

Fast Blocks and Furious Adjustments: Satoshi Drift

Demetris Kyriacou¹(✉), Matthieu Babak², Iain Stewart¹, and
William J Knottenbelt¹

¹ Imperial College London, London, UK
`demetris.kyriacou22@imperial.ac.uk`

² ENSTA Paris, France

Abstract. Difficulty adjustment algorithms (DAAs) are a vital component of every Proof-of-Work (PoW) blockchain. They regulate mining difficulty which in turn modulates interblock times. Maintaining a stable block-production rate and consistent transaction throughput is crucial for the smooth operation of a blockchain and can positively affect its reputation. The challenging part of a DAA is that the actual blockchain hash rate is unknown. Therefore, difficulty adjustments must be based on historical data, estimations and/or predictions. For Bitcoin (BTC), things are relatively simple. The vast hash power of Bitcoin’s network makes it largely immune to fluctuations, allowing the difficulty to be adjusted infrequently (approx. every two weeks) without affecting throughput. On the other hand, the rest of the PoW blockchains, such as Bitcoin Cash (BCH), are more susceptible to hash power fluctuations and require a more adaptive DAA which adjusts the mining difficulty after every block. Such DAAs already exist, have been battle-tested and have proven effective at maintaining average interblock times close to the target. Nevertheless, to our knowledge, no DAA successfully addresses the problem of accumulated drift. In fact, both BTC and BCH halvings of April 2024 occurred almost 9 months ahead of schedule due to this drift. In this paper, we propose a novel yet very simple DAA based on a negative exponential filter that not only keeps the block-production rate stable in the long run but also eliminates any already-accumulated drift.

Keywords: Difficulty Adjustment Algorithm · Mining difficulty · Proof of Work · Bitcoin Cash.

1 Introduction

Over recent years, various difficulty adjustment algorithms (DAAs) have been proposed and implemented in blockchains that support Proof-of-Work (PoW) consensus mechanisms. Some of them proved to be stable and robust while others deteriorated to relics amidst disaster.

The primary purpose of a DAA is to regulate interblock times of a PoW blockchain by adjusting the difficulty of mining a block. In an ideal world where the real total hash rate of a blockchain network is known, adjusting difficulty

would be trivial. In practice, however, the hash rate of a network is unobservable and can only be estimated using data about previously mined blocks such as interblock times and past difficulties. Although some miners proudly and openly declare their hash power, some of them remain rather cryptic. Additionally, there are miners who are not always loyal to mining on a single blockchain but instead perform what is called coin-hopping strategies to maximise their profit [16]. Interestingly, the hash power behavior of the network resembles that of financial markets. Both are driven by underlying human, technological, and architectural components. Another common trait is that poorly designed algorithms or subtle interactions between algorithms can disrupt their orderly regulation [3].

However, even if the DAA does a good job in controlling interblock times in short-term, there is another problem than can be gradually developed in long-term. *Drift* or *skew* is created when blocks are, on average, being mined faster or slower than normal. For BTC it happens mainly because the DAA cannot keep up with the increase of hash rate due to increase in mining popularity and hardware capabilities. Consequently, it accumulates drift every time there is a major leap forward in mining technology.

In August 2017, disputes within the Bitcoin community over scalability and functionality led to the creation of Bitcoin Cash (BCH) [24, 25] as a hard fork. After the hard fork, Bitcoin Cash continued its own, more turbulent path, as an independent blockchain and experienced another schism as a result of Bitcoin Cash’s hash war [17]. The blockchain was split into what is still referred to as Bitcoin Cash (BCH) [6] and Bitcoin SV (BSV) [7]. This paper focuses on the former. To this day, BTC uses its original DAA with biweekly difficulty adjustments, while BCH has experimented with various DAAs that adjust the mining difficulty at every block [4, 9]. Both of them though have piled on more drift to the inherited pre-fork drift and, as result, their halvings on April 2024 happened almost 9 months prior to their originally-scheduled dates.

In this paper, we choose to focus on BCH due to its unsettled past of DAAs along with the fact that it represents one of the most popular PoW blockchains whose mining difficulty is adjusted at every block. Additionally, BCH’s vulnerability to hash rate fluctuations make it more challenging and interesting to investigate. Lastly, our research builds on previous work by Ilie et al. [13].

Concretely, we present an innovative DAA that alleviates the problem of drift by gradually eradicating any that has already been accrued. Although designed primarily for BCH, our DAA is compatible with most PoW blockchains that adjust their mining difficulty at every block. Furthermore, the responsiveness of the DAA can be adjusted using a smoothing factor. Initially, we perform an empirical analysis on the various DAAs used in BCH. We investigate methods of evaluating performance of DAAs and reveal flaws and weaknesses of each one. Moreover, we define our novel DAA based on a negative exponential filter and explain the rationale for its tunable parameters. Finally, we substantiate our findings with simulations that verify the robust performance of our DAA and compare it to the existing DAA of BCH.

2 Related work

Bitcoin Cash has adopted multiple DAAs since the hard fork with Bitcoin; all of them are analysed in depth in Section 4. A noteworthy proposal for BCH that was never adopted was Amaury Sechet’s Grasberg DAA [1]. Grasberg DAA was based on a negative exponential filter and included a drift correction mechanism. It aimed to ensure a stable and more predictable block-production rate and minimise the profit gap between loyal mining and coin-hopping mining. However, the proposal was rejected as it was deemed too complicated and inadequately substantiated. Furthermore, few members of the BCH community argued against the need for drift correction [11, 15, 20]. The main difference between our DAA and Grasberg is that ours is very simple, as we only make a minor adjustment to the current DAA to achieve drift correction. Additionally, our DAA converges to the current DAA as the drift converges to 0. In addition, we provide a detailed analysis of the mathematical background, as well as extensive simulations.

3 Background

3.1 Hash rate

Hash rate is a measurement of a computational unit’s ability to produce outputs of a hashing function. It is calculated as hashes per second and in blockchain terms it is the main method of evaluating computational power [14]. Therefore, throughout this paper, the terms hash rate, hash power and computational power will be used interchangeably.

3.2 Proof of Work (PoW) Mining

Each valid block consists of a collection of transactions along with a header that contains important information about the block such as its timestamp and the root hash of transactions. One of the block header fields is called *nonce*. The sole purpose of a nonce is to pseudorandomly alternate the output of a hash function that takes block header as input. By requiring the hash of a block header to be smaller than a predefined threshold or *target*, the trivial process of creating a block becomes a challenging cryptographic puzzle called PoW mining [21, 10]. PoW is based on the idea that mining a block is computationally demanding, so that no malicious user has the ability to mine blocks faster than the rest of the network. The only way a malicious user can do that is by possessing at least 51% of network’s computational power [8].

