. In Section 3.2, the valid chain selection process is explained, followed by the side-effects of PoW in Section 3.3. The enormous energy wastage of the Bitcoin network, which is also the problem that is investigated in this paper, is discussed in Section 3.3.1.

3.1 Proof-of-Work

PoW is an algorithm for determining which miners managed to create valid new blocks. This entails the calculation of a hash value of sufficient difficulty from the block header data, including the transaction Merkle tree root hash, which serves as a "cryptographic summary" of all the transactions in the block. The target is selected so that it requires each miner to calculate an enormous number of hashes, each time using a new nonce to render a different hash, before finding a suitable block hash (Antonopoulos, 2014; Nakamoto, 2008; Narayanan *et al.*, 2016).

One of the built-in rules of Bitcoin's PoW is that the network recalculates the target every 2016 blocks so that it takes on average 600 seconds (10 minutes) to mine a block. The calculation adjusts the target by the factor that the average block addition time is less than or more than 10 minutes and has the effect that the difficulty changes as the computational power of the network changes. Both the 10-minute average block addition time and the 2016 block target recalculation interval (at 10-minute intervals the mining of 2016 blocks equates to two weeks) are arbitrary values that were introduced by Nakamoto in the original Bitcoin software (Nakamoto, 2008). Any other values could substitute these values. Historically, the Bitcoin block addition times mostly started at 10 minutes and decreased over the span of 2016 blocks (which then also took less time to complete than the theoretical two weeks), as new computational power was continuously added to the network. This decrease in block addition times continued until such time that the target was recalculated (Antonopoulos, 2014; Narayanan *et al.*, 2016).

One of the core innovations of Bitcoin is that any miner is allowed to add a block to the blockchain with the simple rule that the first valid block announced, is the correct one. In practice it does happen, due to the latency of data propagation, that more than one valid block is found by the network, bringing multiple branches of the blockchain into existence. This is called a fork in the blockchain and has the result that different partitions of the network may mine onto different branches (Eyal *et al.*, 2016).

As one of the forks eventually becomes longer than the other, it will eventually replace all other versions of the blockchain on the network (Bonneau *et al.*, 2015). It can be inferred that transactions that were contained in the shorter fork will then be included in future blocks on the longer chain.

3.2 Valid Chain Selection and Transaction Security

A critical requirement of any blockchain system is that the network of miners eventually reaches an agreement on only one valid version of the blockchain, discarding all other forks. In Bitcoin, this is done through the longest chain rule. The fork that is mined by the largest network partition (by computing power) will eventually grow longer (will contain more blocks) than forks that are mined by smaller partitions (by computing power) (Nakamoto, 2008). Nakamoto (2008) referred to this principle in the context of the proportion of honest to dishonest miners, but it applies to network partitions in general.

When a miner receives a version of the blockchain that is longer than the one it is currently mining, it will discard its current blockchain and replace it with the longer one. This process ensures that only the longest blockchain survives and becomes the only valid blockchain accepted by all miners (Zheng *et al.*, 2017). As a block becomes buried under more blocks in the blockchain, it becomes increasingly unlikely that an adversary will be able to apply enough resources to change a transaction in that block, improving the confidence of a payee that a received payment is secure (Bentov *et al.*, 2016).

3.3 The Side-effects of Proof-of-Work

Ma *et al.* (2018) note that finding a suitable hash requires no strategy by a miner, but relies on a form of brute force guessing (making a large number of guesses, each time with another nonce in the header, until a suitable hash is discovered) to solve a computational puzzle. The more computational power commanded by a miner, the more likely it is to solve the puzzle first and claim the prize, although the process remains probabilistic and does not guarantee that the most powerful miner will prevail.

Dimitri (2017) presents a framework whereby each round of the Bitcoin mining process can be viewed as an all-pay contest. An all-pay contest is a competition where the award or prize is known in advance and require everyone to make an investment to participate. Having miners play this type of game has two important desirable effects on the blockchain ecosystem, whereby *firstly*, it results in very few (preferably one) suggested new blocks to be found during each round and *secondly*, reducing the ability for bad actors to take back already spent Bitcoins. This is important as there is no prohibition on anyone to participate as a Bitcoin miner (Ma *et al.*, 2018). On the other hand, brute force guessing and unprohibited participation have led to one of the main undesirable side effects of Bitcoin mining, namely energy wastage.

