arXiv:2405.15089v4 [cs.CR] 21 Aug 2025

Targeted Nakamoto: A Bitcoin Protocol to Balance Network Security and Carbon Emissions

Daniel Aronoff*

August 21, 2025

Abstract

In a Proof-of-Work blockchain like Bitcoin, mining hashrate increases with block rewards. Higher hashrate reduces vulnerability to attacks (lowering network security cost) but raises carbon emissions and electricity use (raising environmental cost), reflecting a tradeoff with a hashrate point or interval that minimizes total cost. Targeted Nakamoto is a protocol that incentivizes miners to home hashrate in on the minimum cost range by capping block rewards when above target and imposing a block reward floor when below. Monetary neutrality is maintained by proportional adjustments in spending potential among UTXO holders, offsetting the additions and subtractions to the block reward caused by the protocol. Bitcoin faces two conflicting existential risks: currently, high mining energy consumption invites political backlash; in the future, reductions in miner rewards will cause hashrate to decline, lowering the cost of an attack. Targeted Nakamoto balances these concerns by guiding hashrate towards a chosen target that places an effective floor on the cost of attack and a ceiling on carbon emissions.

***Keywords*—** Bitcoin, Blockchain, Mechanism Design, Network Security

Contents

*Massachusetts Institute of Technology, Cambridge MA, USA

I wish to thank Madars Virza for his aid and advice in developing the core ideas of this paper; Neha Narula for helping to refine and clarify the presentation; thoughtful comments from participants at the Chaincode Labs research seminar and Mark Serrahn for producing the figures herein. Kristain Praizner wrote the code for the hashrate control algorithm and implemented it on an API that is a companion to this paper.

1 Introduction

Proof-of -Work (PoW) blockchain cryptocurrencies like Bitcoin require the application of computing power by miners to operate the network. Miners assemble blocks and compete to solve a puzzle set by the code. The number of puzzle guesses (one hash per guess) a mining computer makes in a specified interval of time is its hashrate, which consumes electricity. Miners pay the direct cost of electricity. There are, in addition, two externalities created by mining activity. One externality is the vulnerability of the blockchain network to attack. The degree of vulnerability is related to the cost of forking the blockchain, which in turn is related to the cost of hashrate and the volume of hashrate employed by miners. The higher is hashrate, the higher is the cost of forking a PoW blockchain, which implies that network security is increasing in hashrate (equivalently, security cost is decreasing in hashrate). The other externality arises from the electricity consumed by hashrate. Miner demand for electricity increases both the price and aggregate consumption of electricity. The former is a cost imposed on competing users of electricity. The latter generates carbon emissions, which has a social cost. These electricity related costs are increasing in hashrate. The interaction of security risk and carbon emissions imply there is a tradeoff in total cost, defined as the sum of the two externalities costs, at different levels of hashrate. An increase in hashrate reduces the vulnerability of the network (equivalently, security cost is decreased), while it increases carbon emissions and creates upward pressure on the price of electricity. There may be a point or interval of hashrate in which total cost is minimized. Current PoW protocols, which I refer to as "Nakamoto", do not restrict hashrate.[1]

The key lever upon which the new protocol is built is the incentive effect of the block reward. An increase in the USD value of the block reward creates an incentive for miners to apply more hashrate (assuming the dollar is the currency in which miners calculate profit). A decrease in the USD value of the block reward has the opposite effect. Equilibrium hashrate can range from zero to unbounded. Figure 1 shows the tight correlation over time between percentage change in the USD value of the Bitcoin block reward and the percentage change in energy consumption from Bitcoin mining.

Targeted Nakamoto (hereafter "*Targeted*") modifies Nakamoto to create an incentive for miners to push hashrate toward an interval whenever it strays outside the interval. The miner's minimum block reward (in BTC) is increased when hashrate is below the interval, which induces miners to apply more hashrate. The miner's maximum block reward (in BTC) is decreased when hashrate is above the interval, which induces miners to apply less hashrate. Monetary neutrality is maintained by offsetting the increase and decrease in the miner's block reward by opposite adjustments to the spending potential of addresses with UTXO value, in proportion to the value held by each address.

[1] Our focus in this paper is Bitcoin, but the protocol applies to any PoW cryptocurrency.

[3] `https://ccaf.io/cbnsi/cbeci`

[3] `https://coinmetrics.io`

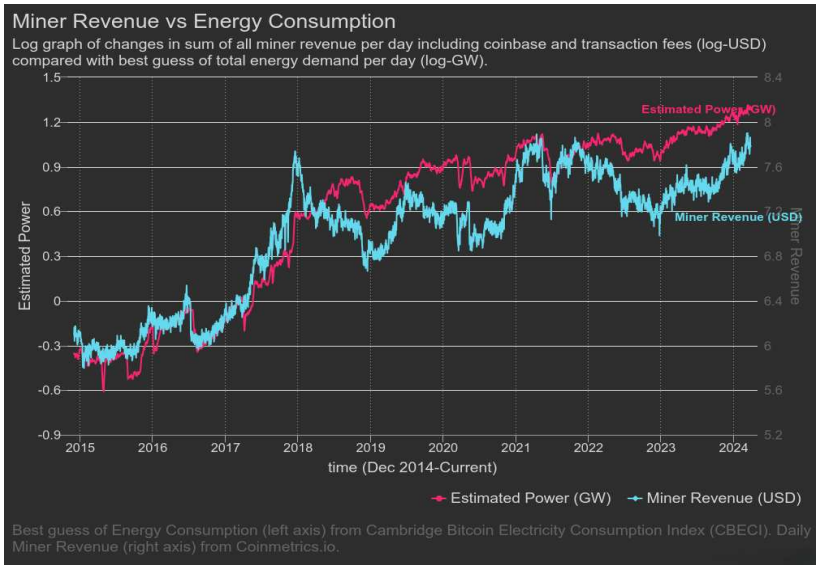

Figure 1: Bitcoin Energy Consumption vs USD Mining Rewards. Source: CBECI[2]and Coinmetrics.[3]

1.1 Balancing Competing Risks

Targeted balances the ecological threat from electricity consumed by hashrate against the threat to network security from low hashrate. The former is a subject of current political controversy, while the latter will materialize in the future when the USD value of the block reward declines as a result of the halving of the coinbase. [4] Transactors will have little incentive to compensate by increasing fees, which are paid to obtain a priority in the qeue of mempool transactions. The decline in block reward will elicit a decline in hashrate (Equation 4). *Targeted* compensates by increasing the block reward, and does it without increasing overall UTXO spending potential.

1.2 Links to code and web API (by Kristian Praizner)

Attached are links to the code-base for controlling Hashrate and a Web API with an empirically estimated hashrate function and a module that enables the user to choose a target hashrate interval and control parameters and to run counterfactual experiments with alternative time-paths of model variables.

The code-base is here `https://github.com/Krisp140/TargetedNakomoto`

[4]Historically, the trend USD/BTC exchange rate has risen sufficiently to offset the halving of the coinbase. However, this cannot continue indefinitely for two reasons. First, coinbase will eventually be zero. Second, maintaining a fixed USD value of the coinbase paid to miners as halving continues would require the inflation adjusted USD to BTC exchange rate to double every four years. This implies a growth rate that exceeds US (and world) GDP by orders of magnitude, which is dynamically unsustainable (Tirole [21]).

The web API is here `https://targetednakamoto.com/About`.

1.3 Related Literature

This paper builds on the contributions of Nakamoto [17] who laid the foundations for the PoW protocol used in Bitcoin.

There are two previous works with which aspects of this study closely align. Pagnotta [19] identified and modeled the complimentarity between mining hashrate and Bitcoin USD price. The interdependence runs through network security. Hashrate determines security, security affects traders valuation of Bitcoin, which affects it price, price affects the USD value of the mining reward, which determines equilibrium hashrate. *Targeted* injects into this causal chain a regulator that adjusts the miner's block reward to incentivize miners to keep hashrate (and thereby security) in the neighborhood of a selected target. Mirkin et.al. [15] show that electricity cost can be reduced, for any level of network security, by adding a "proof-of-delay" (e.g. using a verifiable delay function) threshold for a miner's block to be eligible to append to the chain. The mandatory delay interval creates an incentive for miners to shift expenditure from hashrate to investment in mining equipment. The re-allocation of expenditure from variable to fixed cost reduces the carbon footprint. Notably, proof-of-delay does not alter equilibrium expenditure on mining, which is determined by the miner's block reward. This means it does not affect the cost to attack the network and thereby network security. While proof-of-delay reduces energy consumption for a given level of security, *Targeted*, which is compatible with proof-of-delay, enables network security to be selected and maintained.