3.3 Target

Target is a numerical value that a hashed block header must be less than or equal to in order for a new block to be mined successfully [23]. The probability of a hash being smaller than or equal to the target is:

$$P(\text{hash} \leq \text{target}) = \frac{\text{target} + 1}{2^{256}}$$

where every hash has 2^{256} possible outputs $[0, 2^{256} - 1]$

Also, each attempt to mine a block is a Bernoulli trial with a probability of success equal to $\frac{\text{target} + 1}{2^{256}}$. Thus, the expected work or the number of hashes to mine a single block is:

$$\mathbb{E}[W] = 2^{256} / (\text{target} + 1) \approx 2^{256} / \text{target}$$

3.4 Difficulty

The idea that a higher target means that it is more difficult to mine a block while a lower target means it is easier can be perceived as somewhat counter-intuitive. Thus, the notion of *difficulty* was introduced. Difficulty represents how difficult it is to mine a block using some current target relative to how difficult it would have been if the highest possible target was used [23]. The maximum target is equal to the target of the very first block of Bitcoin and is almost equal to 2^{224} .

$$D = \frac{\text{max_target}}{\text{target}} = \frac{\text{first block's target}}{\text{target}} \approx \frac{2^{224}}{\text{target}}$$

Expected number of hashes to mine a single block can also be expressed as:

$$\mathbb{E}[W] = D \cdot 2^{32}$$

Notably, if the real hash rate H_n at any given time was observable and since the ideal interblock time T is constant, setting the difficulty for each block would be trivial:

$$D_n \cdot 2^{32} = \mathbb{E}[W] = H_n \cdot T$$

However, in reality it is unattainable to observe the actual hash rate of the network *a priori*. In fact, it is impossible to precisely calculate the real hash rate even *a posteriori*. For this reason, designing a robust DAA poses a challenge.

3.5 Difficulty Adjustment Algorithm (DAA)

A Difficulty Adjustment Algorithm (DAA) is a vital element of every PoW blockchain. Its main task is to control mining difficulty to modulate the time between the creation of every new block in the blockchain.

In the short term, a DAA should ensure a stable block-production rate and relatively steady transaction throughput which makes the blockchain dependable. The reliability of a blockchain improves user experience but is also reflected in its coin's value. In the long run, a DAA controls coin issuance, timing of the halvings, and miners' profit. By lowering miners' reward, scarcity is maintained and this is a measure to protect from inflationary effects. Moreover, lowering supply increases demand and affects the price of the coins [18].

Today, more than fifteen years later, Bitcoin is the most well-established cryptocurrency and its network hash power is vast and invariant to minor oscillations. As a result, its primitive original DAA [21] that adjusts mining difficulty on a biweekly basis is sufficient. But, alas, not every PoW blockchain enjoys the same broad and stable support by miners as BTC. Almost all PoW altcoins are plagued by highly volatile hash power, mainly due to coin-hopping strategies, and cannot afford to wait a fortnight to alter mining difficulty.

There are multiple factors that influence a miner's coin-hopping strategy, like mining difficulty or coin value, but the main motive is single and clear, maximising profit. Since various PoW blockchains use the same mining mechanism, it is only a matter of flipping a switch for a miner to "hop" to mining the most profitable coin at any given time [16].

Coin-hopping is when a miner switches the entirety or a part of his hash power from mining one coin to another. Hence, this switch alters the total hash

power of both blockchain networks. When performed on a large scale, coin-hopping can cause serious fluctuations in an altcoin network’s total hash rate and DAAs are required to be able to keep up with this. As a result, numerous DAAs were developed during the previous decade, many of which proved to be resistant to hash rate fluctuations and capable of maintaining relatively stable block-production rates. What was not accounted for though is the long-term effect of DAAs’ inability to keep up with the general increase of hash power in blockchains, primarily due to technology advances.

3.6 Drift (Skew)

Satoshi Nakamoto foresaw the increasing hardware speed and mining interest back in 2008 when the Bitcoin whitepaper was issued [21]. Indeed, as you can see in Fig. 1, BTC’s difficulty has a tendency to increase over the years. Furthermore, there are two obvious rapid jumps in difficulty. The first was around 2010 with the introduction of GPU mining, and the second was around 2013 with the introduction of ASIC mining.

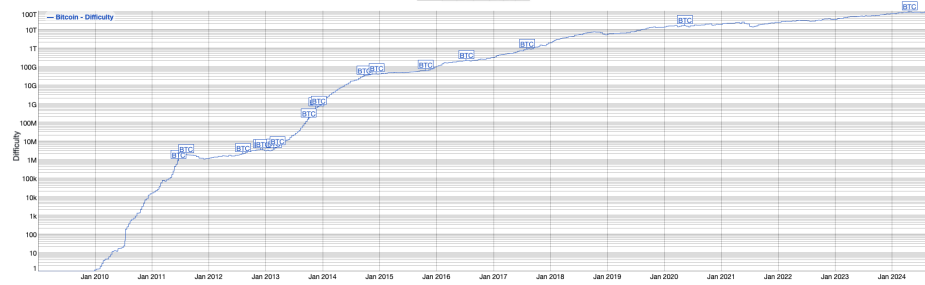


Fig. 1. BTC difficulty history, source: <https://bitinfocharts.com/comparison/bitcoin-difficulty.html>

Of course, BTC’s DAA was able to cope with the difficulty increase and restore the order of block-production rate without significant impact on transaction throughput. However, as shown in Fig. 2, drift was gradually built up because – on average – blocks were mined faster than every 10 minutes. BTC’s DAA remained the same and was never adjusted to mitigate the problem of drift. As a result, the 2nd, 3rd and 4th BTC halvings occurred 139, 234 and 252 days (respectively) ahead of schedule (see Fig. 2 or Table 1 in Appendix A). Meanwhile, when BCH forked out of BTC in August 2017, it inherited any already-accumulated drift. Its stormy quest to find a suitable DAA only made the drift problem worse. Thus, BCH’s 3rd and 4th halvings happened 267 and 269 days (respectively) ahead of their ideal time (see Fig. 2 or Table 1 in Appendix A).

4 Current and Legacy DAAs of BCH

4.1 BTC DAA

Prior to the August 2017 fork, Bitcoin Cash used Bitcoin’s DAA according to which the mining difficulty is updated every 2016 blocks. The DAA formula prevents extreme changes in difficulty by capping adjustments, allowing the new difficulty to be no more than four times higher or lower than the previous value [22].

