

Greedy but Cautious: Conditions for Miner Convergence to Resource Allocation Equilibrium

George Bissias¹, Brian N. Levine¹, and David Thibodeau²

¹College of Information and Computer Sciences, UMass Amherst,
{gbiss,levine}@cs.umass.edu

²Florida Department of Corrections,
davidpthibodeau@gmail.com

August 27, 2019

Abstract

All public blockchains are secured by a proof of opportunity cost among block producers. For example, the security offered by proof-of-work (PoW) systems, like Bitcoin, is due to spent computation; it is work precisely because it cannot be performed for free. In general, more resources provably lost in producing blocks yields more security for the blockchain. When two blockchains share the same mechanism for providing opportunity cost, as is the case when they share the same PoW algorithm, the two chains compete for resources from block producers. Indeed, if there exists a liquid market between resource types, then theoretically all blockchains will compete for resources. In this paper, we show that there exists a resource allocation equilibrium between any two blockchains, which is essentially driven by the fiat value of reward that each chain offers in return for providing security. We go on to prove that this equilibrium is singular and always achieved provided that block producers behave in a greedy, but cautious fashion. The opposite is true when they are overly greedy: resource allocation oscillates in extremes between the two chains. We show that these results hold both in practice and in a block generation simulation. Finally, we demonstrate several applications of this theory including a trustless price-ratio oracle, increased security for blockchains whose coins have lower fiat value, and a quantification of cost to allocating resources away from the equilibrium.

1 Introduction

Cryptocurrencies such as Bitcoin [22] are a confluence of systems engineering, economics, and game theory. In some ways, cryptocurrency economics is a prosaic application of classic economic theory. For example, the economy defined by Bitcoin has an extremely simple monetary policy: a fixed coin issuance schedule, which makes inflation entirely predictable¹. However, the procedural properties

¹Coin destruction, through loss of private keys, is much more difficult to measure.

of cryptocurrencies, more software than policy, give rise to remarkably crisp economic tradeoffs that manifest surprisingly regular macro-level phenomena. In this paper, we examine one such phenomenon that arises from the dynamics of the so-called difficulty adjustment algorithms (DAAs) used by blockchains that employ proof-of-work (PoW) for security. We show that, as predicted by Spiegelman et al. [27], but in contrast to the model of Kwon et al. [17], much of aggregate miner behavior can be explained by their proclivity to increase immediate profit. More specifically, we offer a rationale for the division of hash rate that is manifest between two blockchains that share the same PoW algorithms. Remarkably, the same reasoning generalizes to the division of security resources between blockchains using different PoW algorithms and even using different consensus mechanisms altogether. We make the following contributions over prior work.

- We define a hash rate *allocation equilibrium* between two blockchains that use the same proof-of-work (PoW) algorithm. We show that a unique allocation equilibrium exists and that it aligns with one of the Nash equilibria described by Kwon et al. [17]. The allocation equilibrium is much less stringent than a Nash equilibrium. The former assumes only that miners individually tend to act to maximize their profit according to a metric similar to the popular *difficulty adjusted reward index* (DARI). In contrast, the Nash equilibrium assumes that miners have knowledge of a complex utility function and the strategies of other miners at equilibrium.
- We prove the conditions under which hash rate allocation will converge to the allocation equilibrium and anticipate allocation dynamics when these conditions are not met.
- We show that, given an efficient market for buying and selling PoW hash rate, the allocation equilibrium generalizes to pairs of blockchains that use different PoW algorithms. We also show that the existence of efficient hash rate markets allows for generalization to equilibria between PoW and PoS blockchains.
- We empirically validate the existence of allocation equilibria between several of the top blockchain projects that share the same PoW algorithm, including BTC versus BCH, and ETH versus ETC. Their adherence to the equilibrium is found to be quite strong. We also corroborate our theoretical results in simulation, showing precisely the conditions under which pairs of blockchains converge to, or diverge from, the allocation equilibrium.
- We provide several applications for the allocation equilibrium in the real world, including: a trustless price-ratio oracle, increasing security for minority hash rate blockchains, and measuring the cost to miners who provide security to a blockchain beyond the equilibrium.

2 Related Work

Prat and Walter [23] modeled the impacts of mining difficulty and coin exchange rate on mining profitability for a single blockchain. They found that miners will not purchase new mining hardware if the fiat value of the coinbase reward is insufficient to accommodate the resulting rise in mining difficulty. The paper demonstrates empirically that this relationship holds quite well in practice for data ranging from 2012 until 2018. Furthermore, in the context of a single blockchain, Ma et al. [19] showed that there exists a Nash equilibrium for the computing power allocated by miners given a fixed mining difficulty.

The work of Kwon et al. [17] is similar to ours. They showed that there exist multiple Nash equilibria for miners who allocate their hash rate among two blockchains sharing the same PoW and inter-block time. One equilibrium, $\mathcal{E}_{\text{econ}}$ (which coincides with our allocation equilibrium), exists at the relative price ratios of the two coins, but for the others, which we denote collectively as $\mathcal{E}_{\text{others}}$, no subset of economically rational miners will dedicate their hash rate to the *discounted* blockchain having the lower coin price. And if a sufficiently large fraction of miners commit to mining the discounted blockchain, then they will be alone in mining on that chain. These results suggest the possibility that, in the presence of mostly rational miners, the discounted blockchain may end up supported solely by a centralized cabal of committed miners. Kwon et al. [17] further reported a tendency for profit-seeking miners to generally move toward $\mathcal{E}_{\text{econ}}$, but argued that they can also be pulled toward equilibria $\mathcal{E}_{\text{others}}$, and it is not clear how these dynamics play out in an iterative game. Moreover, their model involves a complex utility function, sophisticated strategies, and requires that each miner knows the strategy of the others in order to maintain equilibrium. In the present work, we assume only that a certain fraction of miners act so as to increase their immediate profit. We show formally that, under those conditions, hash rate allocated to the discounted chain will always converge to $\mathcal{E}_{\text{econ}}$. This result implies that the discounted chain can count on a minimum hash rate proportional to its coin’s price relative to competing chains. Moreover, we demonstrate empirically and in simulation the conditions under which convergence succeeds. Our analysis also generalizes to equilibria between blockchains using different PoW algorithms and even those using proof-of-stake (PoS).