One of the externalities costs addressed in this paper are the carbon emissions generated by the electricity consumed in mining Bitcoin. I rely on two data providers. One is the Bitcoin Electricity Consumption Index (BECI) of the Cambridge U.K. Centre for Alternative Finance [8], which is regarded as a credible - though not infallible - source for information on Bitcoin energy consumption. The other is Coinmeterics [5](which is a source for BECI) for other Bitcoin performance indicators. Coinmetrics is considered to be a careful aggregator of data on cryptocurrencies. Sedlmeir et.al. [20] and Gallersdorfer et.al. [10] review comparative magnitudes of energy consumption for different blockchain protocols. Gallersdorfer [10] compares the estimation of Bitcoin electricity consumption across several sources. For my purposes, the salient finding is that these studies document a time-path of energy consumption that is tightly correlated with the time-path of hashrate in Bitcoin. [6]

As with several previous studies of blockchain protocol, this paper uses mechanism design approach to the study of the consensus protocol. Chen et.al. [14] surveys various aspects of existing PoW blockchain protocols, including the incentive to increase energy consumption in mining, from a mechanism design perspective. Cong et.al [6] analyses the effect of PoW mining competition on energy consumption. Leshno

[5] `https://coinmetrics.io`

[6] See Figure 1 in Gellersdorfer [10].

and Strack [13] identifies features of the miner payoff structure that are required to maintain blockchain stability. They prove that certain of the core properties of a decentralized PoW blockchain - namely free entry of anonymous profit driven miners - are only incentive compatible with a reward mechanism that is similar to Nakamoto. *Targeted* preserves those features.

In their study of the Bitcoin payment system, Huberman et.al. [12] characterize an equilibrium where transactions are appended to blocks in descending order of the size of fee offered by transaction senders, which reflects transactor's relative willingness to pay for a position in the queue for placement on the blockchain. The adjustment to the miner's block reward in *Targeted* is guided by this insight. When the block reward is adjusted, the ordering of the rewards received by the miner from transactions always matches the order of fees offered by transaction senders. This, in turn, leaves unaltered he order in which transactions are appended to blocks.

1.4 The research question and design constraints

Targeted augments Nakamoto by creating an incentive for miners to home in on a hashrate interval. The augmentation comes close to adhering to formal constraints which reflect core values held by the Bitcoin community.[7] The constraints are partitioned into the consensus protocol layer (C constraints) and the transaction protocol layer (**T**- constraints).

C constraints

- No change to proof-of-work mining
- No change to growth rate target and puzzle difficulty adjustment policy
- No change in block size
- No change to the monetary policy (aggregate UTXO value unaffected)
- No required oracle
- No hard forks

T constraints

- No interference with transactions between nodes
- No change to the order in which transactions are appended to the blockchain

[7]What constitutes a core feature is subjective. I developed the list herein after consultation with a number of participants in the Bitcoin network, including developers, miners and exchange operators.

It will be shown that *Targeted* satisfies each of these constraints with one exception. *Targeted* can instruct miners to burn UTXO's from transaction outputs, which may require a hard-fork to implement.

1.5 Roadmap

The rest of the paper is organized as follows. Section 2 provides an overview of the two types of cost, the sum of which *Targeted* is designed to minimize; environmental costs that increase with hashrate and network sercurity costs that decrease with hashrate. Section 3 casts Nakamoto and *Targeted* in a mechanism design framework in order to articulate the key features of each protocol and their interaction and to show how *Targeted* augments Nakamoto. Section 4 shows how the ratio of mining puzzle difficulty to blockchain growth rate is a sufficient statistic for hashrate and then derives the expression for equilibrium hashrate as a function of, inter alia, the value of the block reward. Section 5 uses the relationship between the puzzle difficulty/growth rate ratio and hashrate, along with hashrate equilibrium to construct the policy for adjusting the miner's block reward to incentivize miners to push hashrate toward the target hashrate interval. Section 6 demonstrates that the block reward policy is consistent with Nash equilibrium at the targeted hashrate and demonstrates that the hashrate control algorithm dynamically pushes hashrate toward the target interval. Section 7 states a protocol that renders the aggregate money supply and the distribution of spending potential unaffected by the Hashrate Control Algorithm. The paper concludes with a summary of the key properties of *Targeted*.

2 Hashrate Externalities

There are intensive and extensive margins related to hashrate in Bitcoin mining. The intensive margin is the equilibrium hashrate cost paid by miners. The extensive margins are the social cost of CO^2 emissions from the electricity used to generate hashrate and the network security risks arising from the cost of an attack that forks the blockchain. The former is increasing and the latter is decreasing in aggregate miner hashrate. The causal link from hashrate to these variables is determined by ASIC efficiency and electricity cost. Equation 1 displays the relationship.

$$\underbrace{cost/N}_{\text{per-hash cost}} = \underbrace{(cost/KwH)}_{\text{electricity cost}} \div \underbrace{(N/KwH)}_{\text{ASIC efficiency}} \tag{1}$$

Where N is hashrate (defined in Section 4), KwH is kilowatt hours. Environmental cost is an increasing function of hashrate as follows.

$$\text{environmental cost} \underbrace{\longleftarrow}_{\text{external function}} CO^2 \underbrace{\longleftarrow}_{\text{physical law}} KwH = \boldsymbol{N} \div \text{ASIC efficiency}$$

Network security cost is a decreasing function of hashrate as follows.

$$\text{network security cost} \underbrace{\longleftarrow}_{\text{external function}} \text{mining cost} = \boldsymbol{N} \times \text{electricity cost} \div \text{ASIC efficiency}$$

Figure 2 displays the relationship between environmental cost and network security cost on the vertical axes, and hashrate on the horizontal axis. is a upward sloped curve and network security is a downward sloped curve. We define optimal hashrate interval as the point, or interval, where the sum of the costs are minimized. The goal of *Targeted* is to incentivize miners to home in on this interval.

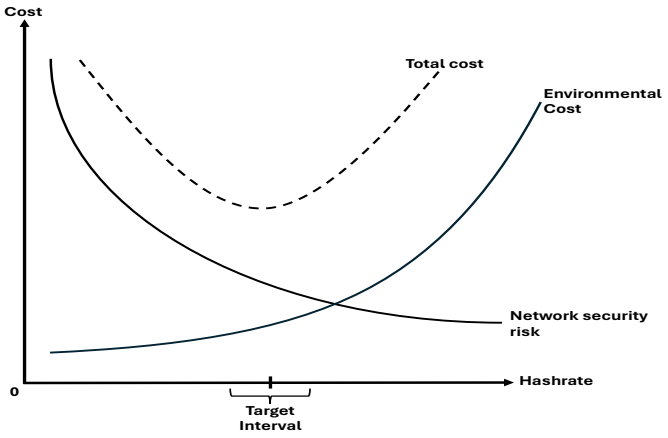

Figure 2: Target Hashrate Interval

2.1 Dependencies

Assuming the external functions are stable over time, environmental risk and network security are conditional on ASIC efficiency and electricity cost.[8] An increase in ASIC efficiency will shift both curves rightward and increase the optimal hashrate target. An increase in electricity cost will shift the security curve rightward while leaving the environmental risk curve unaffected, and reduce the optimal hashrate. ASIC efficiency and electricity cost can be estimated and oracles can be implemented to adjust the hashrate target. Additionally, there is a causal channel running from hashrate to electricity cost, whereby the increased demand for electricity places upward pressure on the price. Fowlie et.al. [9] estimate a price elasticity of demand for electricity in the U.S. of -2.06 with a range of alternative elasticities around -1, which implies that an increase in electricity demand will be split between increases in both price and quantity.

Less clear is how to measure the external functions. There is a literature on measuring the social cost of CO^2 emissions. Nordhaus [18]), Deitz et.al. and Stern [7] are important contributions and Chen et.al. [14]

[8]The function connecting electricity to CO^2 is a physical law. The function connecting forking cost to attack incentives is subject to variables that lie outside our analysis, such as, inter alia, interest rates and the legal environment.

model the flow of carbon emissions from multiple energy sources. Nevertheless, the shape and magnitude of the relationship between CO^2 emissions and social cost is a matter of ongoing debate. There has not been any substantive work, to our knowledge, connecting mining cost to quantifiable network security.[9] A market in Bitcoin security risk, with payoffs tied to attack events, could resolve the matter. But for now, this function can only be guessed at. The practical implementation of *Targeted* will require the formation of a consensus among Bitcoin network participants on the external functions. This may require negotiations that will involve compromise over differing opinions and economic interests.

2.2 Network Security

To elucidate the relationship between hashrate cost and network security, we review three vectors of attack on a PoW blockchain that decrease in profitability as hashrate cost increases.