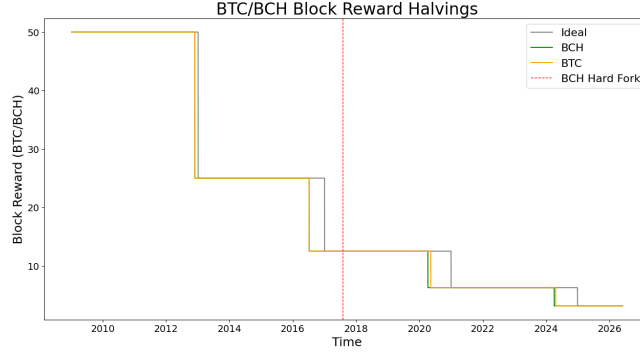


Fig. 2. BTC/BCH Block Reward Halvings.

This DAA has allowed Bitcoin network to maintain a stable block-generation rate of approximately one block every ten minutes. The formula is:

$$D_{new} = D_{current} \cdot \max \left(\min \left(\frac{1,209,600}{t}, 4 \right), 0.25 \right) \quad (1)$$

where:

- D_{new} is the difficulty of the next 2016 blocks.
- $D_{current}$ is the difficulty of the previous 2016 blocks.
- t represents the time difference in seconds between the timestamps of the latest block (B_n) and the 2016th previous block (B_{n-2015}).
- 1,209,600 seconds is the expected time to mine 2016 blocks at a rate of one block every ten minutes.
- The factor is constrained between 0.25 and 4 to prevent drastic changes.

As shown in Figs. 3–4, this DAA works very well for BTC. The block time distribution is converging to Poisson distribution and there is no autocorrelation between interblock times (i.e. correlation between the time series of interblock times and a delayed copy of itself). As explained in previous sections though, it is not viable for most altcoins and it does not solve the drift problem.

4.2 Emergency DAA (EDA)

In anticipation of the BTC–BCH fork, the BCH community decided that a new DAA was needed to ensure the survival of Bitcoin Cash. Although it was impossible to accurately predict the post-fork hash power of the new blockchain, it was certain that BTC would keep the lion’s share. There was also a great uncertainty about the effect of the split on BCH’s value [2]. Hence, emergency DAA was an adjustment to the original DAA that lowered difficulty immediately if there was a huge drop in hash power. More specifically, if the interval between the median time past (MTP) of the current block n and the MTP of block $n - 6$ was greater than 12 hours, the difficulty would drop directly by 20% [22].

4.3 CW-144

Bitcoin cash survived the split with the help of EDA DAA but it did not come away intact. As anticipated, the network’s hash power experienced a free fall

and left the blockchain vulnerable to fluctuations. Figs. 3–4 reveal the extent of hash rate fluctuation and the desperate need for a more stable DAA. Amaury Sechet’s CW-144 [5] was promising to provide just that. CW-144 was activated on November 2017, only three months after the activation of the EDA.

CW-144 adjusted difficulty after each block and used a 144-period simple moving average to estimate hash rate [22]. The formula used to calculate the new difficulty for each block is:

$$D_n = \widehat{H}_n \cdot T = \frac{\sum_{i=old}^{new} D_i}{T_{elapsed}} \cdot T \quad (2)$$

where:

- $\widehat{H}_n$ is the estimated hash rate after mining the n -th block.
- T is the ideal time between two blocks, i.e., 10 minutes.
- B_{new} is the block with the median timestamp between the latest block B_n and the previous two B_{n-1} and B_{n-2} .
- B_{old} is the block with median timestamp between blocks B_{n-144} , B_{n-145} and B_{n-146} .
- $T_{elapsed}$ is the elapsed time between B_{new} and B_{old} .

As shown in Fig. 4, CW-144 achieved an average interblock time close to optimal. However, it caused a serious autocorrelation issue. In simple words, CW-144 caused large and consistent oscillations in hash rate and difficulty throughout the day. This created brief periods of low difficulty and large numbers of new blocks, and large periods of high difficulty and low number of new blocks. As a result, not only did it not protect BCH from coin-hopping but it made it more vulnerable to it instead. On top of that, the BCH–BSV fork happened, causing even more stability issues to the remaining BCH network as can be seen in Fig. 3.

4.4 ASERT/NEFDA

By 2020 it was obvious that BCH’s DAA was suffering and was in urgent need of an upgrade. A few different solutions were put forward, and after a heated debate within BCH community, it was decided to adopt a proposal that used a negative exponential filter. Two very similar proposals were submitted independently by two separate teams under different names, ASERT [12, 19] and NEFDA [13].

NEFDA’s formula is:

$$D_n = \widehat{H}_n \cdot T = \frac{1}{S} \sum_{i=0}^{n-1} D_i e^{\frac{t_i - t_n}{S}} \cdot T \quad (3)$$

where:

- $\widehat{H}_n$ is the estimated hash rate after mining the n -th block.
- T is the ideal time between two blocks, i.e., 10 minutes.
- S is the smoothing factor.
- D_i is the difficulty after mining block i .
- t_i is the timestamp of block i .

Figs. 3–4 show that NEFDA (ASERT) helped stabilise the BCH hash rate, brought the block time distribution much closer to ideal, and solved the autocorrelation problem. Furthermore, due to hash rate stabilisation, it did not introduce any significant additional drift. However, it did not mitigate the problem either, and the blockchain is still exposed to further drift accumulation in the next leap in mining technology.

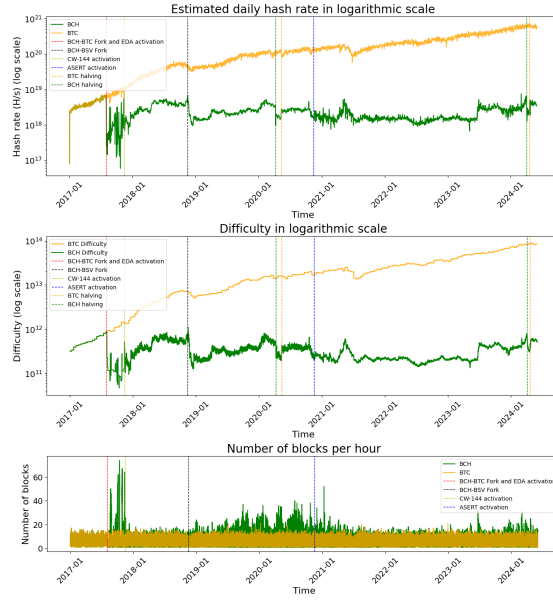


Fig. 3. BTC-BCH historical data comparison.