3.3.1 Energy Wastage by Bitcoin Mining

The incentive structure of Bitcoin's PoW consensus scheme requires that all the miners in the network calculate an enormous amount of hashes in the race to be the first to produce one suitable block hash (Ma *et al.* 2018). This translates into an unsustainable level of energy usage. As the computing power of the network increases, resulting in a higher hash rate by the network in general, the target of the block hash is automatically adjusted downward (difficulty is increased) by the consensus algorithm. This is done to keep the average block creation time constant, but in turn, results in mining entities adding ever more computing power to remain competitive (Tschorsch & Scheuermann, 2016). Ma *et al.* (2018) point out that as the value of Bitcoin has risen over the years (2009 to 2018 were the years covered by their research), the reward from mining has become more substantial, resulting in drawing more participants and consequently, increasing energy usage.

According to the Digiconomist (2020), the Bitcoin network consumed an estimated 66 terawatt-hour (TWh) of electricity per year in August 2020. This is comparable with the energy consumption of The Czech Republic, a country with 10.7 million inhabitants (July 2020 estimate) (Central Intelligence Agency, 2020).

Only a single block, mined by a single miner eventually survives to permanently become part of the blockchain and all the energy expended by unsuccessful miners, which is the vast majority, goes to waste (Zheng *et al.*, 2017). Tschorsch and Scheuermann (2016) state that, given the same block hash target for each miner and given that all miners aim to come up with a solution, the chance of solving the block hash is proportional to the miner's fraction of the total computing power. In other words, the probability of successfully mining a block is linear to the computational power, as explained by (Dimitri, 2017):

Assume a network where each miner is denoted as i and: $i = 1, 2, .., n$

The hash power of each miner is denoted as: h_i

Thus, the hash power of the entire network is denoted as: h_n

Then, the probability that a miner is the first to solve the block hash is:

$$\mathrm{P}(h_i) = \frac{h_i}{h_n} \qquad (1)$$

Since the total computing power of the network can be considered to be a constant, *C*, during the 600-second interval of each block round (Section 3.1), equation 1 becomes a linear equation:

$$\mathrm{P}(h_i) = \frac{h_i}{C} \qquad (2)$$

and the probability for any miner i, becomes linear to its proportion of computing power (h_i) in the network.

3.4 Related work

Many alternative types of consensus schemes have been proposed for blockchain systems, which can broadly be categorised into *proof-based* and *voting-based* algorithms (Nguyen & Kim, 2018). Not all of these focus on addressing the shortcoming of PoW. Voting-based algorithms forego decentralised consensus in favour of improved trust assumptions about the network, afforded by a controlled pool of participants. These types of algorithms fall outside the scope of this research as the primary motivation for PoW is to maintain a completely decentralised consensus.

Some proof-based algorithms have made attempts to lower resource usage during mining, most notably, proof-of-stake, proof-of-importance (Bentov *et al.*, 2016; NEM, 2018; Zheng *et al.*, 2017) and delegated proof-of-stake (BitShares Blockchain Foundation, n.d.; Hasib, 2018). There have also been attempts to exchange energy for other resources during the mining process, these types of solutions include PoW with a cuckoo hash function (relies on memory rather than computational power) (Tromp, 2014), proof-of-human-work (requires inputs that are easy for humans to provide, but difficult for computers) (Blocki & Zhou, 2016), proof-of-burn (uses blockchain tokens) (Jenks, 2018; P4Titan, 2014), proof-of-space (relies on data storage capacity) (Dziembowski *et al.*, 2015), proof-of-entanglement (depends on the quantum properties of light) (Bennet & Daryanoosh, 2019), proof-of-elapsed time and proof-of-luck (uses centralised random number generation) (Nguyen & Kim, 2018) and proof-of-responsibility (relies on trusted nodes) (Coinspace.com, 2019).

Consensus algorithm development is an ongoing process, as new ideas are constantly being added and tested (Zheng *et al.*, 2017). In the discussion section, the solution proposed in this paper will be placed in context with these existing alternatives.

