Unstable Throughput: When the Difficulty Algorithm Breaks

Sam M. Werner
Imperial College London
smw16@imperial.ac.uk

Dragos I. Ilie
Imperial College London
dii14@imperial.ac.uk

Iain Stewart
Imperial College London
ids@imperial.ac.uk

William J. Knottenbelt
Imperial College London
wjk@imperial.ac.uk

ABSTRACT
Difficulty algorithms are a fundamental component of Proof-of-Work blockchains, aimed at maintaining stable block production times by dynamically adjusting the network difficulty in response to the miners’ constantly changing computational power. Targeting stable block times is critical, as this ensures consistent transaction throughput. However, without careful design this could create vulnerabilities that incentivize miners to engage in coin-hopping strategies which yield an unreliable system due to unstable processing of transactions.

We provide an empirical analysis of how Bitcoin Cash exhibits cyclicality in block solve times as a consequence of a positive feedback loop in its difficulty algorithm design. Additionally, we examine the extent to which miners’ behavior contributes towards this phenomenon over time. In response, we mathematically derive a difficulty algorithm based on a negative exponential filter that prohibits the formation of positive feedback loops and exhibits additional desirable properties, such as history agnosticism. We compare the described algorithm to that of Bitcoin Cash in a simulated mining environment and verify that the former would eliminate the severe oscillations in block solve times. Lastly, we outline how this model can more generally replace difficulty algorithms in other Proof-of-Work blockchains.

1 INTRODUCTION
Blockchains offer a decentralized mechanism for recording data in a trustless and immutable manner. Participants encode the exchange of value or information in transactions and broadcast them through a peer-to-peer network. Miners, the backbone of the system, aggregate transactions in data structures called blocks and append them to the blockchain. To reach consensus over the ordering of blocks, miners participate in a leader election process by solving a computationally intensive puzzle named Proof-of-Work (PoW). The first miner to find a valid solution appends his block and receives a reward for the invested computational effort. To ensure stable transaction throughput, the difficulty of the PoW problem is adjusted in response to changes in the miners’ computational power by a difficulty algorithm (DA). However, without careful design the DA can expose vulnerabilities, which when exploited by miners, lead to inappropriate difficulty levels and thus patterns of instability in the transaction throughput. In general, this issue arises in blockchains that lack a consistent amount of computational power due to some miners directing their resources towards other blockchains especially as profitability varies. For instance, such patterns have been observed even in Bitcoin Cash [4, 17] (BCH), the cryptocurrency with the 4th highest market capitalization1. Its developers have announced that fixing the high variations in block solve times is one of their main development goals for 2020 [5].

In this paper, we formally model a DA designed to stabilize transaction throughput even in chains without consistent computational power. To this end, we propose adjusting the difficulty after every block using exponential smoothing, a popular approach for time series data. To justify the specifics of the proposed algorithm we provide a case study2 on BCH’s current DA and investigate inherent vulnerabilities. We discover that even economically rational (i.e. profit-seeking) miner behavior leads to severe instabilities in transaction throughput due to a positive feedback loop in block solve times resulting from the design of the DA. We present desirable properties of the proposed DA and show how they remove the cyclical positive feedback mechanism. Furthermore, we demonstrate through simulations how the proposed DA performs under different scenarios and compare it to BCH’s.3 We find that the proposed DA would be an improvement over BCH’s current DA and suggest that it could be applicable to any PoW blockchain when configured appropriately.

1Data obtained from: https://coinmarketcap.com. Accessed: 26-03-2020.
2The code and data we based this analysis on can be found at: https://github.com/samwerner/DA-Analysis.
3The code base for the simulations can be found at: https://github.com/samwerner/DA-Analysis.
Contributions

This paper makes the following contributions:

- We conduct an empirical analysis of BCH’s current DA and examine how the DA’s design encourages coin-hopping behavior, which then leads to the formation of a positive feedback loop in block solve times.
- We quantify the impact this positive feedback loop has on the transaction throughput by measuring the distribution of blocks in one-hour periods.
- We examine the extent to which miners adopt coin-hopping strategies in response to changes in BCH’s profitability.
- We derive a DA which discourages coin-hopping strategies and present its additional properties that limit high variations in block solve times.
- Through simulations, we verify our claims and study the impact of various configurable parameters, arguing that the proposed DA can be customized to meet the requirements of other blockchains.

2 BACKGROUND

In this section, we introduce readers to the core workings of Bitcoin (BTC) [15] and Bitcoin Cash with particular focus on forks, timestamps, and difficulty algorithms. As BCH is a fork of BTC, the theoretical foundation and even practical implementation of both is mostly similar. Note that when we refer to BCH, we are referring to the Bitcoin ABC [5] full node implementation. Unless otherwise stated, the reader can assume any mention of Bitcoin applies to both BTC and BCH.

2.1 Bitcoin and Mining

Each node in the Bitcoin network maintains a copy of the blockchain, an immutable, publicly-shared distributed ledger. A set of transactions can be added to the blockchain by including them in a data structure referred to as a block. Each block contains a block header which summarizes the contents of this block and references the hash of the previous block’s header. Miners create blocks by solving the Proof-of-Work (PoW) puzzle: trying different values for the nonce field of the block header, such that its SHA-256 [16] hash lies below a specified target value. In Bitcoin, the target is a 256-bit number encoded in the nBits field of the block header and it has a maximum value of \(0x1d00ffff\) \((\approx 2^{224})\). The notion of difficulty expresses the ratio of the maximum target to the current target, \(D \approx \frac{2^{256}}{\text{target}}\). As SHA-256 computations produce random yet deterministic outputs, each attempt can be modeled as a Bernoulli trial with success probability \(\frac{\text{target}}{2^{256}}\). Therefore, the expected number of hashes that need to be computed to mine a block at a specific difficulty \(D\) is approx. \(D \cdot 2^{256}\). As attempts are independent of one another, the time it takes to mine a block, namely the block solve time, follows an exponential distribution with a rate parameter \(\lambda = \frac{H}{D \cdot 2^{256}}\), where \(H\) is the total hash rate of the network. The expected solve time is then, \(\frac{1}{\lambda} = \frac{D \cdot 2^{256}}{H}\). In Bitcoin, the desired block solve time is 10 minutes, which is maintained by adjusting the target depending on the current hash rate estimate; the lower the target, the more difficult it becomes to find a PoW solution.

The miner of a block is compensated for the invested computational efforts with a reward of newly minted Bitcoins and any fees from transactions included in the block. The block reward started at 50 Bitcoins and halves every 210,000 blocks, or approx. every 4 years (e.g. currently it is set at 6.25 Bitcoins).

2.2 Forks

In Bitcoin, the main chain refers to the chain with the most accumulated performed work, i.e. the sum of difficulties of the mined blocks. A fork happens when at least two blocks reference the same past block. This may occur due to slow network propagation of blocks when two miners find a solution to the PoW at nearly the same time. As miners build upon the block that reaches them first, the temporary fork is likely resolved after one branch exceeds the other in terms of work and becomes the new main chain.

Upgrades to the protocol rules can be deployed via hard forks, whereby a permanent split in the blockchain occurs, creating two separate coins. Some of the most prominent Bitcoin hard forks are BCH and Bitcoin SV (BSV) [18]. Note that both BCH and BSV kept the same PoW algorithm as Bitcoin; hence, miners can transition at ease between them.