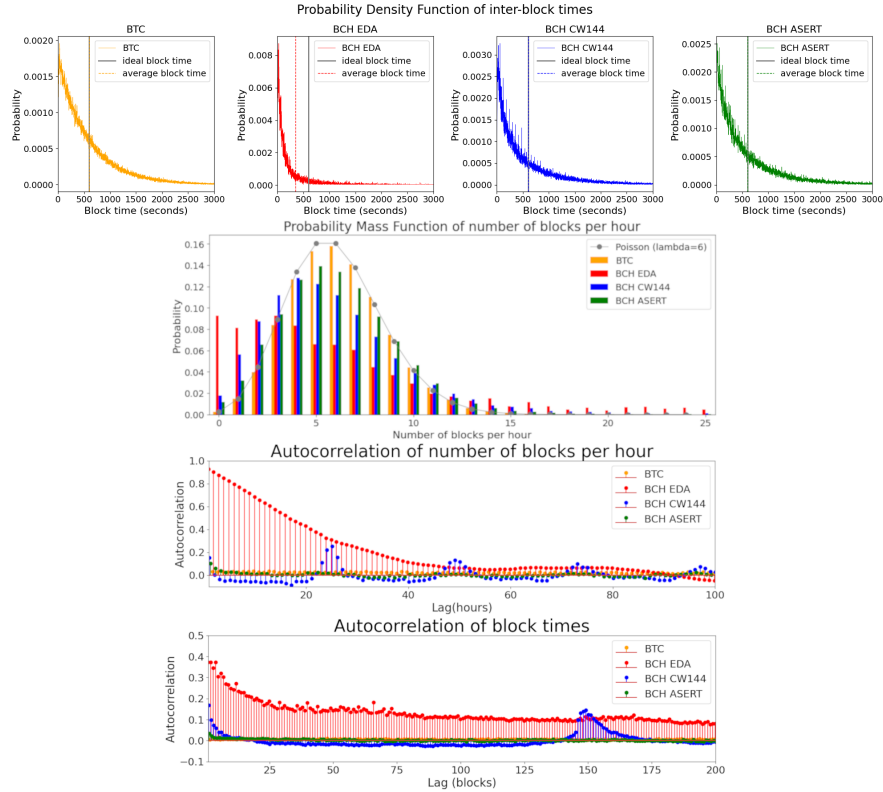


Fig. 4. DAA block time, and autocorrelation comparisons.

5 Modified NEFDA (NEF(DA)²)

To combat the drift problem, we propose a modified version of NEFDA which we call *NEFDA with Drift Adjustment* (NEF(DA)²). This novel algorithm has one critical adjustment over its predecessor: the ideal interval between blocks is no longer constant. Instead, the ideal interval is a variable that adapts to drift, i.e. long-term deviation from target interblock timing. NEF(DA)² formula is:

$$D_n = \widehat{H}_n \cdot T_n = \frac{1}{S} \left[\sum_{i=0}^{n-1} D_i e^{\frac{t_i - t_n}{S}} \right] \cdot T e^{\frac{t_0 - t_n + nT}{S}} \quad (4)$$

where:

- $\widehat{H}_n$ is the estimated hash rate after mining the n -th block.
- T_n is the goal time to mine the next block.
- T is the ideal time between two blocks, i.e., 10 minutes.
- S is the smoothing factor.
- D_i is the difficulty after mining block i .
- t_i is the timestamp of block i .

NEF(DA)² eliminates drift by continuously adjusting difficulty in response to the network's estimated hash rate as well as accumulated drift. Mathematically, the drift is given by:

$$drift = t_n - (t_0 + nT)$$

- t_0 being the Genesis time.

Starting from Equation (4) we can derive (see Appendix B) the following recursive formula:

$$D_n = D_{n-1} \left[1 + \frac{T}{S} \cdot e^{\frac{t_0 - t_{n-1} + (n-1)T}{S}} \right] \cdot e^{\frac{2(t_{n-1} - t_n) + T}{S}} \quad (5)$$

To highlight differences, we recall the equivalent NEFDA [13] formula:

$$D_n = D_{n-1} \left(1 + \frac{T}{S} \right) \cdot e^{\frac{t_{n-1} - t_n}{S}} \quad (\text{NEFDA})$$

To illustrate the effect of the drift correction term, let us assume that the most recent block was mined exactly on time and $t_n - t_{n-1} = T$. Therefore:

$$\frac{t_{n-1} - t_n}{S} = \frac{2(t_{n-1} - t_n) + T}{S}$$

indicating that when the latest interblock time is equal to the ideal, for both DAAs, the negative exponential filter term is:

$$e^{-\frac{T}{S}}$$

Using the assumption $1 + \frac{T}{S} \approx e^{\frac{T}{S}}$, the following holds for NEFDA:

$$D_n = D_{n-1} \cdot e^{\frac{T}{S}} \cdot e^{-\frac{T}{S}} = D_{n-1}$$

which means that in this case the difficulty will remain unchanged regardless of any accumulated drift. In contrast, NEF(DA)² yields:

$$D_n = D_{n-1} \left[1 + \frac{T}{S} \cdot e^{\frac{t_0 - t_{n-1} + (n-1)T}{S}} \right] \cdot e^{-\frac{T}{S}}$$

If $(n-1)T + t_0 - t_{n-1} = 0$, i.e. there is no drift, NEF(DA)² reduces to NEFDA. However, if $(n-1)T + t_0 - t_{n-1} \neq 0$, we encounter two possible scenarios:

1. **Ahead of Schedule:** In this case:

$$\begin{aligned} t_{n-1} - t_0 &\leq (n-1)T \quad \leftrightarrow \quad t_0 - t_{n-1} + (n-1)T \geq 0 \\ \implies e^{\frac{t_0 - t_{n-1} + (n-1)T}{S}} &\geq 1 \end{aligned}$$

Thus implying:

$$D_n \geq D_{n-1} \left(1 + \frac{T}{S}\right) \cdot e^{-\frac{T}{S}} = D_{n-1}$$

thereby increasing difficulty to counteract drift.

2. **Behind Schedule:** In this case, the reverse logic applies:

$$\begin{aligned} t_{n-1} - t_0 &\geq (n-1)T \quad \leftrightarrow \quad t_0 - t_{n-1} + (n-1)T \leq 0 \\ \implies e^{\frac{t_0 - t_{n-1} + (n-1)T}{S}} &\leq 1 \end{aligned}$$

which implies:

$$D_n \leq D_{n-1} \left(1 + \frac{T}{S}\right) \cdot e^{-\frac{T}{S}} = D_{n-1}$$

resulting in a decrease in difficulty to compensate for the lag.

