Unstable Throughput: When the Difficulty Algorithm Breaks

Dragos I. Ilie, Sam M. Werner, Iain D. Stewart and William J. Knottenbelt
Imperial College London
London, United Kingdom

***Abstract*—In Proof-of-Work blockchains, difficulty algorithms serve the crucial purpose of maintaining a stable transaction throughput by dynamically adjusting the block difficulty in response to the miners' constantly changing computational power. Blockchains that may experience severe hash rate fluctuations need difficulty algorithms that quickly adapt the mining difficulty. However, without careful design, the system could be gamed by miners using coin-hopping strategies to manipulate the block difficulty for profit. Such miner behavior results in an unreliable system due to the unstable processing of transactions.**

We provide an empirical analysis of how Bitcoin Cash's difficulty algorithm design leads to cyclicality in block solve times as a consequence of a positive feedback loop. In response, we mathematically derive a difficulty algorithm using a negative exponential filter which prohibits the formation of positive feedback 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 transaction throughput.

***Index Terms*—Blockchain, Bitcoin, Bitcoin Cash, Difficulty Algorithm, Mining, Coin-Hopping**

I. Introduction

Proof-of-Work (PoW) blockchains offer a decentralized mechanism for recording data in a trustless and immutable manner. To reach consensus over the ordering and validity of blocks, miners participate in a leader election process by solving a computationally intensive puzzle named Proof-of-Work. The first miner to find a valid solution appends a block and receives a reward for the invested computational effort. To ensure stable transaction throughput, a difficulty algorithm (DA) adjusts the difficulty of the PoW puzzle in response to changes in the miners' computational power. 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 in Bitcoin Cash (BCH) [3], [1], the cryptocurrency with the 7th highest market capitalization of $5bn[1]. Its developers have announced that fixing the high variations in block solve times is one of their main development goals for 2020 [2]. Two proposals [7], [15] have been put forward to replace BCH's DA known as `cw-144`. In this paper, we present a negative exponential filter DA (NEFDA) which is similar to these proposals and is explicitly referenced[2] by one of them [7]. Additionally, we define desirable properties exhibited by NEFDA and prove their benefits by comparing the performance of NEFDA with `cw-144` in a simulated mining scenario.

[1]Data obtained from: https://coinmarketcap.com. Accessed: 22-11-2020.

II. 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 [21]. He examines the difficulty instabilities in BCH in [19] and simulates the performance of various DAs, including ASERT [12][3] and EMA [22], [5], [6], which is an approximation of ASERT that avoids the computation of exponentials. Our work differentiates by providing a formal derivation of NEFDA starting from first principles and an outline of the desirable properties achieved.

III. Background

In this section, we introduce readers to the difficulty algorithms used in Bitcoin (BTC) [14] and BCH.

A *difficulty algorithm* (DA) is a fundamental component of PoW blockchains as it ensures a stable transaction throughput by adjusting the hardness of generating a PoW solution. 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, 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. Therefore, DAs estimate the current hash rate based on the difficulties and solve times of blocks in the recent past. The extent to which a DA is able to minimize the lag between the actual hash rate and the estimated one is regarded as the reactiveness of the algorithm. Blockchains with relatively stable hash rate can afford to use a less reactive DA to reduce volatility in difficulty, allowing miners to predict expected rewards over near-term time scales. The frequency of difficulty adjustments also affects the reactiveness of a DA.

[2]The reference has been made to an earlier pre-printed version of this paper.

[3]We have become aware of ASERT, which is essentially equivalent to NEFDA, after receiving the unpublished work [11] of Mark B. Lundeberg from zawy12.

A. BTC Difficulty Algorithm

In BTC, the new difficulty D' is updated every 2 016 blocks (approx. 2 weeks) based on the previous difficulty D, using the following formula:

$$D' = D \cdot \max\left(\min\left(\frac{2\,016 \cdot T}{T_A}, 4\right), \frac{1}{4}\right) \tag{1}$$

where T is the ideal inter-block time (i.e. 10 minutes) and T_A is the time it actually took to mine the last 2 016 blocks.