4. PROPOSED SOLUTION

In Section 0.3.1, the high energy wastage of PoW was identified as a major concern for the long-term sustainability of the Bitcoin network. The energy consumption is related to the amount of computation required and is measured in hashes per second per watt (H/s/W) (Narayanan *et al.*, 2016). This energy wastage was linked to the fact that each miner's probability of success is linear to the proportion of its computational power. If this probability of success could be changed to be nonlinear to the computational power of each miner, it is reasonable to argue whether it would have an effect on the energy consumption of the network as a whole. In other words, if every miner received a random target during each block round, will it reduce the energy consumption of the mining network?

The researchers propose a nonlinear proof-of-work (nlPoW) algorithm for assigning a dynamic target (random target) to each miner during each block round (Section 4.1). Furthermore, the researchers want to provide initial indications of the viability of the solution through the simulation of the mining process under nlPoW (Section 4.2). The rationale for this approach is two-fold, *firstly* it is expected that miners that receive a large target (small difficulty) in comparison to other miners may be able to solve the block relatively quickly with fewer computations, thereby saving the amount of energy expended by the network (as soon as a new block is announced, the rest of the network stops mining their current blocks and starts the process of mining new blocks that follow on the newly announced one). *Secondly*, there may arise circumstances where miners who receive a relatively small target (large difficulty) are discouraged from taking part in a block round altogether.

The implication of the first rationale, should it hold, is based on the fact that the average block addition time as set by the original algorithm (Section 3.1), may decrease. As stated before, the 10-minute average block addition time in the original Bitcoin algorithm is an arbitrary value, but it does form the basis of the periodic target adjustment to account for changes in the computing power of the Bitcoin network. The nlPoW algorithm must retain the ability to account for changes in network computing power (Section 4.3.1).

It is very difficult to envisage the implications of the second rationale as it is impossible to predict the circumstances and motivations that drive the decision-making process of individual miners. For the purposes of this study, a simple assumption will be made that a miner will participate as long as its dynamic target is greater than, or equal to the static target (difficulty parameter) during any given target adjustment interval (Section 4.1). If the dynamic target is less than the static target, the miner will not participate.

4.1 Assigning a Random Target

Assigning a dynamic target to each miner entails generating a uniformly distributed random number in the range $0\ to\ 2^{256}$ (the possible outcomes of the SHA256 hash algorithm). Stark and Ottoboni (2018) propose that cryptographic hash functions may be useful as pseudo-random number generators. The unpredictability and collision-resistant properties of hash functions make them suitable for pseudo-random number generators.

For the purposes of this research, the cryptographic hash function provides an elegant solution to generating a pseudo-random number, but it will require a seed (input value) with specific characteristics. It is important that the seed is determined during the design of the algorithm and that it conforms to a number of guidelines that the algorithm must adhere to when selecting the seed.

Firstly, the seed must be unique to each miner so that the target calculated for each miner is different. *Secondly*, the information for selecting the seed must be encoded on the blockchain to allow other miners to confirm the target calculated by the successful miner and by implication, the validity of the block hash produced. *Thirdly*, the seed must change every time a new block is constructed so as not to provide a permanent advantage or disadvantage to any individual miner. *Finally*, since each miner is free to choose any strategy to its advantage, it must not be possible for the miner to manipulate the seed (generate many seeds) until it produces an easy target for itself.

One solution to this problem is to construct the seed from the miner's coinbase address and the previous block hash with the proviso that the coinbase address must contain a previous transaction on the blockchain (the coinbase address may not be created at the same time as the block containing it).

4.2 Simulation of Dynamic Target Assignment

In order to investigate the proposed solution that assigning a dynamic target from a uniform distribution to each miner, during each block round, will reduce the number of computations performed by the network to mine a new block, the mining process was simulated. During the simulation, a random target was awarded to each miner by calculating the SHA256 hash of the miner's address in conjunction with a fictional previous block hash. The address for each miner was unique, and the previous block hash was identical for all miners.

By dividing the result of the SHA256 hash function above by 10000 and taking the remainder, a uniformly distributed random number ranging from zero to 9999 is produced. This number can be standardised to produce a uniformly distributed random number (to four decimals) in the range (0,1] (zero excluded and 1 included) by adding one and dividing by 10000.