Therefore, our proposed NEF(DA)² algorithm allows dynamic tuning of difficulty not only based on the estimated hash rate but also based on the accumulated historical drift. The main benefit of NEF(DA)² is its simplicity, as it achieves drift correction with a very subtle change in complexity compared to NEFDA (modifying only a couple of code lines). Consequently, it ensures very stable and predictable long-term block production. Meanwhile, it inherits all of NEFDA's desired properties and reduces to it as soon as drift is eliminated. Lastly, we acknowledge the necessity of temporarily overshooting difficulty to correct drift, which might not be popular with miners in the short term, but does put the block production schedule back on track.

6 Simulation

As a next step, we develop a simulation that replicates the block-production procedure and implements NEFDA and NEF(DA)² DAAs. Then we use the simulation data to extract information and create visualisations to compare the behaviour of the two algorithms. The code for simulation and visualisation can be found in an open source repository³.

For each simulation, we run the block-production procedure for a nominal duration of 15 years using the same hash rate function twice, one for each DAA. This produces a list of all the blocks that were created along with data (timestamps, difficulties, total work) and metadata (interblock times, drift, hash rate).

For the visualisation, we start by creating block time distribution graphs to ensure that NEF(DA)² follows the desired Poisson distribution similarly to NEFDA. Consequently, we juxtapose the autocorrelation of the algorithms to illustrate the lack of periodic pattern between interblock times. Lastly, we plot

³ <https://github.com/demKyr/BCH-DAA>

drift, difficulty and average block time per day against hash rate to compare the two DAAs.

The main purpose of the simulations is to exemplify the great impact of NEF(DA)² and to experimentally prove that it solves the drift problem while at the same time inherits all the beneficial properties of NEFDA. Fig. 5 demonstrates the code elegance and simplicity of NEF(DA)². Appendix C contains additional simulations.

```
#ifdef NEFDA
    candidate.target = 1.0/(estimated_hash_rate*IDEAL_INTERBLOCK_TIME);
#endif

#ifdef MODIFIED_NEFDA
    double skew = t-(GENESIS_TIME+(n+1)*IDEAL_INTERBLOCK_TIME);
    candidate.target = 1.0/(estimated_hash_rate*IDEAL_INTERBLOCK_TIME)*exp(skew/SHADOW_OF_FUTURE_TIMESCALE);
#endif
```

Fig. 5. Code changes from NEFDA to NEF(DA)².

6.1 Non-Adaptive Hash Rate

For this part, we use predetermined hash rate functions that do not adapt to changes in difficulty. Such simulations compare the response of the two DAAs to different hash rate behaviours like instant changes, linear growth, exponential growth or oscillations.

To avoid repetition, we only present the block time distribution and autocorrelation graphs for the first hash rate function. For this section, those graphs are used to ensure that both algorithms have similar behaviour in terms of block time distribution and autocorrelation.

Staircase hash rate function Hash rate is increased twice by one order of magnitude each time and replicates hash rate increases like the ones from the transition to GPU mining (2010) and then to ASIC mining (2013).

As shown in Figs. 6 and 7, both DAAs follow the same distribution and have similar autocorrelation. However, as shown in Fig. 8, NEF(DA)² can quickly absorb the shock from the sudden change in hash rate and eliminate any drift while NEFDA never recovers. Moreover, NEF(DA)² achieves this with only a slight ephemeral adjustment of difficulty.

Rectangular hash rate function This hash rate function creates a sudden increase of hash power by a factor of 30 and a subsequent drop to initial hash rate. As shown in Fig. 9, NEF(DA)² can effectively respond to both rises and drops although responding to the latter takes more time than responding to the former. This happens because the DAA uses past interblock times as empirical evidence to adjust difficulty. The number of blocks required to impact difficulty is approximately the same for sudden hash rate increases and decreases. However, when the hash rate drops, this number of blocks is produced much slower.

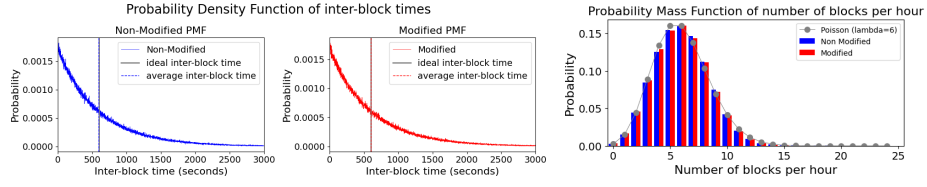


Fig. 6. Block time distributions for non-adaptive staircase hash function.

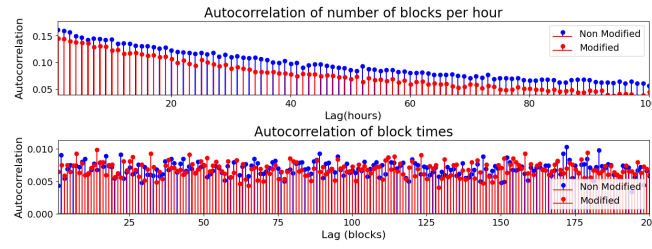


Fig. 7. Autocorrelations for non-adaptive staircase hash function.

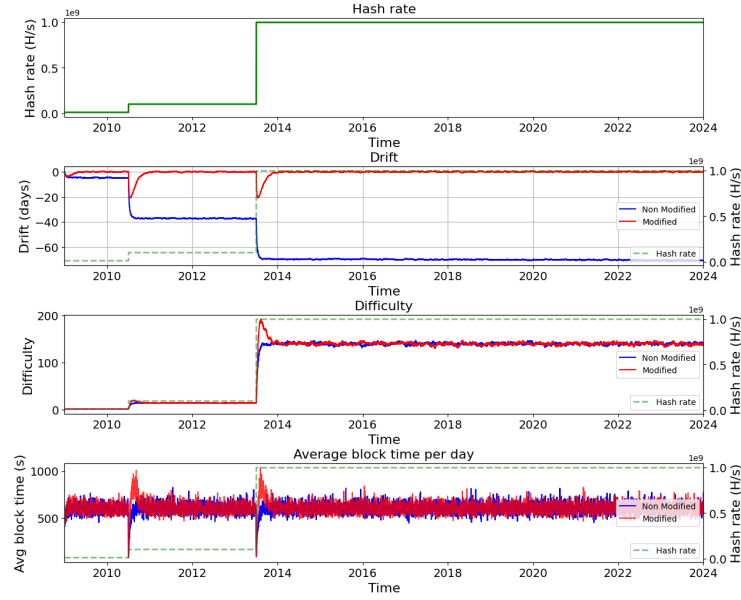


Fig. 8. Comparison for non-adaptive staircase hash rate function.