2.3 Block Timestamps

Each block header contains a UNIX timestamp indicating when the block was mined. As clock synchronization is a well-known problem in distributed networks, block timestamps may not necessarily be in monotonically increasing order. To ensure the blockchain time advances, the Bitcoin protocol requires blocks to have a timestamp greater than the median timestamp of the previous 11 blocks, also known as the Median Time Past (MTP). Additionally, nodes also enact a convention by which they accept new blocks only if their timestamp does not exceed the network adjusted time\(^4\) by more than 2 hours. For a more in-depth analysis of timestamps, potential attacks and improvements we direct the reader to [6, 19, 21].

\(^4\)The network adjusted time is the median timestamp of all the current times received by a node from its peers.
2.4 Difficulty Algorithms

A difficulty algorithm (DA) is a fundamental component of PoW blockchains as it regulates the transaction throughput by adjusting the hardness of generating a PoW solution. A DA is responsible for ensuring stable block times in periods of hash rate oscillations caused by miners joining and leaving the network. Failing to ensure an appropriate difficulty could result in either short time periods with many blocks being found, or long time periods with very few blocks, which results in a highly variable response time for blockchain transactors.

Computing the difficulty of a block must be deterministic and based on data from previous blocks s.t. individual nodes can perform the computation independently and agree on the same results. An omniscient DA with knowledge of the real world hash rate would be able to compute the difficulty of the next block by simply multiplying the current hash rate with the ideal inter-block time (e.g. 10 minutes). However, in practice DAs estimate the current hash rate based on the difficulties and solve times of previous blocks; hence, the estimates always lag behind the actual hash rate. The extent to which a DA is able to minimize this lag is regarded as the responsiveness of the algorithm. Blockchains with relatively stable hash rate, can afford to use a less responsive DA to reduce volatility in difficulty, allowing miners to predict expected rewards over near-term time scales. Note that the responsiveness of a DA also depends on the frequency of when the difficulty is adjusted. Some currencies, such as BCH adjust the difficulty after every block, whereas others do so after some fixed number of blocks. For instance, BTC adjusts its difficulty every 2016 blocks, or approx. 2 weeks.

2.5 Miner Incentives

Blockchains which use the same PoW puzzle can be seen as being PoW compatible from a miner’s perspective, i.e. miners can switch between them with no additional hardware overhead. A common way for comparing the profitability of PoW compatible blockchains is via the Difficulty Adjusted Reward Index (DARI) [7], which is computed as:

\[
R_i \cdot E
\]

(1)

where \(R_i\) is the (expected) reward for block \(i\), \(D_i\) the difficulty of block \(i\) and \(E\) the exchange rate for the coin in some base currency (e.g. USD, BTC). Recall that \(R_i\) consists of transaction fees and the block reward which is constant for long periods of time. Therefore, it can be seen that profitability is mostly impacted by changes in difficulty or the exchange rate. An economically rational miner aiming for short term profits, can thus engage in a strategy called coin-hopping [10, 12, 14], whereby the miner continuously redirects his computational power towards the most profitable cryptocurrency. On the other hand, long term profit-seeking miners allocate most of their computational power to the blockchain which they believe will have the highest valuation in the long run.

Depending on the distribution of hash rate across PoW compatible blockchains, hash rate fluctuations induced by coin-hopping behavior impact the blockchains to different extents. For instance, on average 97% of the total SHA~256 hash rate is concentrated in BTC, while the remaining 3% is distributed between BCH, BSV, and others.\(^5\) As a result, fluctuations in the distribution of the total hash rate impact BCH and BSV significantly more than BTC.

3 EMPIRICAL ANALYSIS OF BCH’S DA

In this section, we provide an empirical analysis on issues stemming from the current DA employed in BCH.

3.1 BCH’s Difficulty Algorithm

Until now, BCH has used two different difficulty adjustment mechanisms – previously a modified version of BTC’s DA and currently a more responsive one. To examine to what extent the current DA has been an improvement, we will also outline the workings of the initial DA.

3.1.1 Emergency Difficulty Algorithm. When the BTC–BCH fork occurred on 1st August 2017, BCH kept BTC’s PoW puzzle, while slightly adapting its DA. In BTC, the new difficulty \(D’\) is updated every 2016 blocks based on the previous difficulty \(D\), using the following formula:

\[
D’ = D \cdot \max\left(\min\left(\frac{2016 \cdot T}{T_A}, 4\right), \frac{1}{4}\right)
\]

(2)

where \(T\) is the ideal inter-block time and \(T_A\) is the time it actually took to mine the last 2016 blocks.

As miners could engage in coin-hopping strategies and the loyal hash rate was expected to be much lower than BTC’s, BCH developers foresaw that a scenario such as the one described in the last paragraph of Section 2.5 would arise. To ensure stable block throughput during large effluxes of hash rate, BCH resorted to the Emergency Difficulty Algorithm (EDA), whereby the difficulty would drop by 20% if the difference between 6 successive block timestamps exceeded 12 hours [1]. Therefore, BCH’s first DA was a combination of BTC’s DA and the EDA. However, it soon became apparent that this difficulty adjustment mechanism did not fulfill its objective. Miners would stop mining BCH in order to cause consecutive 20% drops in the difficulty, which only adjusted back upwards every 2016 blocks. Once the difficulty was sufficiently low, miners would switch back to mining BCH and produce many blocks at very low difficulty until the end of the 2016 blocks window. As a result of this miner behavior, from 1st August 2017 to 13th November 2017 a total of 9,947

\(^5\) Data collected from https://www.fork.lol. Accessed: 2020-04-10
more blocks were mined in BCH than in BTC (Figure 14 from Appendix A).

3.1.2 Amaury’s Difficulty Algorithm. The combination of the 2016 blocks window and the EDA was replaced on 13th November 2017 with a new DA proposed by BCH developer Amaury Sechet. To increase responsiveness to both effluxes and influxes of hash rate, the new DA performs difficulty adjustments on a per-block basis.

The difficulty \( D \) of a new block is derived from the estimated hash rate, \( \tilde{H} \), and the ideal inter-block time, \( T \) (i.e. 10 minutes). To this end, \( \tilde{H} \) is computed using a simple moving average with a sample size of approx. 144 blocks. To mitigate situations when the block timestamps are out-of-order, the bounds of the sliding window over which the average is computed are derived using the median timestamp of 3 blocks. Thus, the block at which the window starts, \( B_{start} \), is the block with the median timestamp out of blocks 144, 145, and 146 in the past. Similarly, the window ends at block, \( B_{end} \), with the median timestamp of the 3 most recent blocks. From these two blocks the DA computes \( W \), the amount of work that was performed between these two blocks, as the sum of difficulties of all blocks in the interval \([B_{start}, B_{end}]\). The estimated hash rate is: \( \tilde{H} = W / T_A \), where \( T_A \) is the actual time elapsed between \( B_{start} \) and \( B_{end} \), capped in the interval from half a day to 2 days to prevent difficulty changing too abruptly. For completeness we give the full equation for the new difficulty:

\[
D = \tilde{H} \cdot T = \frac{\sum_{i=\text{start}}^{\text{end}} \text{diff}(B_i)}{T_A} \cdot T \tag{3}
\]

3.2 Oscillations in Number of Blocks Mined per Hour

As intended, Amaury’s DA achieves a daily average solve time of 10 minutes. This gives the superficial impression of performing well in terms of stable throughput, however certain patterns in the distribution of blocks within a day emerge.