This paper details the simulation results from two methods of using a uniformly distributed, standardised number to generate a random target for each miner, namely unidirectional random target assignment (Section 4.3) and the bidirectional random target assignment (Section 4.4). For the purpose of simulation, the state of the Bitcoin network was selected from block 592583 on 31 August 2019 (BTC.com, 2019).

4.3 nlPoW: Unidirectional random target assignment

Unidirectional random target assignment is a "one-way" adjustment of the static target, using the random number calculated (Section 4.2) for each miner, in order to arrive at the dynamic target. The static target is divided by the standardised uniform random variable to produce a dynamic target that is different (larger than the static target) for each miner, unpredictable and changes with each block round. The change in the random number is precipitated by the fact that the previous block hash changes (unpredictably) with each new block. This dynamic target is unrelated to the computational power of the miner and cannot be predicted in advance. The unpredictability in conjunction with the variability of the target means that there is no linear probability between the miner's computational power and its probability of solving the block. Table 2 shows a sample of the results from a simulation of unidirectional dynamic targets for 1000 miners from a uniform distribution, representing one block round.

Before the simulated random numbers could be applied to produce a dynamic target for each miner, it was necessary to determine if the values in (The "Standardised random variable" column, in Table 2) were indeed random. For this purpose, the Kolmogorov-Smirnov test (Accord, 2017) was used, and the following null hypothesis was formulated:

H_0: The series of standardised random variables (column 2 in Table 2) is random.

At the confidence level of $\alpha = 0.05$, the null hypothesis was not rejected, and the series of standardised random variables were assumed to be random.

Table 2: Sample of simulated results for unidirectional dynamic targets

Miner number	**Standardised random variable**	**Nonlinear target**	**Estimated number of hashes required by a miner**	**Static target**	**Estimated number of hashes required by the static target**
	A random value in the range (0,1]	The dynamic target for this miner in this block round. Static target divided by a standardised random variable	Estimated number of hashes required by a miner to find a hash smaller than the nonlinear target	Original target encoded in the block header - nBits parameter in Table 1	Estimated number of hashes required by a miner to find a hash smaller than the static target
0	0.5514	4.8015E+054	2.41E+022	2.65E+054	4.37E+022
1	0.3021	8.7643E+054	1.32E+022	2.65E+054	4.37E+022
…	…	…	…	…	…
998	0.9309	2.8438E+054	4.07E+022	2.65E+054	4.37E+022
999	0.5652	4.6839E+054	2.47E+022	2.65E+054	4.37E+022
Total network hashes		**nlPow:**	**2.25E+025**	**PoW:**	**4.37E+025**

The next step determined the total number of hashes that would be calculated under existing PoW. The total estimated number of hashes that was calculated by the entire network under the existing PoW algorithm with a static target was

$$4.3738 * 10^{25}\ hashes$$

In contrast, the total estimated number of hashes that was calculated by the entire network under the nlPoW algorithm was

$$2.2486 * 10^{25}\ hashes$$

This nlPoW algorithm produces a saving of

$$2.215 * 10^{25}\ hashes\ OR\ 48.59\%$$

The above results depict a single block round. In order to see what the results will look like over time, the simulation was extended to produce results for 2016 block rounds to represent one target readjustment interval (Table 3).

Table 3: Simulated results for unidirectional dynamic target over one block adjustment interval

Rounds	2 016
Miners	1 000
Static target this interval (from block 592583 on 31 August 2019)	2.6474E+054
Average dynamic target for this interval	5.2978E+054
Expected static target time (seconds)	1 209 600
Expected adjusted interval time (seconds)	604 448
Actual simulated time (seconds)	616 784
Adjustment ratio	0.9800
Static target next interval	2.7014E+054
Expected static PoW hashes	8.8177E+028
Expected nlPoW hashes	4.4063E+028
Estimation of saving in the number of hashes (%)	50.03
Discouraged miners (%)	0.00

As a starting point, the simulated computational power of the network is derived from the selected static target and then randomly adjusted upward or downward during each block round by between 0% and 5%. These mimic the fluctuations in computational power as miners add resources to or withdraw resources from the network. At the end of the interval, an adjustment must be made to account for the net change of computational power over 2016 blocks. The mechanism employed by nlPoW to achieve this is explained in Section 4.3.1.