B. BCH Difficulty Algorithm

BCH's DA, referred to as `cw-144`, attempted to increase responsiveness to both effluxes and influxes of hash rate by performing difficulty adjustments on a per-block basis. The difficulty D of a new block is derived from the estimated hash rate, $\widehat{H}$, and the ideal inter-block time, T. To this end, $\widehat{H}$ is computed using a simple moving average with a sample size of approx. 144 blocks from B_{start} to B_{end}. The DA computes CW, the amount of chain 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: $\widehat{H} = CW/T_A$, 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. Thus, the equation for the new difficulty is:

$$D = \widehat{H} \cdot T = \frac{\sum_{i=start}^{end} \mathrm{diff}(B_i)}{T_A} \cdot T \tag{2}$$

IV. Empirical Analysis of BCH's DA

In this section, we provide an empirical analysis of issues stemming from the use of `cw-144` in BCH.

A. Oscillations in Number of Blocks Mined per Hour

As intended, `cw-144` 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.

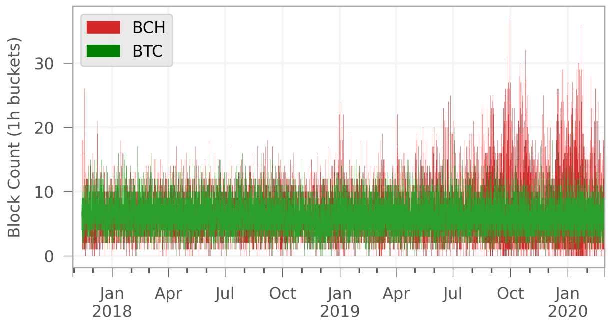

Fig. 1. Number of blocks mined per hour in BTC and BCH.

B. 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 miners' economically rational behavior.

From equation (2) it is apparent that `cw-144` relies (in part) on the relationship of inverse proportionality between T_A and the estimated hash rate, $\widehat{H}$. The same relationship exists between the hash rate fluctuations and the solve times of newly mined blocks. As new solve times are added to T_A, the result of these two relations is that $\widehat{H}$ 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 $\widehat{H}$. Short solve times 144 blocks in the past cause a relative increase in T_A which yields a lower-than-expected $\widehat{H}$. Similarly, long solve times falling off the window imply a relative decrease in T_A and therefore produce a higher $\widehat{H}$. This influence constitutes positive feedback that results in correlation between solve times 144 blocks (24 hours) apart, as can be seen in Figure 2.

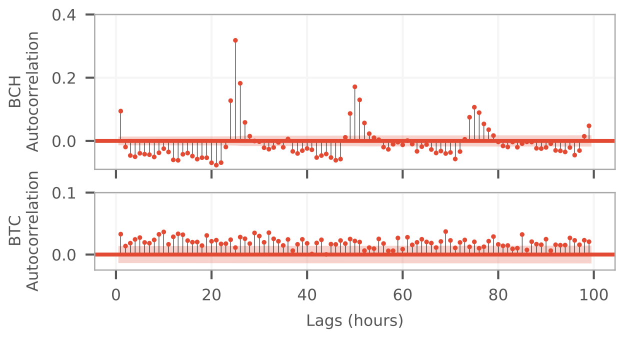

Fig. 2. The autocorrelation in number of blocks mined per hour in BCH and BTC since `cw-144` was deployed.

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 [13], [9], [10]. 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 until M_{CH} leave the network and the hash rate returns to its original value. However, the difficulty is now too high for the network, so a series of blocks with long solve times is produced. The positive feedback of blocks falling off the window, causes this pattern of short solve times followed by long solve times to repeat forming a positive feedback loop. As can be seen from Figure 2, only BCH experiences such a feedback loop because its base hash rate is approx. 3% of BTC's hash rate so coin-hopping miners have a much more significant impact. This phenomenon has also been examined by [20], [19], [17], [8].

V. Negative Exponential Filter Difficulty Algorithm

In this section, we mathematically derive a negative exponential filter difficulty algorithm (NEFDA) based on a common technique for removing noise from time series data known as exponential smoothing. Throughout we make use of the following notation:

$D_i \leftarrow$ difficulty of block i

$t_i \leftarrow$ time of block i

$st_i \leftarrow$ solve time of block $i : st_i = t_i - t_{i-1}$

$T \leftarrow$ ideal block solve time (e.g. 10 minutes)

$S \leftarrow$ decay/smoothing factor (see Section V-B)

$\widehat{H}_i \leftarrow$ estimated hash rate at block i

For simplicity, index 0 refers to the block at which the new DA is deployed, while index n refers to the next block to be appended. Thus, t_n and $\widehat{H}_n$ represent the current time and network hash rate, respectively. NEFDA uses real time targeting (RTT), i.e. the difficulty D_n of the block that is being mined dynamically adjusts as time passes.

$$D_n = D_0 e^{\frac{t_0 + nT - t_n}{S}}$$

Considering RTT is not a popular technique, we argue at length for its safety in Section V-C. In the remainder of this section, we show how this formula is derived from first principles.