From Figure 1 it can be seen that the oscillations in number of blocks mined per hour are notably more severe in BCH than in BTC. Especially during the later months, it is evident that BCH exhibited more 1 hour periods with either many blocks mined or none. As the number of blocks mined in an hour, \( K \), should ideally follow a Poisson distribution with rate parameter \( \lambda = 6 \) blocks, we can compute the expected probability of mining exactly \( k \) blocks in one hour as:

\[
P(K = k) = \frac{\lambda^k}{k!} e^{-\lambda} \tag{4}
\]

We compare these ideal values with empirical results from BCH and BTC in Figure 2. For reference, it can be seen that in BTC the probabilities closely resemble those of a Poisson process. In contrast, BCH shows significant deviations from the ideal distribution during the period in which the EDA was active. After abandoning the EDA, BCH has indeed shifted towards the Poisson distribution, but a skew on the left and right tails remains. This is in line with the aforementioned observations of a more unstable transaction throughput in BCH, as shown in Figure 1.

![Figure 1: Number of blocks mined per hour in BTC and BCH since Amaury’s DA deployment.](image)

![Figure 2: The probabilities of mining exactly \( k \) blocks in a one-hour period in BTC and BCH (pre and post EDA).](image)

3.3 Positive Feedback Loop in Simple Moving Averages

The observed instability in transaction throughput can be explained by a positive feedback loop that stems from a combination of two factors: the use of a simple moving average and the miners’ economically rational behavior.

From formula (3) it is apparent that Amaury’s DA relies (in part) on the relationship of inverse proportionality between the time duration of the sliding window \( T_A \) and the estimated hash rate, \( \tilde{H} \). The same relationship exists between the hash rate fluctuations and the solve times of newly mined blocks; i.e. solve times decrease when there is an increase of hash rate and they increase when there is a decrease. As new solve times are added to \( T_A \), the result of these two relations is
that $\hat{H}$ is adjusted directly proportional to the actual hash rate change. However, the oversight of this DA is that using a simple moving average implies solve times falling off the window (subtracted from $T_A$) have an equal weight in the computation of $\hat{H}$. Short solve times 144 blocks in the past cause a relative increase in $T_A$ which yields a lower than expected $\hat{H}$. Similarly, long solve times falling off the window imply a relative decrease in $T_A$ and therefore produce a higher $\hat{H}$. This influence constitutes positive feedback that results in correlation between solve times 144 blocks apart. This is inherently problematic since mining is supposed to be a Poisson process where block arrivals are independent events.

The second factor that contributes towards the positive feedback loop is the miners’ behavior as they try to maximize profit by engaging in coin-hopping. For instance, assume BCH experiences an increase in profitability which incentivizes a group of coin-hopping miners $M_{CH}$ to switch their computational power towards BCH. This causes an increase in hash rate and consequently a series of blocks with short solve times. As the difficulty adjusts upwards, BCH’s profitability drops causing $M_{CH}$ to direct their hash rate towards more profitable chains. While the hash rate drops to its original value, the difficulty is now too high for the network, so a series of blocks with long solve times is produced.

This phenomenon of short solve times followed by long solve times is not intrinsically problematic, however, due to the positive feedback found in Amaury’s DA this pattern repeats continuously resulting in a positive feedback loop. This phenomenon has also been examined by [20, 22, 23].

Figure 3: The autocorrelation in number of blocks mined per hour in BCH and BTC since Amaury’s DA was deployed.

To investigate the extent of this cyclical phenomenon in BCH, Figure 3 compares the autocorrelation in the number of blocks mined per hour in BTC and BCH. As mining is supposed to be a memoryless process there should not exist any significant autocorrelation which is what we see in BTC’s graph. On the other hand, BCH has a significant positive autocorrelation immediately after 24 lags. Furthermore, in BCH weak positive correlation also persists after multiples of 24 lags. As a lag in this case represents a one hour interval, 24 lags represent the same expected duration as the sliding window of 144 blocks. Therefore, these empirical findings are in line with the positive feedback loop described.

3.4 Coin-hopping Incentives

Given the aforementioned issues related to difficulty adjustments, we attempt to assess the extent to which miners are incentivized to engage in coin-hopping and therefore contribute towards the formation of the positive feedback loop.

3.4.1 Deserts and Spikes. For the purposes of this analysis we define a desert, as a one hour interval during which at most 1 block is mined and a spike as a one hour interval during which 12 or more blocks are mined. Note, we have chosen these thresholds so their probabilities are small and relatively comparable. Building on equation (4), we compute the probabilities of mining at most $k$ blocks per hour:

$$ P(K \leq k) = \sum_{i=0}^{k} P(K = i) $$

Hence, we expect deserts and spikes to occur with a probability of $P(K \leq 1) = 1.74\%$ and $1 - P(K \leq 11) = 2.01\%$, respectively. We refer to a period which is neither a spike nor a desert as a normal period with $P(1 < K \leq 11) = 96.25\%$.

From Figure 4 it becomes apparent that not only are the expected likelihoods of deserts and spikes not achieved, but that the situation appears to be aggravating over time. For instance, over the last 6 months of the examined period, deserts and spikes occurred 13.5% and 12.7% of the time, respectively. By contrast, in BTC the respective percentages are 1.6% and 2.2%, which are significantly closer to the expected values.

Figure 4: The number of one hour intervals classified as spikes, deserts, and normal periods in BCH.

3.4.2 Mining Profitability Comparison. We examine differences in the mining profitability of BCH and BTC by comparing the ratio of their DARIs (i.e. BCH DARI over BTC DARI) in Figure 5. In the long term, mining either coin is
equally profitable as the average DARI ratio has a value of 1.0266. However, as the ratio frequently oscillates this incentivizes miners to adopt a coin-hopping strategy. Notably, in the latter months, the oscillations become significantly more frequent and consistently reaching deviations of 10% and even 15% either in favor of BCH or BTC. These fluctuations are reflected in an increased number of spikes and deserts during the same period as can be seen in Figure 4.

Interestingly, out of the five largest mining pools, only BTC.TOP mined a similar amount of blocks during spikes and normal periods, while the remaining four pools mined on average 4.66% less blocks during spikes. This indicates that BTC.TOP is the only pool that successfully engages in coin-hopping by mining with higher hash rate during periods of lower difficulty. The other pools lose part of their block share to the coin-hopping miners. For instance, miners qp4ajq and qqq9v3 obtained the third and fifth highest shares of blocks, while mining more than 70% of their blocks during spikes and, perhaps rather impressively, none during deserts.

The extent of such coin-hopping behavior can also be measured by analyzing the fluctuation in BCH’s hash rate. The logarithmic scale chart from Figure 6 shows the hash rates of BTC and BCH over the last 6 months, estimated using a moving average of 6 blocks. While BTC’s hash rate consistently oscillates from approx. 90 to 180 Exahashes per second, BCH’s hash rate fluctuates from approx. 2 to 18 Exahashes per second. This means that BCH experiences periods when the hash rate increases even 9 times relatively to the baseline hash rate, which is inline with the results of Table 1.

\[ \text{Figure 5: The minute average DARI ratio of BCH to BTC. Equal profitability is shown by the black line.} \]

3.4.3 **Miner Analysis.** In order to examine the extent to which miners benefit from mining BCH, we analyze the block distribution for a set of high hash rate miners between 11\textsuperscript{th} September 2019 and 11\textsuperscript{th} March 2020. We deem this period relevant as the number of spikes and deserts is considerably higher than before.