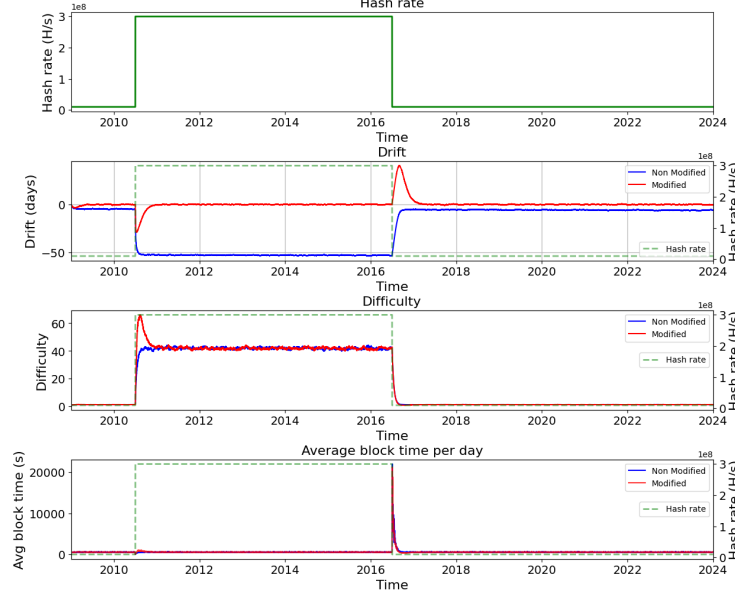


Fig. 9. Comparison for non-adaptive rectangular hash rate function.

6.2 Adaptive Hash Rate

In the real world, miners are sensitive to difficulty and changes in coin value. To simulate this behaviour, we developed an adaptive hash rate function. More specifically, we consider three different classes of miners: *loyal miners*, *variable miners*, and *greedy miners* with hash rates H_l , H_v , and H_g respectively. The total maximum available hash rate is controlled by a predetermined hash rate function but at any given moment, the network's current hash rate is given by:

$$H = H_l + H_v + H_g \quad (6)$$

$$H_v = \begin{cases} 4 \cdot H_l, & \text{if } R_D \leq 1 - \epsilon \\ \frac{4 \cdot H_l}{1 + e^{-\frac{1 - R_D}{\epsilon^2}}}, & \text{if } 1 - \epsilon < R_D \leq 1 + \epsilon \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

$$H_g = \begin{cases} 4 \cdot H_l, & \text{if } R_D \leq 1 - \epsilon/3 \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

where:

- R_D is the ratio of the average difficulty of the latest 6 blocks divided by the average difficulty of the 6 previous.

- H_t is following a predetermined function similar to the hash rate function of the previous section.
- $\epsilon = 0.05$

This hash function is much more representative of real mining behaviour and it provides important information about the similarities between the two algorithms in terms of block-time distribution and autocorrelation. It also complements the mathematical proofs, as many of the findings here cannot be directly derived from the equations of Section 5. As shown in Fig. 11, both DAAs show a large negative autocorrelation to the recent blocks and no autocorrelation to the earlier ones. This is expected as DAAs need to rapidly adjust to interblock times that deviate a lot from ideal but should not have any direct dependencies to any old interblock times.

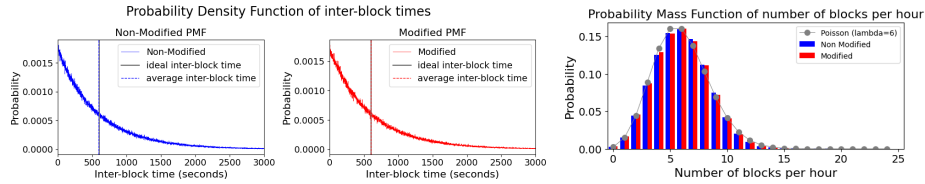


Fig. 10. Block time distributions for adaptive staircase hash function.

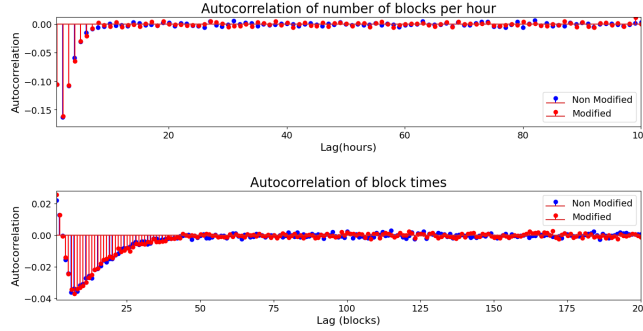


Fig. 11. Autocorrelations for adaptive staircase hash function.

6.3 Smoothing factor effect

The smoothing factor controls DAA's responsiveness. The larger it is, the longer it takes to eliminate drift, but at the same time, difficulty adjustment is smoother, and daily average interblock time does not deviate that far from the ideal of 10 minutes. For the simulations, we used a smoothing factor $S = 14$ days.

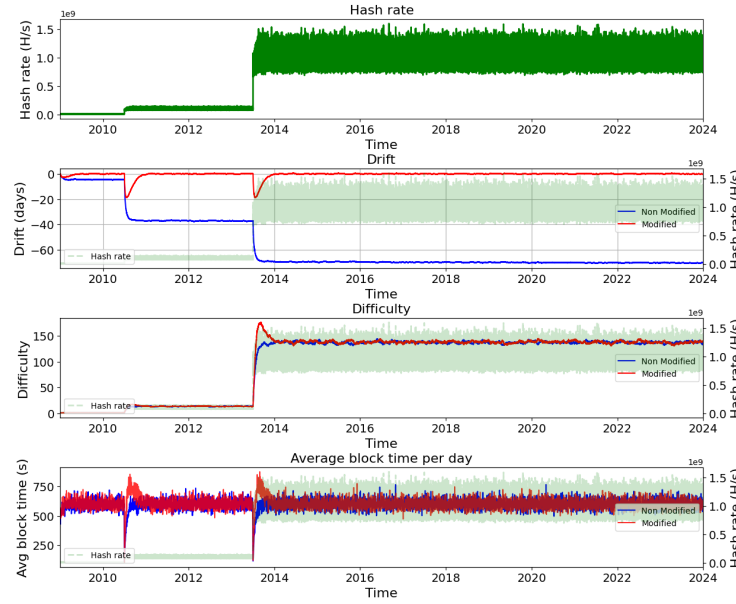


Fig. 12. Comparison for adaptive hash rate function.

7 Future Work

We are currently working on a generalised DAA that can potentially exterminate drift even for super-exponentially increasing hash rate functions. This DAA will introduce the meaning of drift-resistance level. NEFDA and NEF(DA)² can be categorised as special cases of the general DAA. Furthermore, core ideas introduced in this paper could be used to develop a similar BTC-compatible DAA.