The estimated saving in the number of hashes executed over the interval is 50.03% which compares well with the expectation that the mean of the standardised random number for all miners during each block round should be 0.5. Since the dynamic target is larger than the static target, no miners are discouraged from mining under the assumption stated in Section 4.Since discouraging some proportion of miners is desirable, this shortcoming is addressed in Section 4.4 where the second method of using a uniformly distributed standardised number, to generate a random target for each miner, is discussed.

4.3.1 Adjusting the Static Target

Section 3.1 referred to the process whereby the static target (nBits parameter in the block header) is adjusted after every 2016 blocks to account for changes in the computational power of the network. In this respect, the static target is nothing other than the estimation of the hash rate of the network. Under nlPoW, the mean of the dynamic targets over 2016 blocks represents a factor whereby the block creation time should be faster than 10 minutes. This is the same factor whereby the block creation time should decrease under nlPoW because proportionally fewer hashes are required. This estimated nonlinear completion time over 2016 blocks can be compared to the actual completion time to compute a factor for the estimated increase or decrease of the network's computational power over the last 2016 blocks. By multiplying the static target by this factor, the new static target to be used as the basis for assigning dynamic targets for the next 2016 blocks, can be established.

The block creation time over 2016 blocks ("Actual simulated time (seconds)" in Table 3) was slower than the theoretical time ("Expected adjusted interval time (seconds)" in Table 3), the network's hash rate is therefore deemed to have decreased, and the static target was adjusted upward (difficulty reduced). This adjustment is proportional to the ratio between the actual simulated time and expected adjustment interval time ("Adjustment ratio" in Table 3) to yield a new static target for the next interval ("Static target next interval" in Table 3).

4.4 nlPoW: Bidirectional random target assignment

Bidirectional random target assignment seeks to discourage some of the miners from participating in some of the block rounds by decreasing the dynamic target (increasing the difficulty) based on the random number assigned to the miner. This is achieved by multiplying the static target by the random number if it is greater than 0.5. The opposite can be achieved by dividing the static target by the random number if it is less than or equal to 0.5, resulting in a larger dynamic target (decreased difficulty). The results of the simulation are shown in Table 4.

Table 4: Simulated results for bidirectional dynamic target over one block adjustment interval

Rounds	2 016
Miners	1 000
Static target this interval (from block 592583 on 31 August 2019)	2.6474E+054
Average dynamic target for this interval	1.0590E+055
Expected static target time (seconds)	1 209 600
Expected adjusted interval time (seconds)	302 390
Actual simulated time (seconds)	299 396
Adjustment ratio	1.0100
Static target next interval	2.6212E+054
Expected static PoW hashes	8.8177E+028
Expected nlPoW hashes	1.1014E+028
Estimation of saving in the number of hashes (%)	87.51
Discouraged miners (%)	50.04

The results of bidirectional random target assignment were done on the same basis as the simulation in Section 4.3 with regard to fluctuating network computational power, number of miners and number of block rounds. The estimated saving in the number of hashes with bidirectional dynamic target assignment is 87.51%.

5. DISCUSSION

nlPoW requires that a random dynamic target is assigned to each miner during each block round. The two methods proposed in this paper relies on the generation of a uniformly distributed random number that satisfies the requirements named in Section 4.1. There are two ways to apply the random number to the static target to transform it into a dynamic target, namely unidirectional target assignment (Section 4.3) and bidirectional target assignment (Section 4.4).

Simulation shows that both methods will reduce the number of hashes required by the network and therefore, its energy consumption. Since unidirectional target assignment does not discourage mining by a proportion of the network, it is less effective than bidirectional target assignment. nlPoW provides a technique for adjusting the static target in response to changes in the computational power of the network by comparing the actual mean of the block solution times with the estimated mean solution times of the algorithm over a block target interval.