Double-spend attack A double-spend attack requires the attacker to build an alternative chain that overtakes the incumbent canonical chain at in terms of cumulative mining puzzle difficulty. An attacker with over 50% of total hashrate will overtake the incumbent canonical chain at some block with probability measure 1. At less than 50% share, the attacker's probability of success is less than half.[10] The amount of attacker hashrate required to successfully fork the blockchain, and its cost, is increasing in the level of honest miner hashrate. Budish [3] and Aronoff et.al. [1] point out that the cost of a successful attack is at least partially offset by the block rewards that accrue to the attacker. The possibility that a victim of a double-spend attack retaliates by mining on the incumbent canonical chain leads to a war of attrition between the attacker and the victim (Moroz et.al. [16]). Aronoff et.al. [1] show that the outcome of a war of attrition depends on equilibrium refinements and exigent circumstances, even when the costs between players differ. Nevertheless, an increase in the cost of launching the attack will act as a deterrent by increasing the sunk cost that the attacker loses if it is unsuccessful. Equation 2 is the attacker's un-discountedex-ante expected profit when block rewards equal hashrate cost. The key observation is that, for any probability of success, the attacker's expected profit is declining in honest miner hashrate.

$$\begin{aligned}\text{Ex-ante Expected Profit } = \; & Prob(success)V - \\ & \{1 - Prob(success)\}\text{E[hashrate cost per block]E[number of blocks]} \quad (2)\end{aligned}$$

Where V is the double-spend value.

Eclipse attack An eclipse attack involves isolating a node from the network and constructing a chain, forked from the incumbent canonical chain, that is broadcast only to the victim (the "false chain"). On the false

[9] Appendix 2.2 discusses the relationship between mining cost and network attack vectors.

[10] See Gervais et.al. [11] for simulations of attacks using different ratios of attacker hashrate to miner hashrate. The frequency of success is increasing in attacker hashrate and the average number of blocks required to overtake the incumbent canonical chain is decreasing in attacker hashrate.

chain the attacker sends the victim UTXO value, which the victim believes to be on the canonical chain. After the attacker receives the exchange item from the victim, it discontinues applying hashrate to the false chain. The attacker incurs two elements of cost. One element is the hashrate the attacker applies to the false chain. The other element is the cost of isolating the victim from the network. Neither of these costs necessarily increase with honest miner hashrate. However, the attacker may need to apply hashrate that is proximate to honest miner hashrate to convince the victim into believing that the fake chain is canonical. Therefore it is possible that the cost of an eclipse attack is increasing in honest miner hashrate, in which event the profit is declining in honest miner hashrate.

Shorting attack A shorting attack has two parts; (i) placing a bet that a cryptocurrency will decline in value relative to some other currency or asset and (ii) gaining control of the cryptocurrency network in order to cause a disruption that will induce a loss in confidence and fall in exchange value. One type of bet is a forward sale of, say, Bitcoin in dollars. The attacker contracts to sell BTC to a counterparty at a future date for a dollar strike price per unit of BTC. The attacker anticipates that its disruption to Bitcoin will enable it to purchase BTC at a dollar price that is below the strike price, enabling it to resell to the counterparty at a profit. The size of the bet is independent of hashrate. In a liquid futures market an attacker could contract to purchase $x dollars worth of BTC at some future price regardless of the level of hashrate. On the other hand, the cost of gaining control of the network scales with hashrate. Thus, the profitability of the attack is declining in hashrate.[11]

Finally, Carlesten et.al. [4] argued that as the coinbase approaches zero and miners become dependent on transaction fees, the decline and variability of the fees will incentivize selfish mining, which could disrupt the liveness of the blockchain and possibly push down hashrate (and thereby mining cost).

3 Targeted Nakamoto - A Mechanism Design Perspective

This section describes the elements of *Targeted* and compares the mechanism designs of *Targeted* and Nakamoto.

3.1 Key Building Blocks of Targeted Nakamoto

Targeted controls hashrate by leveraging four features that are intrinsic to the blockchain. One feature is informational. The hashrate, which is not recorded on the blockchain, can be estimated from mining puzzle difficulty and block timestamps, which are recorded on the blockchain. The augmented protocol targets bounds on hashrate, for which the ratio of puzzle difficulty to blockchain growth rate is a sufficient statistic.

The second feature is an object that can be modified. The Nakamoto block reward is comprised of two parts;

[11] The value of the block rewards earned by the attacker decline with the Bitcoin exchange value.

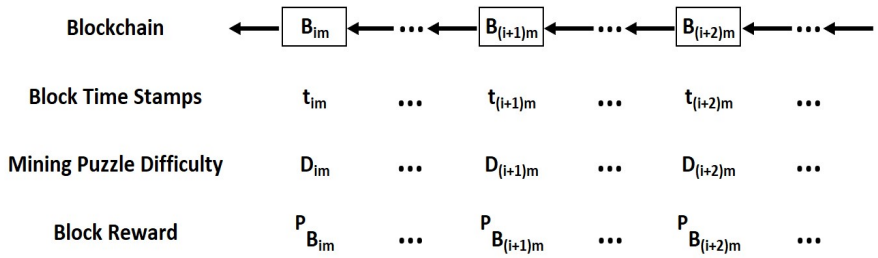

Figure 3: Blockchain Records

an algorithmic minting of new tokens and a discretionary fee added to a transaction. Both parts of the block reward are recorded in the blockchain and can be altered by the code. *Targeted* places a floor or ceiling on the the miner's block reward when hashrate is above or below the target interval.

The third feature is a market incentive. The hashrate applied by miners is affected by the block reward; it increases as the value of the block reward increases and decreases as the value of the block reward decreases. This implies that hashrate can be controlled by manipulation of the block reward.

The fourth feature enforces monetary neutrality by adjusting the aggregate UTXO spending potential to exactly offset the amount added to or subtracted from the total block reward to obtain the miner's block reward.

3.2 An overview of the mechanism design of the protocols

Targeted can be decomposed into two protocols that interact while operating separately. The Nakamoto part is the mechanism associated with the current protocol. The *Targeted* part adds a mechanism to achieve a target hashrate interval. Figure 3 displays the information recorded on the blockchain that is used by each protocol[12].

Table 1 shows the basic elements of Nakamoto and *Targeted*. For each protocol, the information contained in the signal, the instrument and the feedback are recorded on the blockchain. There is a salient difference in the target. The target for Nakamoto, blockchain growth rate, is recorded on the blockchain. The target for *Targeted*, hashrate, is not recorded on the blockchain.

[12]The network also records the addresses of UTXO'a at each block. Neither Nakamoto or *Targeted* use that information.

Protocol Mechanism Design		
Element	Nakamoto	*Targeted*
Target	Blockchain Growth Rate	Hashrate Target Interval
Signal	Block Timestamps	Puzzle Difficulty/Growth Rate
Feedback	Block Timestamps	Puzzle Difficulty/Growth Rate
Instrument	Puzzle Difficulty	Block Reward
Policy	Adjust to Growth Rate	Ceiling or Floor on Block Reward

Table 1: A Comparison of Mechanisms

4 The Model

This section derives and characterizes the equilibrium level of hashrate. Section 4.1 derives the function of puzzle difficulty that is a sufficient statistic for hashrate. Section 4.2 characterizes the equilibrium hashrate as a function of, inter alia, the miner's block reward.

4.1 The puzzle difficulty signal of hashrate

Nakamoto sets a target time for intervals between the addition of new blocks to the blockchain. The timestamps are read by the code at intervals of m blocks, which is an epoch. An epoch in Bitcoin $m = 1,260$ blocks. The instrument used to achieve the target is an adjustment to the difficulty of the mining puzzle, denoted as D. The mining puzzle is the solved when the miner's input into a hash function outputs a value that falls within the target range set by the code. Puzzle difficulty is the ratio of the range set by the code to the range of all possible output values. The probability of guessing the solution is $p = 1/D$ and each guess is independent.[13] D is adjusted after the mth (final) block of a epoch by dividing the number of blocks into the elapsed time from the previous timestamp. When the blockchain growth rate exceeds the target, puzzle difficulty is increased. When the blockchain growth rate falls below target, puzzle difficulty is decreased.

A single mining computer can guess a puzzle solution sequentially in time.[14] I model mining computers as making successive guesses synchronously at times $t \in \{t_1, t_2, ...\}$, each separated by an interval of temporal length $t_{i+1} - t_i$ which I normalize to 1 unit of time[15]. The Nakamoto blockchain growth target is one block

[13]By construction p is a monotone decreasing function $P(D)$ of D. The derivative $\frac{dp}{dD} < 0$ in both the simplified expression in the text as well as e.g. the algorithm used in Bitcoin Core. The derivative is the only property material to our analysis.

[14]A computer with n parallel processors is treated here as n separate computers.