8 Conclusion

Through this paper, we analysed the behaviour of present and past DAAs of BCH and revealed flaws in their designs. Moreover, we proposed NEF(DA)², a novel yet simple DAA that successfully eliminates drift while inheriting the desired properties of the existing BCH DAA. We believe that the incredible simplicity of the new algorithm and the way it can be seemingly added to the existing DAA are its main strengths. Accordingly, our algorithm, immaculately and essentially without increasing computational complexity, ensures long-term stability, dependability, and predictability in block-production rate. We have also supported our claims with mathematical proof and robust open-source simulations. Overall, we provided a complete and substantiated proposal of a Difficulty Adjustment Algorithm compatible with most Proof-Of-Work blockchains.

References

1. Amaury Sechet: Announcing the Grasberg DAA, <https://read.cash/@deadalnix/announcing-the-grasberg-daa-ff52e96d>
2. Antony Zegers: Bringing Stability to Bitcoin Cash Difficulty Adjustments, <https://mengerian.medium.com/bringing-stability-to-bitcoin-cash-difficulty-adjustments-eae8def0efa4>
3. Arena, L., Oriol, N., Veryzhenko, I.: Too fast, too furious? trading algorithmique et instabilité des marchés financiers. *Systèmes d'information & management* **Volume 23**, 81–106 (2018). <https://doi.org/10.3917/sim.182.0081>, <https://shs.cairn.info/revue-systemes-d-information-et-management-2018-2-page-81?lang=fr><https://shs.cairn.info/revue-systemes-d-information-et-management-2018-2-page-81?lang=fr>
4. Bashir, I.: Mastering blockchain : distributed ledger technology, decentralization, and smart contracts explained . Packt Publishing Ltd, Birmingham, 2nd edn. (2018)
5. Bitcoin ABC: Difficulty Adjustment Algorithm Update, <https://www.bitcoinabc.org/2017-11-01-DAA/>
6. Bitcoin Cash: Bitcoin Cash – Peer-to-Peer Electronic Cash, <https://bitcoincash.org/>
7. Bitcoin SV: The Original Bitcoin Blockchain: Bitcoin SV (BSV), <https://bitcoinsv.com/>
8. Buterin, V.: Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. (2014)
9. CoinDesk: Bitcoin Cash Price – BCH Price Index and Live Chart, <https://www.coindesk.com/price/bitcoin-cash/>
10. Ethereum: Proof-of-work (PoW), <https://ethereum.org/en/developers/docs/consensus-mechanisms/pow/>
11. freetrader: On Drift correction, <https://read.cash/@freetrader/on-drift-correction-39974f66>
12. freetrader, Jonathan Toomim, Calin Culianu, Mark Lundeborg: 2020-NOV-15 ASERT Difficulty Adjustment Algorithm (aserti3-2d), <https://upgradespecs.bitcoincashnode.org/2020-11-15-asert/>
13. Ilie, D.I., Werner, S.M., Stewart, I.D., Knottenbelt, W.J.: Unstable throughput: When the difficulty algorithm breaks. In: Proc. IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2021. Institute of Electrical and Electronics Engineers Inc. (5 2021). <https://doi.org/10.1109/ICBC51069.2021.9461086>
14. Jacob Wade: Hash Rate: How It Works and How to Measure, <https://www.investopedia.com/hash-rate-6746261>
15. Jonathan Toomim: Dark secrets of the Grasberg DAA, <https://read.cash/@jtoomim/dark-secrets-of-the-grasberg-daa-a9239fb6>
16. Király, T., Lomoschitz, L.: Protability of the coin-hopping strategy. Tech. rep., Egervary Research Group, Budapest (2018)
17. Kurt Wuckert Jr: History on Hash War, <https://coingeek.com/hash-war-history/>
18. Luke Conway: Bitcoin Halving: What It Is and Why It Matters for Crypto Investors, <https://www.investopedia.com/bitcoin-halving-4843769>
19. Lundeborg, M.B.: Static difficulty adjustments, with absolutely scheduled exponentially rising targets (DA-ASERT)-v.2 (2020), available at <https://www.bitcoinunlimited.info/nexa/da-asert.pdf>

20. Mark Lundeborg: Thoughts on Grasberg DAA, https://www.reddit.com/r/btc/comments/hx4kgz/thoughts_on_grasberg_daa/
21. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008), available at <https://www.bitcoin.org/bitcoin.pdf>
22. Reference Cash: Difficulty Adjustment Algorithm, <https://reference.cash/protocol/blockchain/proof-of-work/difficulty-adjustment-algorithm>
23. Reference Cash: Proof of Work, <https://reference.cash/protocol/blockchain/proof-of-work>
24. Siddhant Kejriwal: Bitcoin vs Bitcoin Cash: An In-Depth Look at BTC & BCH, <https://coinbureau.com/analysis/bitcoin-vs-bitcoin-cash/>
25. The Investopedia Team: What Is Bitcoin Cash (BCH), and How Does It Work?, <https://www.investopedia.com/terms/b/bitcoin-cash.asp>

A Comparison table of BTC/BCH halvings to the ideal

Table 1. Comparison of BTC/BCH halvings to the ideal

	Ideal	BTC	BCH
Genesis Block (block #0 - reward 50 BTC)			
Date	2009-01-09	2009-01-09	
Halving 1 (block #210000 - reward 25.0 BTC)			
Date	2013-01-06	2012-11-28	
Days since last halving (deviation)	1458 (0)	1419 (-39)	
Days since genesis (deviation)	1458 (0)	1419 (-39)	
Halving 2 (block #420000 - reward 12.5 BTC)			
Date	2017-01-03	2016-07-09	
Days since last halving (deviation)	1458 (0)	1319 (-139)	
Days since genesis (deviation)	2916 (0)	2738 (-178)	
Halving 3 (block #630000 - reward 6.25 BTC)			
Date	2020-12-31	2020-05-11	2020-04-08
Days since last halving (deviation)	1458 (0)	1402 (-56)	1369 (-89)
Days since genesis (deviation)	4374 (0)	4140 (-234)	4107 (-267)
Halving 4 (block #840000 - reward 3.125 BTC)			
Date	2024-12-28	2024-04-20	2024-04-03
Days since last halving (deviation)	1458 (0)	1440 (-18)	1456 (-2)
Days since genesis (deviation)	5832 (0)	5580 (-252)	5563 (-269)