In Table 1, we give data for the five largest BCH mining pools (BTC.TOP, Antpool, BTC.com, ViaBTC, Huobi Pool) and three large miners without known identities, sorted by their respective shares of total blocks mined during the examined period.

\[ \text{Figure 6: Estimated hash rates of BTC and BCH using a 6 block average.} \]

| Miner         | % of Total Blocks Mined in: |
|---------------|-------------------------------|
|               | Normal | Spikes | Deserts | Total  |
| BTC.TOP       | 8.95   | 8.28   | 0.12    | 17.35  |
| Antpool       | 8.40   | 1.84   | 0.29    | 10.53  |
| qp4ajq        | 2.36   | 6.76   | 0.00    | 9.12   |
| BTC.com       | 6.80   | 1.18   | 0.31    | 8.29   |
| qqq9v3        | 2.21   | 5.36   | 0.00    | 7.57   |
| ViaBTC        | 5.91   | 1.18   | 0.29    | 7.38   |
| qzkuv6        | 2.77   | 2.39   | 0.06    | 5.16   |
| Huobi Pool    | 2.47   | 0.73   | 0.06    | 3.26   |

\[ \text{Table 1: Proportion of blocks mined by large miners during normal, spike and desert periods between block numbers 599798 and 625989.} \]

\[ \text{A mining pool allows multiple miners to combine their computational efforts and share the rewards. A pool can be seen as a single miner entity.} \]

4 **NEGATIVE EXPONENTIAL FILTER DIFFICULTY ALGORITHM**

In this section, we mathematically derive a DA based on a negative exponential, low-pass filter. Although the initial formula may seem rather complicated, we can reduce it to a surprisingly elegant form which reveals several desirable properties, such as the lack of positive feedback.

To simplify mathematical computations and explanations we abstract the implementation details of the PoW target and mining difficulty. We model the target as a real number between 0 and 1 and ignore the maximum target requirement which is merely an implementation optimization.\[^{6}\] We refer to the inverse of the target as difficulty and note that it

\[^{6}\] Full address of miners from Table 1 are given in Table 2 from Appendix A.

\[^{7}\] The maximum target was also the target of the genesis block and it was set to match the hash rate capabilities of the first miners.
represents the expected number of hashes that need to be computed to obtain a valid PoW solution.

### 4.1 Difficulty Algorithm Requirements

DAs need to be reactive to hash rate fluctuations, especially in the case of coins where there might be regular influxes causing, e.g., a doubling or tripling of the overall hash rate. To ensure the difficulty adapts swiftly, the adjustment can be performed on a per-block basis, which is indeed the design of the current BCH algorithm. The DA we propose will maintain this property s.t. sudden hash rate fluctuations can be accounted for. Most DAs employ some kind of sliding window, considering only the most recent blocks. Intuitively, this approach is justified as the difficulty of older blocks is mostly a measure of the less evolved technology available at that time. However, for the reasons discussed in Section 3.3, we are interested in avoiding the use of a sliding window while still implementing its intention.

### 4.2 Mathematical Derivation

To satisfy the aforementioned requirements, we apply a negative exponential filter over all the block difficulties, weighing them based on time and a decay factor. For example, Figure 7 shows the effect of such a filter applied to 10,000 blocks from BCH. Notice how the more recent block difficulties are barely affected by the filter while the blocks far in the past will bring little contribution to the overall result. To obtain the estimated hash rate, we compute the weighted mean of difficulties over the full time span of the blockchain.

![Figure 7: Difficulties of 10 000 blocks are filtered with a negative exponential to obtain the weighted difficulties.](image)

Throughout the remaining explanations, we make use of the following notation:

\[ \tilde{H}_i \leftarrow \text{estimated hash rate for block } i \]
\[ D_i \leftarrow \text{difficulty of block } i \]
\[ t_i \leftarrow \text{time of block } i \]
\[ s_i \leftarrow \text{solve time of block } i : s_i = t_i - t_{i-1} \]
\[ T \leftarrow \text{ideal block solve time (e.g., 10 minutes)} \]
\[ S \leftarrow \text{decay/smoothing factor (see Section 4.3.2)} \]

Index \( i \) refers to the index of a block and, by convention, index 0 refers to the block at which the new DA is deployed, while index \( n \) refers to the height of the next block to be appended. Thus, \( t_n \) is the current time and \( \tilde{H}_n \) is the current hash rate.

Then, the difficulty of the block currently being mined can be computed using the ideal inter block time, \( T \):

\[ D_n = \tilde{H}_n \cdot T \] (6)

#### 4.2.1 Estimating Current Hash Rate

We estimate the current hash rate, \( \tilde{H}_n \), by applying a popular time series averaging technique known as exponential smoothing (or exponentially weighted moving average). Ideally, we would perform this operation over the continuous function describing the hash rate of the network. However, the actual hash rate is not known at any given point; hence the best we can do is rely on sampling this function when information is available.

For instance, the difficulty \( D_i \) of block \( i \) represents the expected number of hashes that were performed in the time interval \( (t_{i-1}, t_i] \), so we could consider that \( D_i \) hashes were computed at time \( t_i \). Essentially, this approximation allows us to sample the hash rate at specific points in time, i.e., when blocks are mined.

Using the aforementioned approximation we can apply exponential smoothing to the series of block difficulties. The weights of data points decrease exponentially based on the formula: \( f(i) = e^{(t_i-t_n)/S} \), where \( S \) is the smoothing factor. Given the honest timestamp assumption, the exponent is always negative and the following ordering applies: \( f(i) > f(i-1) \). Therefore, block difficulties far in the past have a lower weight compared to more recent ones. Equation (7) computes the current hash rate by applying exponential smoothing to the series of block difficulties.

\[ \tilde{H}_n = \sum_{i=0}^{n-1} D_i e^{t_i-t_n} = \frac{1}{S} \sum_{i=0}^{n-1} D_i e^{t_i-t_n} \] (7)

We have chosen the bounds for computing the weighted length of the time interval s.t. the reduction to \( S \), the decay factor, becomes apparent. By convention, we place the
current time at the origin of the timeline and extend the filter all the way to \(-\infty\). Therefore, the reduction is given in equation (8).

\[
\int_{-\infty}^{0} e^{\frac{t}{S}} dt = S e^{\frac{0}{S}} = S - S e^{-\infty} = S \tag{8}
\]

4.2.2 Difficulty Computation. Using equations (6) and (7) we obtain the formula for the difficulty of the next block:

\[
D_n = T \cdot \hat{H}_n = T \sum_{i=0}^{n-1} D_i e^{-\frac{t_i - t_n}{S}} \tag{9}
\]

Although it seems rather inefficient to compute a sum over all blocks, we can apply several transformations which will simplify the above form. Adding and subtracting \(t_{n-1}\) at the numerator in the exponent of \(e\) allows grouping and distributing a common term:

\[
D_n = \frac{T}{S} \sum_{i=0}^{n-1} D_i e^{-\frac{t_i - t_{n-1} + t_{n-1} - t_n}{S}} \tag{10}
\]

\[
= \frac{T}{S} \sum_{i=0}^{n-1} D_i e^{-\frac{t_i - t_n}{S}} \tag{11}
\]