Section 3.4 referred to the myriad of other types of consensus algorithms that are available for blockchain systems. For the purposes of this study, where the preservation of decentralised consensus is of concern, algorithms that rely directly on some form of centralised control over the network of participating nodes do not apply. These include all the voting-based algorithms and some proof-based algorithms like proof-of-entanglement (Bennet & Daryanoosh, 2019), proof-of-elapsed time (Nguyen & Kim, 2018), proof-of-luck (Nguyen & Kim, 2018) and proof-of-responsibility (Coinspace.com, 2019).

Of the remaining types of proof-based algorithms, some exchange computing power (and therefore energy) with another resource such as memory, storage or human input (Blocki & Zhou, 2016; Dziembowski *et al.*, 2015; Tromp, 2014). The authors argue that it is foreseeable that the same type of resource arms race, described in Section 3.3.1 would occur in a decentralised environment as it did with PoW in Bitcoin if they were implemented on a large scale. This may simply trade one type of problem for another.

The most promising competitors for PoW seems to be the stake-based algorithms (proof-of-stake, proof-of-importance and delegated proof-of-stake) (Bentov *et al.*, 2016; Hasib, 2018; NEM, 2018; Zheng *et al.*, 2017). These algorithms seek to limit the number of participating miners during each block round, which is in line with the aim of nlPoW, by ownership of blockchain tokens. Although the rules between these three algorithms vary slightly, they operate by allowing owners with larger proportions of the blockchain token, to mine a greater proportion of the transaction blocks. This may lead to hoarding strategies by miners and my in the long term reduce the randomness of the mining process and therefore its decentralised nature.

nlPoW does not compromise on the decentralised nature of the consensus process. It still relies on computational power but seeks to discourage a computational arms race, by introducing significant uncertainty for any party that aims to invest in additional computing power simply for the sake of blockchain mining. nlPoW does not require the use of any other resource as named above and does not encourage the hoarding of blockchain tokens as a strategy to increase mining success.

6. CONCLUSION

The approach described in this paper establishes a new direction of research, whereby the energy wastage of PoW type consensus algorithms in blockchain systems can be addressed. It proposes the critical requirements for nlPoW that randomly distributes the targets (difficulty) of mining blocks on the Bitcoin blockchain between the population of miners. This approach aims to lower the number of hash calculations required by the network to create new blocks on the blockchain. Since the amount of energy required for Bitcoin mining is proportional to the number of hash calculations (Narayanan *et al.*, 2016), the saving in energy usage is also directly proportional to the saving in hash calculations.

7. FUTURE RESEARCH

Although early results from simulations of Bitcoin mining under nlPoW shows promising results, further research is needed to establish the optimal distribution of dynamic targets between the population of miners. The results, as presented in this paper, distributes dynamic targets uniformly between miners, which serves the purpose of casting light on the viability of the concept. The researchers are currently investigating other types of distributions to investigate if it may yield more optimal results.

In this study, it was assumed that all miners possess equal computational power. In practice it is known that it has become infeasible for individual miners to mine Bitcoin with success (Eyal & Sirer, 2014), leading to alternative strategies like pooled mining (Tschorsch & Scheuermann, 2016). As a better understanding of nlPoW emerges, it will be necessary to investigate what effect pooled mining will have on the results.

A simplistic approach was followed with regard to the establishment of the participation threshold for each miner. In practice, the decision-making process of every miner may be different and much more complex than assumed here. This needs specific attention in future studies.

8. REFERENCES

Accord. (2017). Accord.NET Framework. Retrieved November 12, 2019, from http://accord-framework.net/docs/html/T_Accord_Statistics_Testing_KolmogorovSmirnovTest.htm

Antonopoulos, A. M. (2014). Mastering Bitcoin. O'Reilly Media. Retrieved from https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch05.html

Bennet, A., & Daryanoosh, S. (2019). Energy-Efficient Mining on a Quantum-Enabled Blockchain Using Light. Ledger, 4. https://doi.org/10.5195/ledger.2019.143

Bentov, I., Gabizon, A., & Mizrahi, A. (2016). Cryptocurrencies without Proof of Work. In 2016 Financial Cryptography and Data Security Conference.