[15]The synchronicity enables the expression of the time-path of puzzle solutions compactly as a Bernoulli Process, which simplifies the analysis WLOG.

every T^* units of time. During this time interval each computer has guessed (or hashed) T^* proposed solutions. I define a unit of hashrate as a single mining computer making T^* puzzle guesses. If there are N mining computers making guesses, the number of total guesses made in the time interval of the blockchain growth target is NT^* and the hashrate is N.[16] The framework for guessing puzzle solutions over time is a Bernoulli Process. Let T denote the time the first puzzle solution is broadcast to the network. For a hashrate of N the expected time to reach the solution is expressed by Equation 3.

$$\underbrace{\mathbb{E}[T]}_{\text{expected time between blocks}} = D/N \tag{3}$$[17]

Equation 3 shows that, in order to achieve the growth rate target of T^* in expectation, puzzle difficulty D must be adjusted to offset hashrate N. Since T for the prior epoch is recorded on the blockchain and D is set by the code, the formula can be inverted to estimate hashrate as a function of the growth rate and puzzle difficulty, $\hat{N} = D/T$.[18] $\hat{N}$ is unbiased since the observed T is an unbiased estimate of the mean time interval implied by the Bernoulli Process. It is also a sufficient statistic for hashrate since D and T contain all of the information on N that is encoded on the blockchain.

4.2 Mining equilibrium

The miner who is first to broadcast a puzzle solution receives a block reward of cryptocurrency. Miners incur cost in terms of the local official currency, which we denote USD.[19] Mining equilibrium is a state where miners choose hashrate to maximize expected profit. When miners adjust hashrate at short block intervals profit maximization implies that the profit from a marginal unit of hashrate equals the competitive rate of profit. The independence of puzzle guesses means each unit of hashrate has a $1/N$ probability of receiving the block reward. Equation 4 is the equilibrium profit from mining one unit of hashrate (i.e. operating one mining computer for T^* units of time).[20]

$$\pi = \frac{1}{N} e P_{B_n}^M (1 + \Delta_T) - c = \phi \tag{4}$$

[16] This accords with the conventional definition of hashrate used e.g. by Coinmetrics [5], as puzzle guesses per second.

[17] The formula for the expected time of first solution is: $\mathbb{E}$[Number of guesses] = 1/Prob, where Pr is the probability of solving the puzzle. In our case, $Prob = 1/D$ and Number of guesses = NT. When N is a fixed number, the Bernoulli Process implies that $N\mathbb{E}[T] = 1/Pr = D$. This yields the expression in Equation 3

[18] $\hat{N} = N + \epsilon$, where ϵ is a mean zero iid random variable.

[19] Profit can be expressed in terms of cryptocurrency. In that case the unit hashrate cost cost c is multiplied by $1/e$, which is the USD to BTC exchange rate. The equilibrium hashrate is the same whatever currency is chosen to be the unit of account.

[20] Equation 4 is a dynamic equilibrium in the sense that it includes a growth rate that deviates from target. A rigorous expression would show Δ_T as a function of puzzle difficulty and hashrate.

$\Delta_T = T^* - T$ is the difference between the target growth rate, T^*, and the expected actual growth rate T.[21] $(1 + \Delta_T)$ is the number of blocks per T^* units of time; c is the marginal cost of a unit of hashrate (i.e. one computer for T^* units of time);[22] e is the USD/BTC exchange rate and $p^M_{B_n}$ is the expected miner's block reward on the n^{th} block. (p_{B_n} is the total block reward). I make no assumption about the structure of the mining market. $\phi \geq 0$ a competition parameter. Under free entry in the mining market $\phi = 0$. Under less competitive conditions $\phi > 0$. Equation 5 is the mining market equilibrium.

$$N = eP^M_{B_n}(1 + \Delta_T)/(c + \phi) \tag{5}$$

Equation 5 shows that hashrate N is a function of the market conditions faced by miners. Puzzle difficulty does not appear in Equation 5. Equation 6 substitutes D/T for N in Equation 5.

$$D/T = eP^M_{B_n}(1 + \Delta_T)/(c + \phi) - \epsilon \text{[23]} \tag{6}$$

Equation 6 displays the monotone relationship between the miner's block reward, p^M_B, and the sufficient statistic for hashrate, D/T.

5 The *Targeted* Block Reward Policy

This Section explains the implementation of the policy of adjusting the miner's hashrate. Section 5.1 shows how the allocation of an addition or subtraction to the miner's block reward leaves unchanged the miner's incentive to append transactions onto blocks according to the ranking of the size of fee offered by the transactor. Section 5.2 builds on the analysis of Section 4 to state the policy of adjusting the miner's block reward in response to the ratio of puzzle difficulty to blockchain growth rate. Section 5.3 states the algorithm for controlling the miner's block reward.

5.1 The block reward adjustment allocations

One of the **T** constraints is that $Targeted$ should not change the order in which transactions are appended to blocks. Huberman et.al. [12] demonstrate that Nakamoto induces an equilibrium ordering of transaction fees and transactions on the blockchain that reflects transactors' relative time preferences. Transactions are

[21] Blockchain growth rate is expressed as an expectation since it is stochastic.

[22] Marginal cost is the cost of electricity to run the ASIC's. Fixed costs, such as the rent on the building where the computers are stored, or the sunk cost of purchasing the computers, are not included.

[23] ϵ is the stochastic noise from the relation $\hat{N} = N + \epsilon$.

appended to blocks in descending order of the size fee offered and the ranking of fees match the ranking of transactors' time-preference for executing their transaction.[24] It is a constrained efficient outcome in the sense that if agents A and B send their respective transactions to the mempool at the same time and agent A incurs a higher cost for delay in completing its transaction than agent B, agent A's transaction will be appended to the blockchain ahead of agent B's transaction. *Targeted* preserves the same ordering of transactions by adjusting the coinbase and each transaction fee by the same percentage. Fee_{f,B_n} denotes the fee paid by the fth transaction appended to block B_n. Denoting $\xi = (p^M_{B_n} - p_{B_n})/p_{B_n}$, the ordering $Fee_{i,B_n} > Fee_{j,B_n}$ implies the ordering $\xi Fee_{i,B_n} > \xi Fee_{j,B_n}$, i.e. the ordering of transaction fees sent to the miner is unchanged. R_n denotes the coinbase portion of the block reward. The miner's block reward is $p^M_{B_n} = (1+\xi)\big[\sum_f Fee_{f,B_n} + R_n\big]$. When $\xi < 1$ the quantity ξp_{B_n} UTXO's is burned. When $\xi > 1$ the quantity ξp_{B_n} UTXO's are minted (above the scheduled coinbase).

Crucially, this method of proportional adjustment of the fee paid to the miner does not alter the Nash equilibrium of the pattern of fees offered by transactors. This is readily apparent by considering the incentive for A or B to alter their fee after a proportional adjustment ξ of the miner's fee. Miners will still have an incentive to append transactions to blocks in ascending order of fees. If B does not change its fee, A's incentives are unaffected. It still must offer a higher fee than B to secure a position in the queue of transactions ahead of B. Similarly, if B was unwilling to offer a higher fee than A before the adjustment, it not have an incentive to do so after the adjustment.

5.1.1 Addition and Subtraction of UTXO Value

When $p^M_{B_n} - p_{B_n} > 0$, the difference is reflected in an additional payment to the miner above the total block reward. When $p^M_{B_n} - p_{B_n} < 0$, the difference is reflected in a subtraction in payment to the miner from the total block reward. The source of increased UTXO value in the former case (minting or sending from another address) and the allocation of the unused UTXO value in the latter case, are dealt with by the *Targeted* monetary policy in Section 7.

5.2 The puzzle difficulty signal and policy switch points

Equation 5 shows that hashrate N is over-determined. In addition to the miner block reward P^M_B, it is affected by mining cost c and the exchange rate e. These variables move independently of each other. This means the block reward instrument will not be able to perfectly achieve the target. To compensate, $Target$ dynamically adjusts the block reward to move hashrate toward the target interval over time.

The mechanism works as follows. The endpoints of the hashrate target interval, $\{N_{UB}, N_{LB}\}$ map into puzzle difficulty signals $\{\frac{D}{T}(N_{UB}), \frac{D}{T}(N_{LB})\}$, which bound the puzzle difficulty target interval. An adjust-

[24] See Theorem 1 and Proposition 2. Note that the result relies on parametric assumptions. The reader can consult Huberman et.al. [12] for details.

ment to the miner's block reward can be made after the end of each epoch when puzzle difficulty is adjusted. Epochs are numbered sequentially in temporal order, starting from 1. Epoch 1 includes blocks numbered 1 (Genesis) to m. Epoch 2 includes blocks numbered $m+1$ to $2m$, and so forth. The end-block of epoch q is denoted E_q. The ratio D/T is measured for epoch q at block E_q (the "adjustment block"). D is the puzzle difficulty in effect for epoch q and N is the average hashrate in epoch q. It does not include the adjustment to D made after E_q is appended to the blockchain. The monotone relationship between the sufficient statistic and hashrate implies that hashrate is above target when $D/T > \frac{D}{T}(N_{UB})$ and below target when $D/T < \frac{D}{T}(N_{LB})$.