Notice that for the last term of the summation the exponent is simply: \(\frac{t_n - t_{n-1}}{S} = 0\). Thus, after extracting the last term out of the summation (12) and distributing the \(\frac{t}{S}\) factor (13), we notice the first term can be replaced from equation (9) obtaining a recurrent relation (15):

\[
D_n = \frac{T}{S} \left( \sum_{i=0}^{n-2} D_i e^{-\frac{t_i - t_{n-1}}{S}} + D_{n-1} \right) e^{-\frac{st_n}{S}} \tag{12}
\]

\[
= \frac{T}{S} \sum_{i=0}^{n-2} D_i e^{-\frac{t_i - t_n}{S}} + \frac{T}{S} D_{n-1} \right) e^{-\frac{st_n}{S}} \tag{13}
\]

\[
= \left( D_{n-1} + \frac{T}{S} D_{n-1} \right) e^{-\frac{st_n}{S}} \tag{14}
\]

\[
= D_{n-1} \left( 1 + \frac{T}{S} \right) e^{-\frac{st_n}{S}} \tag{15}
\]

If we unwind the recurrent relation up to \(D_0\) we obtain equation (16). This can be simplified further by using properties of exponentials and replacing the solve time with its extended form:

\[
D_n = D_0 \left( 1 + \frac{T}{S} \right)^n \sum_{i=1}^{n} -\frac{st_i}{S} \tag{16}
\]

\[
= D_0 \left( 1 + \frac{T}{S} \right)^n \sum_{i=1}^{n} -\frac{t_i}{S} e^{-\frac{st_i}{S}} \tag{17}
\]

\[
= D_0 \left( 1 + \frac{T}{S} \right)^n \sum_{i=1}^{n} -\frac{t_i}{S} e^{-\frac{t_i}{S}} \tag{18}
\]

\[
= D_0 \left( 1 + \frac{T}{S} \right)^n \sum_{i=1}^{n} -\frac{t_i}{S} e^{-\frac{t_i-t_0-t_1}{S}} \tag{19}
\]

\[
= D_0 \left( 1 + \frac{T}{S} \right)^n \sum_{i=1}^{n} -\frac{t_i-t_0}{S} \tag{20}
\]

Finally, we note that when \(T \ll S\) we can approximate \(1 + T/S \approx e^{T/S}\). In fact, this is not only an approximation but a correction needed to mitigate the bias introduced by considering a discrete series of difficulties instead of the continuous function of hash rate. We detail the specifics of this correction and give a formal argument motivating its need in Section 4.5. Applying the correction in equations (15) and (20) we obtain the following equivalent but concise forms of the proposed DA:

\[
D_n = D_{n-1} e^{-\frac{T - st_n}{S}} \tag{21}
\]

\[
D_n = D_0 e^{-\frac{nT + t_0 - t_n}{S}} \tag{22}
\]

Even though the two formulas are equivalent, we suggest the use of the non-recurrent version (22) to avoid compounding possible floating point errors in computing exponentials.

4.3 Interpretation

Having derived these formulas, we provide intuition on specific aspects such as the decay factor and the nature of difficulty adjustments. The recurrent form (21) depends solely on the last block’s difficulty and the current solve time, while formula (22) relies only on the number of blocks, \(n\), and the current solve time, \(t_n\).

Equation (21) is somewhat intuitive as the new difficulty is computed from the previous difficulty based on the current solve time. On average, if a block is found faster than the ideal inter block time \(T\), this signifies that the hash rate increased, hence the difficulty should be adjusted upwards. On the other hand, if on average \(T < st_n\), the difficulty should be adjusted downwards to decrease solve times.

4.3.1 Clockwork Toy Time. Formula (22) can be interpreted by introducing the concept of “clockwork toy time” (CTT). We define the clockwork toy time as \(CTT = nT + t_0\), where \(n\) is the height of the block currently being mined and \(t_0\) is the timestamp of the block at which the DA was deployed. Therefore the proposed DA simply computes the difficulty of a new block by comparing the CTT with the current time. On average, if the CTT is ahead of the actual time it signifies that miners found more blocks than expected, so difficulty should be increased, which is what the formula accomplishes.

If we assume a scenario in which the hash rate increases and then remains constant it might not be immediately apparent why the difficulty stabilizes. This is the case as the current time never catches up with the CTT because \(t_n\) is a summation of solve times tending on average towards \(T\), while the CTT is a summation of \(T\), so there will always be a lag between them. The only reason the difficulty would drop is a decrease in hash rate, which on average would lead to longer solve times, therefore allowing the blockchain time to catch up with (and possibly exceed) the CTT.
4.3.2 **Smoothing Factor.** The smoothing factor $S$ is introduced to impose a maximum rate of change for the difficulty. More specifically, the difficulty can change by at most a factor of $e$ in $S/T$ blocks. This setting mitigates the inherently random nature of the Poisson distributed blocks s.t. variations in individual solve times are not reflected in over- or under-adjustments. Consequently, depending on the requirements of the application, $S$ should be chosen carefully. Blockchains that are prone to experience large hash rate fluctuations on a regular basis (e.g. BCH), should aim for smaller values of $S$ to obtain a more reactive DA. On the other hand, blockchains with a relatively stable hash rate (e.g. BTC) can choose larger values for $S$, which will reduce the difficulty’s volatility. Therefore, we could make an analogy between $S$ and the size of the sliding window used to compute the simple moving average in DAs. However, there is no direct relationship between the smoothing factor of an exponential moving average and the sample size of a simple moving average, as their operation is considerably different. Empirical studies such as [24] suggest that in order to obtain similarly stable difficulties the smoothing factor of an exponentially moving average should be chosen to represent $(N + 1)/2$ blocks where $N$ is the length of the sliding window used in simple moving averages. Applying this heuristic to BCH which has a sliding window of 144 blocks, $S$ should be set at approx. 12 hours.

4.4 **Properties**

4.4.1 **History Agnosticism.** We define this property to signify the fact that the distribution of blocks in a given time period does not influence the difficulty of a block with a certain height and timestamp. To illustrate this property we rewrite formula (22) as follows:

$$D_n = D_0 e^{\frac{nt_f - t_a}{S}}$$

(23)

Notice that for each block the difficulty is adjusted upwards by a constant factor of $e^{t_f/S}$, regardless of the block’s timestamp. At the same time, the passing of time $t_n$ adjusts the difficulty downwards. Therefore, the difficulty of a block being mined at time $t_a$ depends solely on $t_a$ and the number of blocks on the respective chain. Consequently, the proposed DA computes the same value regardless of whether all blocks were mined a very long time in the past, in the last hour, or equally distributed throughout the history of the chain.

4.4.2 **Lack of Autocorrelation.** Not only does this algorithm avoid the use of a sliding window, but the lack of autocorrelation is an emergent property entailed by history agnosticism. Sudden influxes or effluxes of hash rate will still produce temporary spikes or deserts, yet arguably shorter. However, these will not create a positive feedback loop as the distribution of blocks in time has no influence on the future.

4.5 **Correction**

When estimating the current hash rate, $H_n$, we would ideally apply the exponential filter over the continuous function describing the hash rate of the network. However, as the true network hash rate is unknown, we rely on sampling this function at specific points in time, i.e. when blocks are mined. For block $i$, we consider $D_i$ hashes were computed at time $t_i$. In reality these hashes were performed throughout the interval $(t_{i-1}, t_i]$, so a certain bias is introduced by clustering all the hash rate towards the end of the intervals.