B Analysis for NEF(DA)²

Recursive formula We have:

$$\hat{H}_n = \frac{1}{S} \left[\sum_{i=0}^{n-1} D_i e^{\frac{t_i - t_n}{S}} \right]$$

$$T_n = T \cdot \exp \left(\frac{t_0 - t_n + nT}{S} \right)$$

Thus:

$$D_n = \hat{H}_n \times T_n$$

$$= \frac{T}{S} \left[\sum_{i=0}^{n-1} D_i \exp \left(\frac{t_i - t_n}{S} \right) \right] \cdot \exp \left(\frac{t_0 - t_n + nT}{S} \right)$$

We make t_{n-1} appear in the sum, and the exponential factor:

$$= \frac{T}{S} \left[\sum_{i=0}^{n-1} D_i \exp \left(\frac{t_i - t_{n-1}}{S} \right) \right] \exp \left(\frac{t_{n-1} - t_n}{S} \right) \cdot \exp \left(\frac{t_0 - t_n + nT}{S} \right)$$

$$= \frac{T}{S} \left[\sum_{i=0}^{n-1} D_i \exp \left(\frac{t_i - t_{n-1}}{S} \right) \right] \cdot \exp \left(\frac{t_0 - t_{n-1} + (n-1)T}{S} \right) \exp \left(\frac{t_{n-1} - t_n}{S} \right) \exp \left(\frac{t_{n-1} - t_n + T}{S} \right)$$

We simplify:

$$= \frac{T}{S} \left[\sum_{i=0}^{n-1} D_i \exp \left(\frac{t_i - t_{n-1}}{S} \right) \right] \cdot \exp \left(\frac{t_0 - t_{n-1} + (n-1)T}{S} \right) \exp \left(\frac{2(t_{n-1} - t_n) + T}{S} \right)$$

We develop the sum

$$= \frac{T}{S} \left[\sum_{i=0}^{n-2} D_i \exp \left(\frac{t_i - t_{n-1}}{S} \right) + D_{n-1} \right] \cdot \exp \left(\frac{t_0 - t_{n-1} + (n-1)T}{S} \right) \exp \left(\frac{2(t_{n-1} - t_n) + T}{S} \right)$$

Thus we have

$$= \left[D_{n-1} + \frac{T}{S} D_{n-1} \cdot \exp \left(\frac{t_0 - t_{n-1} + (n-1)T}{S} \right) \right] \cdot \exp \left(\frac{2(t_{n-1} - t_n) + T}{S} \right)$$

Which gives us the final formula :

$$D_n = D_{n-1} \left[1 + \frac{T}{S} \cdot \exp \left(\frac{t_0 - t_{n-1} + (n-1)T}{S} \right) \right] \cdot \exp \left(\frac{2(t_{n-1} - t_n) + T}{S} \right)$$

Non recursive formula Let us with :

$$D_n = D_{n-1} \left[1 + \frac{T}{S} \cdot \exp \left(\frac{t_0 - t_{n-1} + (n-1)T}{S} \right) \right] \cdot \exp \left(\frac{2(t_{n-1} - t_n) + T}{S} \right)$$

Using the following hypothesis: $x = \frac{T}{S} \times \exp \left(\frac{t_0 - t_{n-1} + (n-1)T}{S} \right) \ll 1$, we have $1 + x \approx e^x$, thus giving us :

$$D_n = D_{n-1} \exp \left(\frac{T}{S} \exp \left(\frac{t_0 - t_{n-1} + (n-1)T}{S} \right) \right) \cdot \exp \left(\frac{2(t_{n-1} - t_n) + T}{S} \right)$$

$$D_n = D_{n-1} \exp \left(\frac{T}{S} \exp \left(\frac{t_0 - t_{n-1} + (n-1)T}{S} \right) + \frac{2(t_{n-1} - t_n) + T}{S} \right)$$

We finally have:

$$D_n = D_0 \exp \left[\sum_{i=0}^{n-1} \left(\frac{T}{S} \exp \left(\frac{t_0 - t_i + iT}{S} \right) + \frac{2(t_i - t_{i+1}) + T}{S} \right) \right]$$

$$D_n = D_0 \exp \left[\frac{T}{S} \left(\sum_{i=0}^{n-1} \exp \left(\frac{t_0 - t_i + iT}{S} \right) \right) + \frac{2(t_0 - t_n) + nT}{S} \right]$$

C Additional simulations

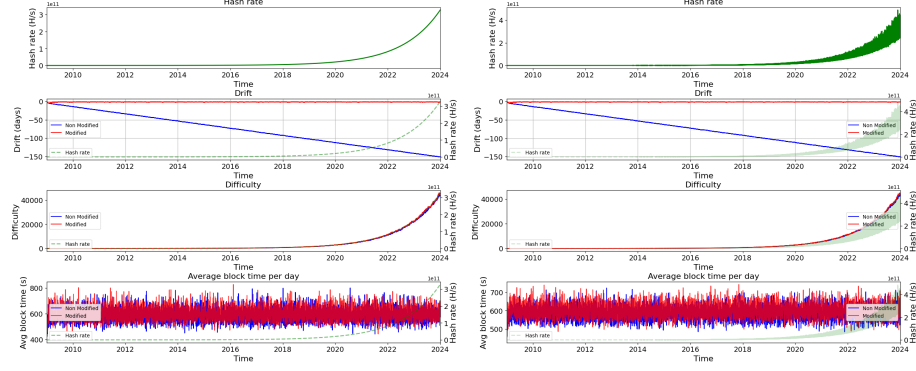


Fig. 13. Comparison for non-adaptive and adaptive exponential hash rate function.

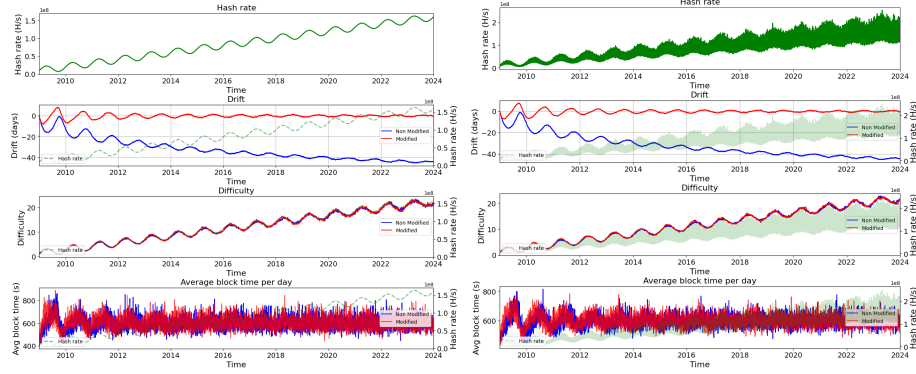


Fig. 14. Comparison for non-adaptive and adaptive sinusoidal and linearly increasing hash rate function.