Targeted imposes a ceiling on the miner's block reward in epoch $q+1$ when D/T, measured for epoch q, is above the target interval. After a ceiling has been put in place, it is lowered at each subsequent adjustment block until D/T drops below $\frac{D}{T}(N_{UB})$. Similarly, *Targeted* imposes a floor on the miner's block reward in epoch $q+1$ when D/T, measured for epoch q when D/T is below $\frac{D}{T}(N_{LB})$. When a floor has been put in place, it is raised at each subsequent adjustment block E_q until the sufficient statistic rises above $\frac{D}{T}(N_{LB})$. When D/T is inside the target interval, if a ceiling is in place from the prior adjustment block, the ceiling is raised, and so forth until it is removed altogether. If a floor is in place from the previous adjustment block, the floor is lowered at the current adjustment block, and so forth until it is removed altogether. The switching is displayed in Figure 4. At E_1 the sufficient statistic crosses the $\frac{D}{T}(N^{UB})$ switch-point from below. This turns "on" the instrument that imposes a ceiling on the miner block reward, which is lowered at epoch E_2 and raised at E_3. D/T is below $\frac{D}{T}(N_{LB})$ at E_4, which causes the ceiling to be removed and a floor to be imposed. The floor is raised at E_5.

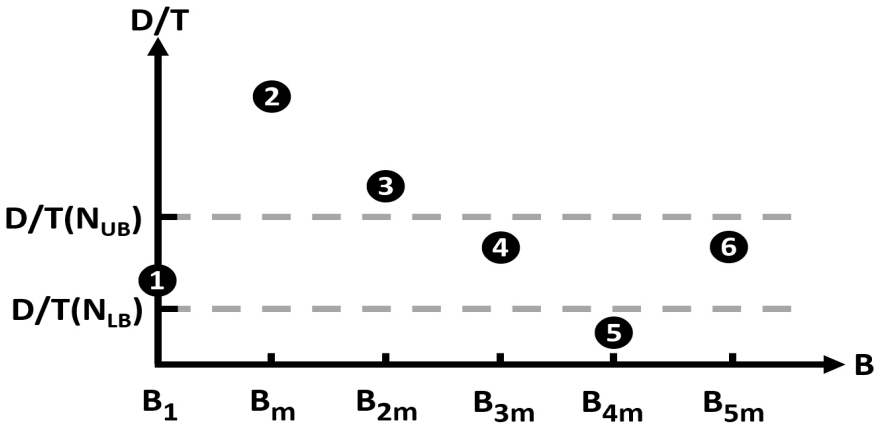

Figure 4: Mining Puzzle Difficulty Signal Switch Points

The dynamic feature of the policy can be seen when another RHS variable of Equation 6 is increasing. For example, if the exchange rate is rising when D/T is above $\frac{D}{T}(N_{UB})$, the block reward ceiling will be pushed down at each adjustment point. It may transpire that hashrate continues to rise for some period of time, but

so long as the exchange rate is bounded, the absolute value of the block reward ceiling adjustments will eventually rise above the exchange rate increase, which will push down hashrate.

5.3 Policy to control hashrate

The Hashrate control algorithm controls the imposition of ceilings and floors on the miner's block reward.[25]

The Hashrate Control Algorithm A ceiling or floor on the miner's block reward is adjusted after the end-block E_q of each epoch q as a function of D/T. There are four cases to consider.

Set parameters $\tau, \gamma \in [0, 1]$.

Case 1: Hashrate is above the upper bound of the hashrate interval, $\frac{D}{T}(N_{UB})$.

A ceiling on the miner's block reward is imposed. For each block in epoch $q+1$, payment to miner is capped at $\text{ceil}_q = \tau\overline{P_B}(q),\ \tau < 1$, where $\overline{P_B}(q)$ is the median block reward in epoch q.

For each subsequent epoch until $\frac{D}{T}(N_{UB})$ is crossed from above, the ceiling is lowered by the formula $\text{ceil}_{q+1} \leftarrow \tau\text{ceil}_q$.

Case 2: The puzzle adjustment at block E_q implies that D/T is below the lower bound of the target hashrate interval, $\frac{D}{T}(N_{LB})$.

A floor on the miner's block reward is imposed. For each block in epoch $q + 1$, payment to miner is capped at $\text{floor}_q = (1 + (1 - \tau))\overline{P_B}(q)$, where $\overline{P_B}(q)$ is the median block reward in Epoch q.

For each subsequent epoch until $\frac{D}{T}(N_{LB})$ is crossed from below, the floor is raised by the formula $\text{floor}_{q+1} \leftarrow (1 + (1 - \tau))\text{ceil}_q$.

Case 3: The puzzle adjustment at block E_q implies that D/T is inside the target hashrate interval.

For each subsequent epoch until $\text{ceil}_q \leq \overline{P_B}(q)$, the ceiling is increased by the formula $\text{ceil}_{q+1} \leftarrow (1 + (1 - \gamma)\text{ceil}_q$ Thereafter, the ceiling is removed.

For each subsequent epoch until $\text{floor}_q \geq \overline{P_B}(q)$, the floor is reduced by the formula $\text{floor}_{q+1} \leftarrow \gamma\text{floor}_q$ Thereafter, the floor is removed.[26]

[25] Appendix A contains Python code for the Hashrate Control Algorithm.

[26] The formulas τ and $(1 + (1 - \tau))$ ensure symmetry in the adjustment of the ceiling and the floor, i.e. that the ceiling and floor adjust by the same percentage each epoch.

6 Strategic Effects of Block Adjustment Policy

The adjustment of the miner's block reward introduces a new variable into equilibrium profit formula of Equation 4. Treating the target hashrate as a point, $N^T = N_{UB} = N_{LB}$, the block reward can be expressed as $p^M_{B_n}(N^T - N)$. This shows that the block reward adjusts to deviations from the target hashrate. A question arises as to whether the block adjustment policy can induce strategic behavior that upsets the monotone response to block reward adjustments that was assumed in the previous section. Here we evaluate the stability of equilibrium at the targeted hashrate from two angles. First, we derive bounds on the block reward adjustment that ensure deviation from the equilibrium is unprofitable. Second, we show the adjustment mechanism causes hashrate to home in on the target under an assumption of constant miner profit. There may be other evaluations that could be carried out by placing different restrictions, but I believe the two chosen here show the reasonableness of the assumption that hashrate responds monotonically to changes in the block reward.[27]

6.1 Stability of Miner Equilibrium

Equation 7 is the profit of a miner with hashrate S at the target hashrate.

$$\pi(S) = \frac{S}{N^T} eP(N^T)^M_B(1 - \Delta_T) - cS \tag{7}$$

The profit from deviating and adding or subtracting hashrate by Δ_S is

$$\pi(S + \Delta_S) = \frac{S + \Delta_S}{N^T + \Delta_S} eP(N^T + \Delta_S)^M_B(1 - \Delta_T) - c(S + \Delta_S)$$

A deviation is unprofitable if $\pi(S + \Delta_S) < \pi(S),\ \forall \Delta_S$. Equation 8 is a sufficient condition.

$$P(N^T + \Delta_S) < \frac{N^T + \Delta_S}{S + \Delta_S} \times \frac{S}{N^T} P(N^T)^M_B \tag{8}$$

Since $S \leq N^T$ by definition, it is immediate that, in order to maintain the inequality, an increase in hashrate, $\Delta_S > 0$ must cause the block reward to decrease and a decrease in hashrate does the opposite. This is consistent with the block adjustment policy. We can derive bounds on the magnitude of adjustment by evaluating the required adjustment at the extremes of $\Delta_S = 1$ and $\Delta_S = 0$. This yields the range of adjustment multiple $(\frac{1+\Delta_S/N^T}{1+\Delta_S}, 1)$. Since the lower bond approaches 1 as Δ_S increases from 1 to N^T, the

[27] A maintained assumption of this paper is that block timestamps are correct. I do not explore whether *Targeted* creates a new vulnerability to attack by miner manipulation of timestamps.

lower bound can be restricted to its value at $\Delta_S = 1$, which yields $\frac{1+1/N^T}{2}$.[28]

6.2 Dynamic Adjustment of Hashrate

This section evaluates the dynamic response of hashrate to changes in the value of the miner's block reward which occur when hashrate is outside the target interval. The dynamic adjustment of hashrate to a change in the miner's block reward can be evaluated when hashrate adjusts to maintain a profit ϕ per target time T^*.[29]

Consider an epoch q where the total block reward is constant at p_B^M and the puzzle difficulty adjustment at the end of epoch $q-1$ achieves the target growth rate when hashrate is N, and N is the equilibrium hashrate when the miner's block reward is p_B^M. Suppose a ceiling or floor is imposed at the end of epoch $q-1$ resulting in a marginal change in the miner's block reward equal to Δp (i.e. holding fixed the exchange rate, e, and hashrate cost c). Let Δ_T denote the distance between the actual blockchain growth rate and the target blockchain growth rate induced by a change in hashrate in epoch q, denoted ΔN.