1) Estimating Current Hash Rate: Difficulty algorithms are in the business of estimating the current network hash rate, $\widehat{H}_n$. As the actual function of hash rate cannot be known at any given time we rely on sampling when information is available, i.e. when blocks are mined. On average, the difficulty D_i represents the number of hashes computed throughout the interval $(t_{i-1}, t_i]$. Approximating that D_i hashes are computed at time t_i we can estimate the current hash rate $\widehat{H}_n$ using exponential smoothing over the series of block difficulties, i.e. by taking their exponentially weighted average.

$$\widehat{H}_n = \frac{\sum_{i=0}^{n-1} D_i e^{\frac{t_i - t_n}{S}}}{\int_{-\infty}^{0} e^{\frac{x}{S}} dx} = \frac{1}{S} \sum_{i=0}^{n-1} D_i e^{\frac{t_i - t_n}{S}} \tag{3}$$

2) Difficulty Computation: Therefore, the difficulty D_n of the next block is:

$$D_n = T \cdot \widehat{H}_n = \frac{T}{S} \sum_{i=0}^{n-1} D_i e^{\frac{t_i - t_n}{S}} \tag{4}$$

$$= \frac{T}{S} \sum_{i=0}^{n-1} D_i e^{\frac{t_i - t_{n-1}}{S}} e^{\frac{t_{n-1} - t_n}{S}} \tag{5}$$

$$= e^{\frac{-st_n}{S}} \left(\frac{T}{S} \sum_{i=0}^{n-2} D_i e^{\frac{t_i - t_{n-1}}{S}} + \frac{T}{S} D_{n-1} \right) \tag{6}$$

$$= e^{\frac{-st_n}{S}} \left(D_{n-1} + \frac{T}{S} D_{n-1} \right) \tag{7}$$

$$= D_{n-1} \left(1 + \tfrac{T}{S}\right) e^{\frac{-st_n}{S}} \tag{8}$$

When unwinding the recurrence relation (8) all the way to D_0 we obtain:

$$D_n = D_0 \left(1 + \tfrac{T}{S}\right)^n \prod_{i=1}^{n} e^{\frac{-st_i}{S}} = D_0 \left(1 + \tfrac{T}{S}\right)^n e^{\frac{t_0 - t_n}{S}} \tag{9}$$

3) Correction: Notice that when $T \ll S$ we can approximate $1 + T/S \approx e^{T/S}$. In fact, this is actually a correction needed to mitigate the bias introduced when considering a discrete series of difficulties instead of the continuous function of hash rate. To prove this, we replace the constant term: $1 + T/S$ with c and compute its value when the DA operates under a simple theoretical scenario. Specifically, we assume the hash rate remains constant for many blocks between m and n. Thus, we expect the average rate of change in difficulty $\overline{R} = 1$, indicating that on average the difficulty does not change. We take the geometric mean of ratios of consecutive difficulties from block m to n and use equation (9) with the c replacement:

$$\overline{R} = \sqrt[n-m]{\prod_{i=m+1}^{n} \frac{D_i}{D_{i-1}}} = \sqrt[n-m]{\frac{D_n}{D_m}} \tag{10}$$

$$= \sqrt[n-m]{\frac{D_0 c^n e^{(t_0 - t_n)/S}}{D_0 c^m e^{(t_0 - t_m)/S}}} = c \cdot e^{\frac{t_m - t_n}{(n-m)S}} \tag{11}$$

Assuming the DA is working correctly, the average solve time of blocks from m to n is $(t_n - t_m)/(n - m) = T$. Replacing in equation (11) we obtain: $\overline{R} = 1 = c \cdot e^{\frac{-T}{S}} = 1$ which implies $c = e^{T/S}$.