To mitigate this bias we will use equation (15) and replace $1 + T/S$ with a constant $k$. We will deduce the correct value of $k$ by expecting a correct behavior from the DA under a theoretical scenario. Specifically, we assume the hash rate remains constant: $H_{const}$. We expect the difficulty to adjust towards the ideal value $H_{const} \cdot T$, oscillating within some margin around it due to the nature of the Poisson process that models block arrival. We take a sample of blocks from $m$ to $n$ that were mined by this constant hash rate. The average rate of change in difficulty should be $\overline{R} = 1$, indicating that on average the difficulty does not change. Thus, we take the geometric mean of ratios of consecutive difficulties from block $m$ to $n$:

$$\overline{R} = \frac{n-m}{\prod_{i=m+1}^{n} \frac{D_i}{D_{i-1}}} = \frac{n-m}{\sqrt[\alpha]{\frac{D_{m+1}}{D_m} \frac{D_{m+2}}{D_{m+1}} \cdots \frac{D_{n}}{D_{n-1}}} / D_n}$$

$$= \frac{n-m}{\sqrt[\alpha]{D_n / D_m}} = \frac{n-m}{\sqrt[\alpha]{D_0 k^e^{(t_0 - t_m)/S}}}$$

$$= \frac{n-m}{\sqrt[\alpha]{D_0 k^e^{(t_0 - t_m)/S}}} = k e^{(t_n - t_m) / S}$$

Assuming the DA is working correctly, the average solve time of blocks from $m$ to $n$ is $T$ so $\frac{t_n - t_m}{(n - m)} = T$. Performing this replacement in the previous equation we obtain $k$, as follows:

$$ke^{\frac{T}{S}} = 1 \implies k = e^{\frac{T}{S}}$$

Therefore, the correction we have performed in the proposed DA is indeed justified.

4.6 **Real Time Targeting Consideration**

In an ideal world, miners do not lie about the timestamp of the block they are currently mining on, allowing for real time targeting (RTT) in a DA. In practice, blockchains such as BCH and BTC generally have honest timestamps because miners enact the convention that blocks should not be more than 2 hours in the future from the network adjusted time, therefore limiting the extent of manipulation possible. Note that this rule is applied even though it cannot be part of the consensus protocol as blocks that are invalid at a certain time may become valid if enough time passes.
Assuming such a rule would continue to be enforced, the proposed DA is susceptible to a 2 hour timestamp manipulation. Not only does this amount to a very small gain, but due to history agnosticism this only lowers the difficulty of the dishonest block. In contrast, in BCH the same manipulation would lower the difficulty for successive blocks as well, thereby creating incentives for miners to accept the dishonest block when there are multiple temporary forks in competition. For these reasons, when relying on this “2 hour” rule, we can safely apply RTT as dishonesty is discouraged. It might be the case that the proposed DA creates incentives for miners to stop enforcing the informal “2 hour” rule, and lie endlessly. We argue this scenario is not probable for the following reasons. Building on a dishonest block \(B_i\) (i.e. more than 2 hours in the future) implies mining towards a difficulty that is \(e^{T/5}\) times higher than that of the previous block \(B_{i-1}\). For this reason, a miner would only accept this block if he is willing to perpetuate the lie even further, thereby mitigating the increase in difficulty. As miners perpetuate each others’ lies, the blockchain’s time advances unnaturally while the difficulty is considerably lower than expected. As the main chain is determined by the most accumulated work, a sequence of blocks with cumulative work \(W'\) (sum of difficulties) can be replaced by a series of (potentially fewer) blocks with cumulative work \(W > W'\). Assuming the majority of hash rate is honest, miners are not incentivized to operate on a chain with dishonest timestamps as it produces less cumulative work than the honest chain and therefore risks being replaced. Only an attack supported by a majority of the hash rate would be successful, but this is no different than timestamp manipulation attacks that are currently possible in BCH or even BTC [6, 19, 21].

As an alternative solution, to completely remove possible issues arising from using RTT, we can replace any occurrence of a time \(t_i\) with the median of the 11 blocks preceding block \(i\). As the MTP of 11 consecutive blocks is guaranteed to be ordered by consensus rules, this satisfies our requirement for monotonic time while not changing any of the assumptions existing DAs rely on. This does incur a delay of approx. 1 hour in estimating the current hash rate, but the proposed DA should still produce significantly better results than BCH’s current DA. We note that a greater smoothing factor mitigates the drawbacks produced by the delay in hash rate estimation. Therefore, although we believe formulating the proposed DA using RTT is a valid alternative, the community can replace this using the MTP and configuring an appropriate smoothing factor.

5 SIMULATION

In this section, we empirically analyze the robustness of the Negative Exponential Filter DA by comparing it with BCH’s current DA (described in Section 3.1.2). We perform this analysis by simulating the evolution of a blockchain under different scenarios of hash rate fluctuations. In particular, we focus on mimicking the behavior of coin-hopping miners by adjusting the total hash rate in response to changes in profitability. Throughout this analysis we are mainly interested in the metrics we have already presented in the empirical analysis performed on BCH (see Section 3). For brevity, we abbreviate the Negative Exponential Filter DA derived in Section 4 with the acronym NEFDA and refer to BCH’s current DA simply as BCH.

5.1 Setup

To simulate mining, we adjust the total hash rate to create various relevant scenarios that a DA could be exposed to.

For simulating Amaury’s DA and the MTP variation of NEFDA it suffices to simulate the blockchain evolution on a per-block basis. As we have knowledge of the actual hash rate we model the block solve times by using a random number generator that produces values distributed according to an exponential distribution with rate parameter \(\lambda = H \cdot T / 2^{256}\), where \(T / 2^{256}\) represents the success probability of one hash computation.

On the other hand, simulating NEFDA with RTT requires more expensive computation as we have to update the target more frequently than at every block; we are satisfied with a per-second precision.

Through experimentation, we found that running the simulation for 100,000 blocks, corresponding to approx. two years of simulated time, is enough to clearly reveal any features of the metrics we are considering. We start simulations with a hash rate of 1 Exahashes per second, i.e. \(H = 10^{18}\) hashes/s, to maintain the scale of BCH’s hash rate. Aiming for an ideal inter block time of 10 minutes, we initialize the blockchain with an appropriate target of \(T \approx 2^{256} / (H \cdot 10\text{ mins}) \approx 1.92^{56}\).

5.2 Modeling Miner Behavior

If all miners would focus on short term profit, then all the hash rate would be directed solely towards the chain with the highest profitability. However, in practice this scenario does not occur as miners have socio-political beliefs and might incur switching costs due to their mining configuration. As we are only interested in simulating miners’ behavior and not the relation between specific chains, we simplify profitability computations by only comparing the estimated DARI with its initial value and assuming a constant exchange rate as most SHA-256 coins are highly correlated. Thus, we assume miners compute DARI as the ratio between the average target of the last \(N_{\text{DARI}}\) blocks and the initial target of the blockchain. Although hash rate fluctuations appear in response to DARI
unstable throughput: when the difficulty algorithm breaks

oscillations, it remains unclear what exact switching logic miners employ. To this end, we believe the following 3 model of miners are general enough to capture the behavior of any real miner.

*Idealistic miners* allocate all their hash rate to BCH regardless of how much profitability drops; they represent the baseline hash rate \( H_B \).

*Greedy coin-hopping miners* allocate all their hash rate \( H_G \) towards BCH only when the DARI increases by at least 5%.