$\Delta_T = T^* - T = \frac{D}{N} - \frac{D}{N+\Delta N}$

Equation 9 is the equilibrium of an interval of T^* units of time in epoch q.

$$\pi_{\Delta p,\Delta N} = \underbrace{\frac{1}{N+\Delta N}}_{\text{lottery effect}}\Big(\underbrace{[1+D\{\underbrace{\frac{1}{N}-\frac{1}{N+\Delta N}}_{\Delta_T}\}]}_{\text{growth rate effect}}(e(P_B^M+\Delta p)\Big) - c = \phi^{30} \quad (9)$$

The change in miner profit can be decomposed into a lottery effect and a growth rate effect. The lottery effect reflects that a change in hashrate changes the odds of any unit of hashrate mining the next block. The growth rate effect reflects that a change in hashrate alters the blockchain growth rate, which changes the number of puzzle lotteries in which a unit of hashrate can compete in T^* units of time. Noting that $\frac{D}{N}|modulo\ T^* = 1$, the growth rate effect is $2 - \frac{D}{N+\Delta N}$. Equation 10 is the marginal effect of hashrate on profit.

[28] It is straightforward to add this constraint to the block adjustment formula replacing the point N^T with the interval interval N_{LB}, N_{UB}.

[29] The assumption of a constant profit rules out the possibility of a dynamic strategy to influence hashrate by a miner coalition. Arguably, ϕ internalizes the exercise of market power.

$$\frac{\Delta\pi}{\Delta N} = \big(e(P_B^M + \Delta p)\big)\Big[\frac{\Delta\text{lottery effect}}{\Delta N} + \frac{\Delta\text{growth rate effect}}{\Delta N}\Big]$$
$$= -2e(P_B^M + \Delta p)\frac{1}{(N+\Delta N)^2} < 0 \quad (10)$$

Equation 11 is the marginal effect of the miner's block reward on profit.

$$\frac{\Delta\pi}{\Delta p} = \text{lottery effect} \times \text{growth rate effect} \times p_B^M > 0 \quad (11)$$

Using the implicit function theorem to combine equations 10 and 11 shows that an increase in the block reward induces an increase in hashrate (and vice versa) when hashrate adjusts to maintain a constant profit rate per unit of time $\Delta N/\Delta p = -\frac{\Delta\pi}{\Delta p}/\frac{\Delta\pi}{\Delta N} = +/+ > 0$.[31] The key point is that, when hashrate is responsive to changes in the block reward, the marginal effect of the Hashrate Control Algorithm is to induce an adjustment of hashrate in the intended direction when a ceiling or floor pushes the miner's block reward below or above the total block reward.[32] The foregoing proves the following proposition.[33]

Proposition 1. *The marginal effect of the Hashrate Control Algorithm is to dynamically push hashrate toward the target interval.*

7 Monetary Neutrality

This section describes a protocol that ensures the implementation of the Hashrate Control Algorithm does not cause any substantive change to the Nakamoto monetary policy in terms of the spending potential of addresses. *Targeted* induces offsetting additions or subtractions from the UTXO value of an address that occurs when the address sends its UTXO value in a transaction. The concept of money supply is adapted to include the UTXO adjustments that are committed for future transactions.

[31] The analysis of dynamic equilibrium can be extended to the case where the difficulty adjustment at the end of epoch q does not achieve the target growth rate. In that case the ratio $D/N \neq T^*$ and

$$\frac{\Delta\pi}{\Delta N} = -e(P_B^M + \Delta p)\frac{1}{(N+\Delta N)^2}\big(1 + D/N\big) < 0$$

which yields the same result in terms of the direction of the response of hashrate to the value of the miner's block reward.

[32] Assuming that e and c are bounded, the result can easily be extended to show that the Hashrate Control Algorithm will eventually push hashrate toward the target interval, but it will not necessarily do so monotonically.

[33] Bissias et.al. (2022) [2] document that miner spending allocation between Bitcoin and Bitcoin Cash adjust at high frequency to maintain a zero arbitrage condition. This validates that hashrate tends to adjust quickly. A richer formulation of my model would include the shadow profit of mining an alternative blockchain.

Definition 1. *Spending Potential*

(i) ***Address Spending Potential*** *The UTXO value in address x plus the UTXO value that is scheduled to be added to or subtracted from x when it sends a transaction.*

(ii) ***Aggregate Spending Potential*** *The sum of the spending potential of addresses plus the increase or decrease in spending potential that is not currently assigned to addresses, but which under the protocol can be assigned in the future.*

The concept of monetary neutrality has two dimensions.

Definition 2. *Monetary Neutrality*

(i) ***Aggregate Monetary Neutrality*** *Aggregate UTXO spending potential under Targeted is the same as Nakamoto at every block.*

(ii) ***Distributional Monetary Neutrality*** *At any block, an addition or subtraction of UTXO from the total block reward is allocated to an address in proportion to its percentage of aggregate spending potential.*

7.1 The *Targeted* Monetary Policy

This section presents a high-level description of the monetary policy as it affects a single address x. Python code describing the logic of the policy is in Appendix B.

Subtractions from the total block reward: When a UTXO value of q is subtracted from the total block reward at block B_n the miner is instructed to send q to an address denoted the "UTXO pool". The UTXO pool is a single address from which UTXO value is sent to other addresses and has an initial value P to which q is added. Every address x holding UTXO value is assigned spending potential in the UTXO pool (the "assigned value") which is denoted e_x. When q is sent to the UTXO pool, x's assigned value is increased by x's fraction of aggregate UTXO spending potential implied by the state after the block B_n transactions, denoted $S_{x|B_{n-1}}$ (x's "percentage") times q, rounded down to the nearest Satoshi.

Additions to the total block reward: When q is added to the total block reward at block B_n there are two possible adjustments to accounts. If $P - q > 0$, x's assigned value in the UTXO pool is reduced by the product of its percentage times q, rounded down to the nearest Satoshi. When $P - q \leq 0$ two accounting changes occur; (i) P and x's assigned value e_x are each reduced to zero and (ii) the value $q - P$ is added to an account that tracks value that is subtracted from the spending potential of addresses (the "Remainder Accounts") which has initial value R. Each address x is assigned a value in the remainder account, denoted r_x, which is increased by its percentage times $P - q$, rounded down to the nearest Satoshi.[34]

[34]The UTXO value subtracted from P can either be burned or sent to the miner. The latter would reduce the minting of UTXO required to increase the miner's block reward.

Note that the spending potential of an address x with UTXO value x is $x + e_x - r_x$.

Rounding Adjustment At each block B_n the unassigned UTXO value in the UTXO pool and the Remainder Accounts are assigned to the individual address accounts, e_x and r_x in accordance to the percentage of address x, rounded down to the nearest Satoshi.

Transactions Prior to the adjustments to the UTXO pool and Remainder pool, the transactions in block B_n are executed. An adjustment is made when address x sends UTXO value in a transaction that is appended to block B_n. If $e_x - r_x > 0$, the difference is sent to x. If $r_x - e_x > 0$, the difference is burned from x's UTXO value.[35] The initial values of e_x and r_x are the ending values for block B_{n-1}. The output values of $S_{x|B_{n,1}}$, e_x and r_x are the initial state variables in the UTXO pool and the Remainder Accounts adjustments.

Figure 5 depicts the operation of the *Targeted* monetary policy at block B_n. The timing sequence is denoted by the circled numbers. The first step is the execution of the transactions. The adjustment to a transaction sender's UTXO value is determined by the difference between its account value in the UTXO pool and its account value in the Remainder Accounts. The second step is the payment of the miner block reward. If $P^M_{B_n} < p^M_{B_n}$, UTXO value of q is minted to cover the payment. The third step is to send of UTXO value q to the UTXO Pool if $P^M_{B_n} < p^M_{B_n}$ or else burn UTXO value up to q in the UTXO pool and increase the value of the Remainder Accounts by the residual. Adjustments to the address account values in the UTXO Pool and the Remainder Accounts are made.[36]

[35] The burning requirement invalidates the transaction when the burn value exceeds the unspent UTXO.

[36] Steps 2 and 3 can be simultaneous. When $q < 0$ the minting of UTXO value is reduced by the amount of value in the UTXO pool, which is sent to the miner from the UTXO pool. This is implicit in the code in Appendix B.