BitcoinProject. (2020). Bitcoin Developer Reference. Retrieved January 12, 2020, from https://bitcoin.org/en/developer-reference

BitShares Blockchain Foundation. (n.d.). Delegated Proof-of-Stake Consensus. Retrieved August 11, 2019, from https://bitshares.org/technology/delegated-proof-of-stake-consensus/

Blocki, J., & Zhou, H.-S. (2016). Designing Proof of Human-Work Puzzles for Cryptocurrency and Beyond. In Theory of Cryptography (pp. 517–546). Springer. Retrieved from https://eprint.iacr.org/2016/145.pdf

Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., & Felten, E. W. (2015). SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In 2015 IEEE Symposium on Security and Privacy. Retrieved from https://ieeexplore.ieee.org/abstract/document/7163021

BTC.com. (2019). Pool Distribution. Retrieved October 11, 2019, from https://btc.com/stats/pool

Central Intelligence Agency. (2020). Czech Republic. Retrieved July 22, 2019, from https://www.cia.gov/library/publications/resources/the-world-factbook/

Coinspace.com. (2019). Electroneum's Revolutionary Proof of Responsibility Blockchain is Now Live. Retrieved from https://coinspace.com/news/altcoin-news/electroneums-revolutionary-proof-responsibility-blockchain-now-live

Digiconomist. (2020). Bitcoin Energy Consumption Index. Retrieved April 7, 2019, from https://digiconomist.net/bitcoin-energy-consumption

Dimitri, N. (2017). Bitcoin Mining as a Contest. Ledger, 96.

Dziembowski, S., Faust, S., Kolmogorov, V., & Pietrzak, K. (2015). Proofs of space. In CRYPTO: International Cryptology Conference. https://doi.org/10.1007/978-3-662-48000-7_29

Eyal, I., Gencer, A. E., Sirer, E. G., & Van Renesse, R. (2016). Bitcoin-NG: A Scalable Blockchain Protocol. In 13th USENIX Symposium on Networked Systems Design and Implementation. Santa Clara.

Eyal, I., & Sirer, E. G. (2014). Majority is not Enough: Bitcoin Mining is Vulnerable. In 18th International Conference on Financial Cryptography and Data Security.

Fischer, M. J. (1983). The Consensus Problem in Unreliable Distributed Systems. In International Conference on Foundations of Computation Theory.

Hasib, A. (2018). 101 Blockchains. Retrieved from https://101blockchains.com/consensus-algorithms-blockchain/#6

Jenks, T. (2018). Pros and Cons of Different Blockchain Consensus Protocols. Retrieved from https://www.verypossible.com/blog/pros-and-cons-of-different-blockchain-consensus-protocols

Ma, J., Gans, J. S., & Tourky, R. (2018). Market Structure in Bitcoin Mining (No. 24242). Cambridge, MA.

Mulár, B. O. (Masaryk U. (2018). Blockchain technology in the enterprise environment.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System.

Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2016). Bitcoin and Cryptocurrency Technologies. Princeton: Princeton University Press.

NEM. (2018). Proof-of-importance. Retrieved from https://nemplatform.com/wp-content/uploads/2020/05/NEM_techRef.pdf

Nguyen, G.-T., & Kim, K. (2018). A Survey about Consensus Algorithms Used in Blockchain. Journal of Information Processing Systems, 14(1).

P4Titan. (2014). Slimcoin: A Peer-to-Peer Crypto-Currency with Proof-of-Burn. Retrieved from https://www.doc.ic.ac.uk/~ids/realdotdot/crypto_papers_etc_worth_reading/proof_of_burn/slimcoin_whitepaper.pdf

Stark, P. B., & Ottoboni, K. (2018). Random Sampling: Practice Makes Imperfect. ArXiv.Org. Retrieved from https://arxiv.org/pdf/1810.10985.pdf

Tromp, J. (2014). Cuckoo Cycle: a memory bound graph-theoretic proof-of-work. Retrieved from https://eprint.iacr.org/2014/059.pdf

Tschorsch, F., & Scheuermann, B. (2016). Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. IEEE Communications Surveys & Tutorials, 2084–2123.

Zheng, Z., Xi, S., Dai, H., Chen, X., & Wang, H. (2017). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In 2017 IEEE 6th International Congress on Big Data.