*Variable coin-hopping miners* allocate part of their total hash rate \( H_V \) in relation to the current profitability. Although this relation is not clear in reality, we consider a model based on the logistic curve. The intention is to emulate both the initial stage when the hash rate increases exponentially as miners realize the advantage in profitability, and the later stage when the hash rate influx gradually slows down. The model directs all the hash rate away from or towards BCH, if drops or increases in profitability larger than 15%⁹ occur. Otherwise, a variation \( x \) between −15% and 15% leads to a contribution of \( H = H_v/(1 + e^{-6/0.15 \cdot x}) \) towards the total hash rate.

As expected, the reason for this massive discrepancy is the positive feedback loop present in BCH. Figure 9 shows the significant amount of positive correlation that appears at multiples of \( W \) (the size of BCH’s sliding window). Interestingly, BCH also shows negative correlation between blocks that are \( W/2 \) apart, indicating that there is a delay of 12 hours in hash rate estimation. On the other hand, NEFDA–RTT shows negative correlation between neighboring hour-buckets indicating that the DA rapidly responds to sudden hash rate fluctuations. By adjusting the target more quickly the DARIs oscillations are limited and miners are less incentivized to abandon mining (see Figure 16 from Appendix A for target, hash rate and DARI results).

5.4 Smoothing Factor Trade-offs
To examine the influence of the smoothing factor over the performance of NEFDA, we model a slightly more extreme environment by increasing the total hash rate of variable miners: \( H_V = 6 \times H_B \). Simulations are run using NEFDA–RTT with 3 smoothing factors that are multiples of the ideal inter block time, in order to represent 36, 72 and 144 blocks. An initial analysis of the average solve times: 600.0015 s for NEFDA–36, 599.9835 s for NEFDA–72 and 599.9895 s for NEFDA–144 shows how NEFDA in general copes much better than BCH even in more extreme environments. However, we do observe slight improvements in the targeting of the ideal average of 600s as the smoothing factor is increased.

![Figure 8: The probabilities of mining exactly \( k \) blocks in a one-hour period using NEFDA–RTT and BCH.](image)

![Figure 9: The autocorrelation in number of blocks mined per hour in NEFDA–RTT (top) and BCH (bottom)](image)
Figure 10: The probabilities of mining exactly $k$ blocks in a one-hour period using various smoothing factors for NEFDA–RTT.

Figure 10 reveals that all distributions have an appropriate center of mass, but lower smoothing factors slightly flatten the curve skewing the distribution from the ideal values. These small discrepancies affect the proportion of deserts: 6.77%, 4.14%, 2.81% and spikes: 4.84%, 3.06%, 2.14% (values are given in order for each of the smoothing factors considered: 36, 72 and 144 block). As shorter smoothing factors attribute greater relevance to recent blocks the algorithm risks being too reactive, and therefore over or underestimating the current network hash rate. As shown in Figure 11, this effect translates in more volatile targets, especially when underestimating the hash rate due to long block solve times. In turn, this leads to more volatile profitability which encourages coin-hopping behavior.

Figure 11: Comparison of the targets produced using different smoothing factors in NEFDA.

On the other hand, greater smoothing factors are not always desirable. Figure 12 shows the evolution of targets for each of the three variants of NEFDA when simulating a scenario in which the hash rate increases exponentially without any coin-hopping behavior. Although the lower smoothing factors still imply a more volatile target, the average solve times are now: 599.5518s for NEFDA–36, 599.1267s for NEFDA–72 and 598.2724s for NEFDA–144, which imply that the more reactive nature of NEFDA–36 and NEFDA–72 is desirable under these conditions.

Figure 12: Comparison of the targets produced using different smoothing factors in NEFDA under exponentially increasing hash rate.

5.5 Median Time Past Considerations

Lastly, we analyze the effects of using the MTP variant of NEFDA, by simulating it under the first scenario ($H_V = H_G = 4 \times H_B$ and $N_{DARI} = 6$ blocks.) As mentioned in Section 4.6, the MTP of the last 11 blocks introduces a lag of approx. 1 hour in hash rate estimations. A small smoothing factor implies NEFDA’s current hash rate estimate is mainly based on the most recent blocks. Due to the lag introduced by the MTP these recent blocks may not have their difficulties updated accordingly and a positive feedback might be introduced in the computation. However, unlike in BCH, where the positive feedback leads to a perpetuating loop due to the use of equal weights and a sliding window, NEFDA’s positive feedback quickly diminishes as the weights of the problematic blocks fade in time (see Figure 15 from Appendix A).

Figure 13: The probability distribution of exactly $k$ blocks being mined during a one hour period using various smoothing factors for NEFDA-MTP, compared to the ideal values.

This effect is reflected in skewed block distributions (see Figure 13). Notice how higher smoothing factors mitigate
the use of MTP. Therefore, if a community decides in favor of MTP, we suggest assuming the costs of a less reactive DA and selecting a smoothing factor much larger (e.g., 20 times) than the MTP’s window size.

6 RELATED WORK

The most extensive body of difficulty algorithm research has been done by the pseudonym zawy12, who provides a comprehensive overview of various difficulty algorithms in [24]. He also examines the difficulty instabilities in BCH in [22]. Regarding our proposed DA, zawy12 simulates the performance of two algorithms also based on exponential filters: ASERT [13] and EMA [25], which is an approximation of ASERT that avoids the computation of exponentials. We have become aware of ASERT which is essentially equivalent to our proposed DA, only after receiving the unpublished work of Mark B. Lundeberg from zawy12. Our additional contribution consists of the mathematical derivation of this algorithm, an outline of desirable properties, and motivation for the correction.

In [11] the author shows how Bitcoin’s difficulty algorithm causes shorter block solve-times in the case of an exponentially hash rate growth and defines an alternative model which achieves desired average block times in the long-run, yet is subject to increased solve time fluctuations.

A stochastic model is presented in [8], where the difficulty target is modeled as a random variable that is a function of previous block times.

The difficulty adjustment problem is addressed from a feedback control engineering perspective in [9], where the difficulty is adjusted on a per-block basis using a non-linear feedback controller based on a moving average filter of recent block timestamps for ensuring stable block solve times.

Bobtail is an alternative difficulty algorithm is presented in [2], which reduces block solve-time variance, yet comes at the cost of requiring significantly larger block headers.

In [3] a new difficulty algorithm is presented based on “bonded mining”, whereby a miner has to put up collateral and commit to mine at an offered hash rate over a period of blocks.

An investigation into the decision-making process driving miner behavior during times of BCH’s EDA is presented in [1].

A game theoretic framework of miners switching between PoW-compatible blockchains based on difficulty is proposed in [12]. The authors further show that BCH experienced a lack of loyal miners prior to the introduction of Amaury’s difficulty adjustment algorithm, which temporarily undermined the security of the system. The consequences of miners switching under the current BCH DA remained unexamined.

In parallel work, an explanation of how a DA based on an exponentially weighted moving average could mitigate the difficulty instabilities in BCH was provided by [20].