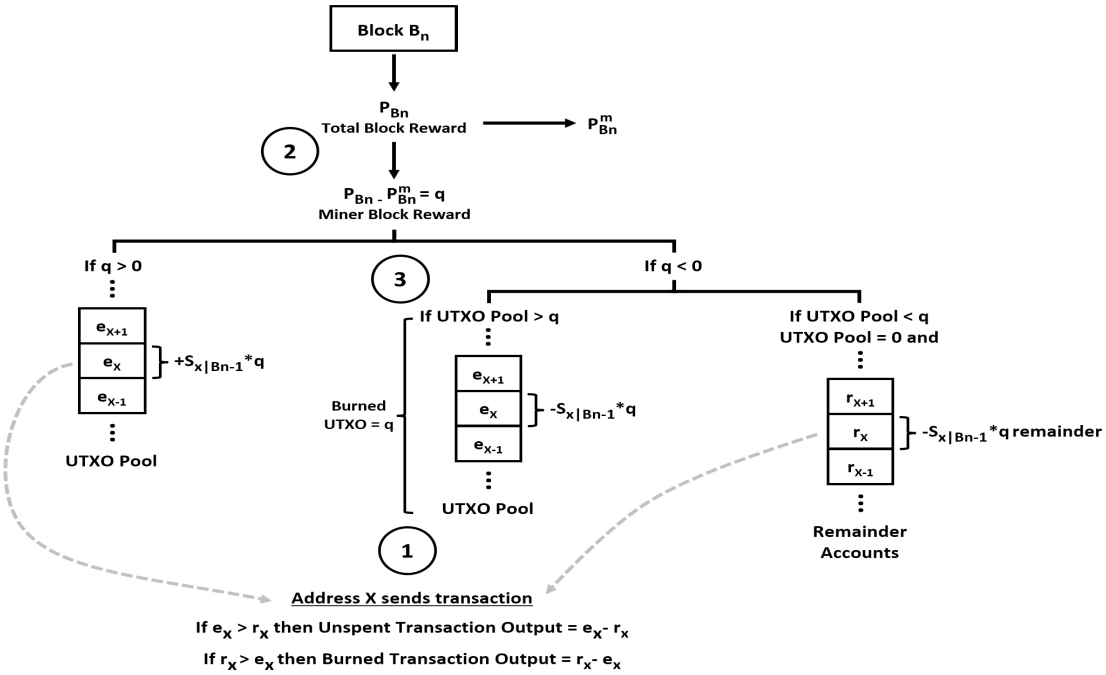

Figure 5: Targeted Monetary Policy

Proposition 2. *Targeted Achieves Monetary Neutrality*

Proof. The difference between the total block reward and the miner's block reward, q, is offset by changes of opposite sign in the value of the UTXO pool and the Remainder Accounts; $q = \Delta P + \Delta R$. This proves aggregate monetary neutrality. The value change in the UTXO Pool and the Remainder Accounts are allocated to addresses in proportion to their spending potential, rounded down to the nearest Satoshi and the unassigned residual is assigned to addresses in accordance with the same formula at each block. This proves distributional neutrality. □

8 Conclusion

Targeted is motivated by two observations about PoW blockchains. One observation is the two external hashrate costs of network security and CO^2 emissions, the former decreasing and the latter increasing in hashrate. This observation suggests there is an interval of hashrate in which total costs are minimized. The second observation is that hashrate is a function of the value of the block reward. Nakamoto does not provide an incentive for miners to set hashrate at the optimal level. *Targeted* incentivizes miners to home in on a target hashrate interval. This is achieved by imposing a floor to the miner's block reward when hashrate is

below target and imposing a ceiling on the miner's block reward when hashrate is above target. The ceiling is progressively lowered and the floor is progressively raised for so long as hashrate is outside the target interval. *Targeted* achieves aggregate and distributional monetary neutrality. Aggregate monetary neutrality means that the UTXO spending potential of the network is the same under *Targeted* and Nakamoto at each block. Distributional monetary neutrality means that the UTXO spending potential of addresses are adjusted to offset increases and decreases in the block reward in proportion to their UTXO value.

Identifying the optimal hashrate interval entails some difficulty. Three of the key inputs; estimating hashrate, ASIC efficiency and electricity cost, are, in principle, solvable. But quantifying the other two inputs; the costs of CO^2 emissions and network security, are elusive. To implement *Targeted* at the present time will require the formation of a consensus among network participants concerning those quantifications.

References

[1] Daniel Aronoff and Isaac Ardis. Adess: A proof-of-work protocol to deter double-spend attacks. In Kohei Arai, editor, Advances in Information and Communication, pages 131–157, Cham, 2024. Springer Nature Switzerland.

[2] George Bissias, Rainer Böhme, David Thibodeau, and Brian Levine. Pricing security in proof-of-work systems. WEIS 2022, 2022.

[3] Eric Budish. The economic limits of bitcoin and the blockchain. Working Paper 24717, National Bureau of Economic Research, June 2018.

[4] Miles Carlesten, Harry Kalodner, Arvind Naryanan, and S. Matthew Weinberg. On the instability of bitcoin without the block reward. CCS'16, Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pages 154–167, October 2016.

[5] Coinmetrics. Cmbi mining series methodology version 1.0. Technical report, Coinmetrics, 2020.

[6] Lin William Cong, Zhiguo He, and Jiasun Li. Decentralized mining in centralized pools. The Review of Financial Studies, 34:1191–1235, March 2021.

[7] Simon Dietz, Nicholas Stern, and William D. Nordhaus. Endogenous growth, convexity of damage and climate risk: How nordhaus' framework supports deep cuts in carbon emissions. The Economic Journal, 125(583):574–620, 2015.

[8] Cambridge Centre for Alternative Finance. Cambridge bitcoin electricity consumption index, 2024.

[9] Meredith Fowlie, Mar Reguant, and Stephen P. Ryan. Market-based emissions regulation and industry dynamics. The Journal of Political Economy, 124(1), 2016.

[10] Ulrich Gallersdorfer, Lena Klassen, and Christian Stoll. Energy consumption of cryptocurrencies beyond bitcoin. Joule Commentary, 4:1839–1851, September 2020.

[11] Arthur Gervais, Vasileios Glykantzis, Ghassan O. Karame, Karl Wurst Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. CCS '16: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pages 3–16, September 2016.

[12] Gur Huberman, Jacob D. Leshno, and Ciamac Moallemi. Monopoly without a monopolist: An economic analysis of the bitcoin payment system. Review of Economic Studies, 88:3011–3040, November 2021.

[13] Jacob D. Leshno and Phillipe Strack. Bitcoin: An axiomatic approach and an impossibility theorem. American Economic Review: Insights, 2:269–286, September 2020.

[14] Che Long, Lin William Cong, and Yizhou Xiao. A Brief Introduction to Blockchain Economics, chapter Chapter 1, pages 1–40. World Scientific, 2020.

[15] Michael Mirkin, Lulu Zhou, Ittay Eyal, and Fan Zhang. Sprints: Intermittent blockchain PoW mining. In 33rd USENIX Security Symposium (USENIX Security 24), pages 6273–6289, Philadelphia, PA, August 2024. USENIX Association.

[16] Daniel J. Moroz, Daniel J. Aronoff, Neha Narula, and David C. Parkes. Double-spend counterattacks: Threat of retaliation in proof-of-work systems, 2020.

[17] Satoshi Nakamoto. Bitcoin; a peer - to - peer electronic cash system, 2008.

[18] William Nordhaus. Climate change: The ultimate challenge for economics. The American Economic Review, 109(6):1991–2014, 2019.

[19] Emiliano S Pagnotta. Decentralizing money: Bitcoin prices and blockchain security. The Review of Financial Studies, 35(2):866–907, 01 2021.

[20] Johannes Sedlmeir, Hans Ulrich Buhl, Gilbert Fridgen, and Robert Keller. The energy consumption of blockchain technology: Beyond myth. Business Information Systems Engineering, 62:599–608, June 2020.

[21] Jean Tirole. Asset bubbles and overlapping generations. Econometrica, 53(6):1499–1528, 1985.

A Code for The Hashrate Control Algorithm

This appendix states the logic steps of the Hashrate Control Algorithm. It is not the code that would be written to implement the *Targeted* on servers in the Bitcoin network.