Therefore, the correction is indeed justified and applying it in equations (8) and (9), gives the following relative and absolute forms:

$$D_n = D_{n-1} e^{\frac{T - st_n}{S}} \tag{12}$$

$$D_n = D_0 e^{\frac{t_0 + nT - t_n}{S}} \tag{13}$$

A. Properties

1) History Agnosticism: The distribution of blocks in a given time period does not influence the difficulty of the block currently being mined. This property is desirable as block arrivals should be independent of each other so the difficulty of a block should not depend on the history of the chain. Equation (13) shows how the difficulty at time t_α depends only on the blockchain height, regardless of whether blocks were mined a long time in the past, in the last hour, or equally distributed in time.

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 may still produce temporary spikes or deserts, yet their duration will be much shorter. However, these will not create a positive feedback loop as the distribution of blocks in time has no influence on the future. Therefore, the inherent negative feedback present in NEFDA is the only force acting on solve times.

B. Smoothing Factor Considerations

The smoothing factor S has the function of configuring the reactiveness of NEFDA by setting the maximum rate of upward adjustments for the difficulty. More specifically, the difficulty can increase by at most a factor of e in S/T blocks. Depending on the requirements of the application, S should be chosen carefully: blockchains that are expected 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, while blockchains with a relatively stable hash rate (e.g. BTC) can choose larger values for S to reduce the difficulty's volatility. There is no direct relationship between the smoothing factor of an exponential moving average and the sample size of the simple moving average used in `cw-144`, as their operation is considerably different, but our simulations as well as other empirical studies [21] suggest that in order to obtain similarly stable difficulties the smoothing factor 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, suggests S should be set at approx. 12 hours.

C. Real Time Targeting Considerations

Real time targeting DAs assume miners have no incentive to report incorrect timestamps. To prove this assumption we compare NEFDA's RTT formulation with BCH's `cw-144` and argue that NEFDA reduces the incentives for timestamp manipulation. In `cw-144` reporting a dishonest timestamp, with a value in the future, would lower the difficulty for the next 144 blocks. This creates short term incentives for other miners to accept the dishonest block as they also benefit from the reduced difficulty even if they are not planning to be dishonest themselves. In contrast, NEFDA's history agnosticism implies that only the difficulty of the block with dishonest timestamp is affected, so there are no incentives for other miners to accept it. In fact, building on a dishonest block (B_i) implies mining towards a difficulty that is $e^{T/S}$ times higher than that of the previous block (B_{i-1}). Thus, a miner would only accept this block if it is willing to report an even higher timestamp to mitigate the increase in difficulty. This behavior leads to an unstable chain as it could be replaced by a potentially shorter chain with more accumulated work (higher difficulty blocks), so honest miners would not risk accepting blocks with dishonest timestamps. Only an attack supported by a majority of the hash rate would be successful, which is no different than 51% attacks [16], [4], [18] that are currently possible in BCH or even BTC.

VI. Simulation

In this section, we empirically analyze the robustness of NEFDA, by comparing it with `cw-144` over a 100 000 blocks period. We simulate the behavior of coin-hopping miners by adjusting the total hash rate in response to fluctuations in profitability. To stress test NEFDA, we consider a rather extreme scenario where greedy and variable coin-hopping miners have hash rates $H_G = H_V = 4 \times H_B$ (the base hash rate). Greedy miners allocate all their hash rate, H_G, when profitability is 5% higher compared to the initial value while variable miners allocate their hash rate using a logistic curve: $H = H_V/(1+e^{-6/0.15\cdot x})$ where x represents the change in profitability.

A brief analysis of the average solve times: 599.97 s for NEFDA and 604.34 s for `cw-144`, already reveals how NEFDA achieves a more appropriate value under this extreme scenario. Furthermore, Figure 3 plots the autocorrelation in number of blocks mined per hour in `cw-144` and NEFDA. For the former, a significant amount of positive autocorrelation appears at multiples of 24 (the number of hours in BCH's sliding window). The negative correlation between periods that are 12 hours apart is expected considering the effect of averaging over a sliding window is to estimate the middle value. This delay in estimation is what gives coin-hopping miners the necessary time to mine many blocks while the difficulty is still low. On the other hand, NEFDA shows negative correlation between consecutive hour-buckets which indicates that it responds rapidly to hash rate fluctuations. More importantly, no positive feedback is present which is what is expected given the properties of history agnosticism and lack of autocorrelation.