7 CONCLUSION

In this paper, we first provide a case study on BCH’s DA and show how the behavior of economically rational miners can lead to severe instabilities in throughput as a consequence of a positive feedback loop stemming from Amaury’s DA. Furthermore, we analyze the extent to which miners contribute towards sustaining the positive feedback loop in BCH’s DA and find that some large miners were able to successfully mine BCH only during periods of lower difficulty. The cyclic pattern in block solve times skews the distribution of the number of blocks per one-hour intervals therefore having a negative impact over the transaction throughput. In order to mitigate periods of undesired (either too low or too high) throughput, we propose and model a DA, which does not lead to the formation of a positive feedback loop and can cope effectively with sudden hash rate fluctuations. We explain how the proposed DA exhibits desirable properties in the form of history agnosticism and lack of any significant autocorrelation. Additionally, we show how the modeled DA is configurable in the level of responsiveness. Through simulations, we demonstrate how the proposed DA outperforms BCH’s current DA in terms of reducing target volatility and in turn high variations in block solve times. Furthermore, we show how to mitigate drawbacks introduced by the MTP variation of the Negative Exponential Filter DA, by configuring the smoothing factor. Therefore, the modeled DA constitutes a viable alternative for both large and small blockchains (in terms of baseline hash rate) when configured appropriately.
A ADDITIONAL FIGURES

Figure 14: Number of blocks mined per hour (top) and block difficulties (bottom) for the period during which the Emergency Difficulty Algorithm was active.

Figure 15: The autocorrelation in number of blocks mined per hour for NEFDA–MTP with smoothing factors 288, 144, 72 and 36, as well as for BCH in a coin hopping simulation of 100 000 blocks.

Table 2: The Bitcoin Cash addresses for selected miners.

| Address       | Miner      |
|---------------|------------|
| qpk4hk3wuxe2uqtqc97a8tzzrr6r5mlecz99sur4h | BTC.TOP    |
| qpc3bxylme9w87c5j2wdmsqlo6e84xcmmdswzy   | Antpool    |
| qrcuqadqzrp2uztj9wn5thqeqk22maiyxw4gmw6p  | ViaBTC     |
| qrd9kmeg4nqag3h5gzuv9j537pm7e85cauzece   | BTC.com    |
| qrj9yevckldh1zy3ewuz8f78s2w4jw5h6j9rmpq  | Huobi Pool |
| qp4ajqtcqy5mfhsq47kgm9whsappq5tw9z85h   | unknown    |
| qqq9v3ihlt0vga8w5ct6dx5aa8xep2v2svppp5cxn | unknown    |
| qzku6fvt20v6hau44r58tjupqzn3nqslf5qzf   | unknown    |

Figure 16: The targets, hash rates and blocks mined per hour for NEFDA–RTT and Amaury’s DA in a coin-hopping simulation of 100 000 blocks.
REFERENCES

[1] Vipul Aggarwal and Yong Tan. 2019. A Structural Analysis of Bitcoin Cash’s Emergency Difficulty Adjustment Algorithm. Available at SSRN 3383739 (2019).

[2] George Bissias and Brian Neil Levine. 2017. Bobtail: A Proof-of-Work Target that Minimizes Blockchain Mining Variance (Draft). arXiv:cs.CR/1709.08750

[3] George Bissias, David Thibodeau, and Brian N Levine. 2019. Bonded Mining: Difficulty Adjustment by Miner Commitment. In Data Privacy Management, Cryptocurrencies and Blockchain Technology. Springer, 372–390.

[4] BitcoinABC. 2017. Difficulty Adjustment Algorithm Update. https://www.bitcoinabc.org/2017-11-01-DAA. https://www.bitcoinabc.org/2017-11-01-DAA Accessed: 2019-10-16.

[5] BitcoinABC. 2020. Bitcoin ABC. https://www.bitcoinabc.org. https://www.bitcoinabc.org Accessed: 2020-03-02.

[6] Alex Boverman. 2011. Timejacking & Bitcoin. https://culubas.blogspot.com/2011/05/timejacking-bitcoin_802.html. https://culubas.blogspot.com/2011/05/timejacking-bitcoin_802.html Accessed: 2019-11-15.

[7] forklo. [n.d.]. https://fork.lol/reward/dari/btc. https://fork.lol/reward/dari/btc Accessed: 2020-03-31.

[8] Daniel Fullmer and A Stephen Morse. 2018. Analysis of Difficulty Control in Bitcoin and Proof-of-Work Blockchains. In 2018 IEEE Conference on Decision and Control (CDC). IEEE, 5988–5992.

[9] Geir Hovland and Jan Kucera. 2017. Nonlinear Feedback Control and Stability Analysis of a Proof-of-Work Blockchain. Modeling, Identification and Control 38, 4 (2017), 157–168. https://doi.org/10.4173/mic.2017.4.1

[10] Tamás Király and Lilla Lomoschitz. 2018. Profitability of the coin-hopping strategy. EGRES quick proof 2018-03 (2018).

[11] Daniel Kraft. 2016. Difficulty Control for Blockchain-Based Consensus Systems. https://bitcoinsv.io/. In Peer-to-Peer Networking and Applications, Vol. 9. 397–413. https://doi.org/10.1007/s12083-015-0347-x

[12] Yujin Kwon, Hyoungshick Kim, Jinwoo Shin, and Yongdae Kim. 2019. Bitcoin vs. Bitcoin Cash: Coexistence or Downfall of Bitcoin Cash? arXiv preprint arXiv:1902.11064 (2019).

[13] Mark B. Lundeberg. 2019. Static difficulty adjustments, with absolutely scheduled exponentially rising targets (DA-ASERT). (2019). unpublished.

[14] Dmitry Meshkov, Alexander Chepurnoy, and Marc Jansen. 2017. Short paper: Revisiting difficulty control for blockchain systems. In Data Privacy Management, Cryptocurrencies and Blockchain Technology. Springer, 429–436.

[15] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf. https://bitcoin.org/bitcoin.pdf Accessed: 2015-07-01.

[16] Wouter Penard and Tim van Werkhoven. 2008. On the secure hash algorithm family. Cryptography in Context (2008), 1–18.

[17] BitMEX Research. 2019. Bitcoin CashâĂŹs October 2019 Hashrate Volatility Increase. https://blog.bitmex.com/bitcoin-cashs-october-2019-hashrate-volatility-increase/ Accessed: 2019-11-08.

[18] Bitcoin SV. [n.d.]. Bitcoin SV. https://bitcoinsv.io/ Accessed: 2019-10-21.

[19] Pawel Szalachowski. 2018. (Short Paper) Towards More Reliable Bitcoin Timestamps. In 2018 Crypto Valley Conference on Blockchain Technology (CVCBT). IEEE, 101–104.

[20] Jonathan Toomin. 2020. The BCH difficulty adjustment algorithm is broken. Here’s how to fix it. https://www.reddit.com/r/btc/comments/fanc6o/the_bch_difficulty_adjustment_algorithm_is_broken/ Accessed: 2020-03-01.

[21] zawy12. 2018. Timestamp Attacks. https://github.com/zawy12/difficulty-algorithms/issues/30. https://github.com/zawy12/difficulty-algorithms/issues/30 Accessed: 2019-11-15.

[22] zawy12. 2019. BCH needs a new DA. https://github.com/zawy12/difficulty-algorithms/issues/47. Accessed: 2019-10-01.

[23] zawy12. 2019. Oscillations in Simple Moving Averages. https://github.com/zawy12/difficulty-algorithms/issues/48. Accessed: 2019-10-01.

[24] zawy12. 2019. Summary of Difficulty Algorithms. https://github.com/zawy12/difficulty-algorithms/issues/50. Accessed: 2020-03-02.

[25] zawy12. 2019. Using EMA for BCH’s new DA. https://github.com/zawy12/difficulty-algorithms/issues/49. Accessed: 2020-03-02.