1: Python code for Hashrate Control Algorithm

```
class Epoch:
    def __init__(self):
        self.rewards = []  # To store the rewards of each block
        within an epoch
        self.hashrates = [] # To store the hashrates of each block
        within an epoch
        self.ceil = None
        self.floor = None
        self.difficulty = None
        self.elapsed_time = None # T_n

    def median_block_reward(self):
        sorted_rewards = sorted(self.rewards)
        return np.median(sorted_rewards)

    def average_hashrate(self):
        sorted_hashrates = sorted(self.hashrates)
        return np.mean(sorted_hashrates)

    def add_reward(self, reward):
        self.rewards.append(reward)

    def add_hashrate(self, n):
        self.hashrates.append(n)

class Blockchain:
    def __init__(self, tau, gamma, upper_bound, lower_bound, initial_difficulty):
        self.tau = tau
        self.gamma = gamma
        self.epochs = [Epoch()]
        self.DT = None  # Puzzle difficulty/Blockchain growth rate;
        get this value from blockchain source code
        self.DT_N_UB = upper_bound  # Upper boundary
        self.DT_N_LB = lower_bound  # Lower boundary

    def adjust_reward(self, reward, epoch_idx):
        epoch = self.epochs[epoch_idx]
        if epoch.ceil is not None:
            reward = min(reward, epoch.ceil)
        if epoch.floor is not None:
            reward = max(reward, epoch.floor)
        epoch.add_reward(reward)

    def adjust_hashrate(self, hashrate, epoch_idx, e_current, P_current,
        efficiency_current, electricity_cost_current):
        alpha1 = None
```

```
    alpha2 = None
    alpha3 = None
    alpha4 = None
    # Found through regression on past values

    epoch = self.epochs[epoch_idx]
    log_eP_current = np.log(1+ (e_current * P_current))
    log_eff_current = np.log(1+efficiency_current)
    log_electricity_cost_current = np.log(1+electricity_cost_current)
    # Put it all together:
    # bN_{t+1} = alpha1 + alpha2*log(eP_t) + + alpha3*log(efficiency_t) + alpha4*
        log(c_t)
    predicted_log_hashrate = (
        alpha1
        + alpha2 * log_eP_current
        + alpha3 * log_eff_current
        + alpha4 * log_electricity_cost_current
    )

    # Exponentiate to get N_{t+1}
    new_hashrate = np.exp(predicted_log_hashrate)
    epoch.add_hashrate(new_hashrate)
    return new_hashrate

def end_of_epoch(self):
    last_epoch = self.epochs[-1]
    P_B_median = last_epoch.median_block_reward()
    N_n = last_epoch.average_hashrate()
    new_epoch = Epoch()

    BLOCKS_PER_EPOCH = 14
    T_STAR = BLOCKS_PER_EPOCH * 10 * 60 # 1,209,600 for 2016 blocks
    D_n = last_epoch.difficulty

    T_n = (BLOCKS_PER_EPOCH * D_n * 2**32) / N_n
    last_epoch.elapsed_time = T_n

    T_hat = max(T_STAR /4, min(T_n, 4 *T_STAR))
    D_next = D_n * (T_STAR / T_hat)
    new_epoch.difficulty = D_next

    # Case 1: Hashrate too high - need to decrease rewards
    if self.DT > self.DT_N_UB:
        #("Case 1: Hashrate above upper bound")
        new_epoch.ceil = (self.tau) * P_B_median
        new_epoch.floor = None  # Clear any existing floor

    # Case 2: Hashrate within bounds - gradual adjustment
    elif self.DT_N_LB < self.DT < self.DT_N_UB:

        # Previous epoch had a ceiling
        if last_epoch.ceil is not None:
            if last_epoch.ceil >= self.DT:
                # Gradually relax the ceiling
                new_epoch.ceil =  (1+self.gamma) * last_epoch.ceil
```

```
            else:
                # Remove ceiling if it's no longer needed
                new_epoch.ceil = None

        # Previous epoch had a floor
        elif last_epoch.floor is not None:
            if last_epoch.floor <= self.DT:
                # Gradually relax the floor
                new_epoch.ceil = (self.gamma) * last_epoch.floor
            else:
                # Remove floor if it's no longer needed
                new_epoch.floor = None
        else:
            new_epoch.floor = last_epoch.floor
            new_epoch.ceil = last_epoch.ceil

    # Case 3: Hashrate too low - need to increase rewards
    elif self.DT < self.DT_N_LB:
        #("Case 3: Hashrate below lower bound")
        new_epoch.floor =  (1+(1-self.tau)) * P_B_median
        new_epoch.ceil = None  # Clear any existing ceiling

    self.epochs.append(new_epoch)
```

B Code for the *Targeted* Monetary Policy

This appendix states Python code for the *Targeted* monetary policy. The interpretation of the variables are commented. The code states the logic for the re-allocation of spending potential of addresses at each block that is induced by transactions and the addition or subtraction of the block reward, $p^m_{B_n} - p_{B_n}$, in accordance with the Hashrate Control Algorithm. It is not the code that would be written to implement the *Targeted* on servers in the Bitcoin network.

2: Python code for *Targeted* monetary policy

```
def update_utxo_values(p_B_n_m, p_B_n, S_x_B_n_minus_1, P, FP, e_values, r_values, R,
    FR):
    q = p_B_n_m - p_B_n

    for x in range(len(e_values)):
        if q < 0:
            P += q
            e_values[x] = math.floor(e_values[x] + S_x_B_n_minus_1[x] * (q + FP))
        elif q > 0:
            if P > q:
                P -= q
                e_values[x] = math.floor(e_values[x] - S_x_B_n_minus_1[x] * (q - FP))
            else:
                R += q - P
                r_values[x] = math.floor(r_values[x] + S_x_B_n_minus_1[x] * (P + FR))
```

```
                P = 0
                e_values[x] = 0

    x_values = [x_initial[i] + e_values[i] - r_values[i] for i in range(len(e_values))
        ]
    return P, e_values, r_values, R, FP, FR, x_values

def process_transactions(x_initial, e_values, r_values, transaction_indices):

# initial spending potential of addresses
x_values = [x_initial[i] + e_values[i] - r_values[i] for i in range(len(e_values))]

    #transaction sender update step
    for i in transaction_indices:
        x_value = math.floor(x_initial[i] + e_values[i] - r_values[i])
        e_values[i] = 0
        r_values[i] = 0

     #miner update step
     x_value[miner_index] += p_B_n_m

    total_sum = sum(x_values[i] + e_values[i] - r_values[i] for i in range(len(
        e_values)))

     # Spending potential shares of addresses after transactions and miner fee, before
         allocations to UTXO Pool and Remainder Accounts
     S_x_B_n_minus_1 = [(x_values[i] + e_values[i] - r_values[i]) / total_sum for i in
         range(len(e_values))]

    return e_values, r_values, S_x_B_n_minus_1

def update_fp_fr(FP, FR, e_values, r_values):
    FP += sum(e_values)
    FR += sum(r_values)
    return FP, FR

def compute_S_x_B_n(x_values, e_values, r_values):
    total_sum = sum(x + e - r for x, e, r in zip(x_values, e_values, r_values))

     # Spending potential shares of addresses after transactions and miner fee and
         assingments to UTXO Pool and Remainder Accounts.
     S_x_B_n = [(x + e - r) / total_sum for x, e, r in zip(x_values, e_values,
         r_values)]
    return S_x_B_n

Interpretation of variables:

x_initial = [<values to be inserted>]  # initial values for x
e_initial = [<values to be inserted>]  # initial UTXO values of addresses
r_initial = [<values to be inserted>]  # initial values of r_x

p_B_n_m = <value to be inserted> # miner's block reward
p_B_n = <value to be inserted> # total block reward
P = <value to be inserted> # aggregate value of UTXO Pool
e_values = e_initial[:]  # copy of initial UTXO values in UTXO Pool
```

```
r_values = r_initial[:]  # copy of initial values of r_x, the value for each address
    in the Remainder Accounts
R = <value to be inserted> #  aggregate value in the Remainder Accounts
FP = <value to be inserted> #  sum of values e_x in the UTXO Pool
FR = <values to be inserted> # sum of values r_x in the Remainder Accounts
transaction_indices = [:] # indices of addresses that send transactions in block B_n
miner_index = x # address of the miner of block B_N

Computation:

# Process transactions for each address in transaction_indices
e_values, r_values, S_x_B_n_minus_1 = process_transactions(x_initial, e_values,
    r_values, transaction_indices, p_B_n_m, miner_index)

# Update UTXO values for each address
P, e_values, r_values, R, FP, FR, x_values = update_utxo_values(p_B_n_m, p_B_n,
    S_x_B_n_minus_1, P, FP, e_values, r_values, R, FR)

# Update FP and FR with the residuals
FP, FR = update_fp_fr(FP, FR, e_values, r_values)

# Compute S_x_B_n for each address
S_x_B_n = compute_S_x_B_n(x_values, e_values, r_values)

# Compute the sum of initial and final x values
initial_x_sum = sum(x_initial)
final_x_sum = sum(x_values)
x_difference = initial_x_sum - final_x_sum

print("Updated values:")
print(f"P: {P}, e_values: {e_values}, r_values: {r_values}, x_values: {x_values}, R: {
    R}, FP: {FP}, FR: {FR}")
print(f"S_x_B_n: {S_x_B_n}")
print(f"Difference between the sum of initial and final x values: {x_difference}")
```