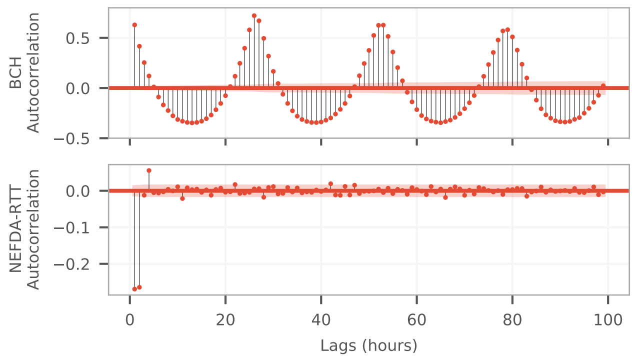

Fig. 3. The autocorrelation in number of blocks mined per hour in `cw-144` (top) and NEFDA (bottom)

VII. Conclusion

We have showed how the behavior of economically rational miners can lead to severe instabilities in transaction throughput as a consequence of a positive feedback loop in `cw-144`. To mitigate periods of undesired (either too low or too high) transaction throughput, we derived NEFDA, a DA which does not lead to the formation of a positive feedback loop and can cope effectively with sudden hash rate fluctuations. We explained how NEFDA exhibits desirable properties in the form of history agnosticism and lack of significant positive autocorrelation and demonstrated through simulations how NEFDA reduces target volatility, and in turn high variations in block solve times. Ultimately, NEFDA constitutes a viable alternative for both large and small blockchains (in terms of baseline hash rate) when configured appropriately and may thus guarantee more stable transaction throughput.

References

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

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

[3] BitMEX Research. BCH's hashrate volatility increase. https://blog.bitmex.com/bitcoin-cashs-october-2019-hashrate-volatility-increase/, 2019. 2019-11-08.

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

[5] J. Eliosoff. Added "ema" algos (exp moving avg). https://github.com/kyuupichan/difficulty/pull/26#issuecomment-342290398, 2017. Accessed: 13-11-2020.

[6] J. Eliosoff. Add wtema-72 weighted-target exponential moving average. https://github.com/kyuupichan/difficulty/pull/30#issuecomment-355854001, 2018. Accessed: 13-11-2020.

[7] freetrader, J. Toomim, C. Culianu, and M. Lundeberg. 2020-nov-15 ASERT difficulty adjustment algorithm (aserti3-2d), 2020. Accessed: 13-11-2020.

[8] jtoomim. BCH upgrade proposal: Use ASERT as the new DAA, 2020.

[9] T. Király and L. Lomoschitz. Profitability of the coin-hopping strategy. *EGRES quick proof*, 3(2018), 2018.

[10] Y. Kwon, H. Kim, J. Shin, and Y. Kim. Bitcoin vs. bitcoin cash: Coexistence or downfall of bitcoin cash? *arXiv preprint arXiv:1902.11064*, 2019.

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

[12] M. B. Lundeberg. Static difficulty adjustments, with absolutely scheduled exponentially rising targets (DA-ASERT) — v.2. Accessed: 13-11-2020., 2020.

[13] D. Meshkov, A. Chepurnoy, and M. Jansen. Short paper: Revisiting difficulty control for blockchain systems. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 429–436. Springer, 2017.

[14] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, Dec 2008. Accessed: 2015-07-01.

[15] A. Sechet. Announcing the Grasberg DAA, 2020. Accessed: 13-11-2020.

[16] P. Szalachowski. (short paper) Towards more reliable bitcoin timestamps. In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 101–104. IEEE, 2018.

[17] J. Toomim. 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/, 2020. Accessed: 2020-03-01.

[18] zawy12. Timestamp attacks on difficulty algorithms. https://github.com/zawy12/difficulty-algorithms/issues/30, 2018. Accessed: 2019-11-15.

[19] zawy12. BCH needs a new Difficulty Algorithm. https://github.com/zawy12/difficulty-algorithms/issues/47, 2019. Accessed: 2019-10-01.

[20] zawy12. Oscillations in simple moving averages. https://github.com/zawy12/difficulty-algorithms/issues/48, 2019. Accessed: 2019-10-01.

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

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