Also closely related is the work of Spiegelman et al. [27] who apply the theory of Potential Games [21] to the problem of miner hash rate allocation across multiple blockchains. They prove that, regardless of individual hash rate and coinbase rewards for each of the blockchains, hash rate allocation will converge to a pure equilibrium provided that miners follow *better response learning*. The model assumes “minimal rationality on behalf of the players, i.e., that they follow an arbitrary better response step improving their individual payoffs.” Spiegelman et al. [27] do not identify a specific equilibrium point, nor do they specify what the *better response* should be. But their work anticipates some of the theoretical results we present in Section 3. Furthermore, they show that the equilibrium point can be changed by changing a blockchain’s coinbase reward, a property

that is emergent from the properties of the equilibrium and one that we exploit to increase security in Section 6.2. Altman et al. [9] reached similar conclusions as Spiegelman et al. [27] using a slightly different game theoretical model of hash rate allocation across cryptocurrencies and mining pools.

Meshkov et al. [20] introduced the term *coin-hopping* to describe the strategy that involves some subset of miners moving among blockchains using the same PoW according to which is most profitable at a given time. They showed that this behavior can lead to unstable block times and proposed a modified difficulty adjustment algorithm to compensate. Coin-hopping corresponds to our definition of *greedy behavior* in Section 3. Kirly and Lomoschitz [16] expanded on the study of the coin-hopping strategy, which they show can be profitable in the long-term for miners with at least 12% of the total hash power.

Han et al. [15] investigate double-spend on blockchains with relatively low hash rate instigated by either miners from a higher hash rate chain or attackers who purchase hash rate from a marketplace such as NiceHash². They find that double-spend transactions with fiat value on the order of 1e5 USD are sufficient to motivate Bitcoin miners to carry out an attack on Bitcoin Cash.

Several authors have sought to determine the optimal hash rate allocation between blockchains for *individual* miners or mining pools. Bissias et al. [10] argue that miners allocate their hash rate between multiple blockchains so as to minimize the risk associated with fluctuations in coin price. Cong et al. [12] make a similar argument except that their measure of risk is volatility in the payout rate between mining pools. Chatzigiannis et al. [11] extend this model to mining across blockchains with different PoW algorithms. All of the above approaches are complimentary to the present work, which seeks only to explain *aggregate* miner behavior. In fact, miner-specific behavioral choices help to explain why the aggregate hash rate allocation does not fully allocate to one chain over another (see Section 7.1 for details).

Sapirshstein et al. [25] devised a Markov Decision Process (MDP) for discovering optimal selfish mining [13] strategies. Gervais et al. [14] expanded the model to incorporate adjustable network parameters and include analysis of double-spend attacks. Sai et al. [24] extend the MDP of Gervais et al. [14] to model mining difficulty adjustment. The biggest differences between these approaches and the present work is that the former analyze optimal *deviant* behavior in single blockchains while the present work attempts to explain *protocol compliant* behavior across multiple blockchains.

3 Miner Allocation Among Blockchains

In this section, we consider two blockchains A and B , each generally using different PoW algorithms W_A and W_B . Having different PoW algorithms, we imagine that the sets of miners M_A and M_B of each coin are generally disjoint, but in the special case where $W_A = W_B$ or when the algorithms support the same mining hardware, the sets can be equal or intersect. The *native hash rate*

²<https://www.nicehash.com>

(hashes per second) for miner m is denoted $\mathcal{H}(m)$, and with \mathcal{H}_A and \mathcal{H}_B we denote the *aggregate* native hash rate on chains A and B , respectively. Through secondary markets such as NiceHash³, an economically rational miner $m \in M_A$ will trade her hash power in A for hash power in B when the latter can earn her more fiat reward during the next moments of mining. Thus, miners $M_A \cup M_B$ collectively represent the aggregate achievable security of coins A and B , which is fluid, subject to changes in the profitability of mining across chains A and B .

DEFINITION 1: The *spot hash price* at time s , $\mathcal{S}_X(s)$, is the quantity of hashes per second using PoW algorithm W_X that can be traded for 1 unit of fiat.

Using definition 1, and assuming a perfectly efficient hash rate market, we can translate native hash rate on chain B into units of native hash rate on chain A , a process we term *hash rate regularization*. The regularized hash rates for chains A and B are equal to $H_A = \mathcal{H}_A$ and $H_B = \mathcal{H}_B \frac{\mathcal{S}_A}{\mathcal{S}_B}$. The regularized, aggregate hash rate across chains is given by $H = H_A + H_B$, where H is native to chain A , a convention that we will follow throughout this document. By $H(m)$, we denote the regularized hash rate for miner m . Finally, define *regularized allocation vector* $\mathbf{w} = (w_A, w_B)$ (or simply *allocation* for brevity) to be the fraction of H that miners devote to chains A and B , respectively. The following are definitions useful for discussing miner hash rate allocations and their relationship with blockchain security.

DEFINITION 2: The *hash weight* of miner m , denoted $\mathcal{W}(m)$ is equal to $H(m)/H$, and the weight of a set of miners M , denoted $\mathcal{W}(M)$ is given by $\sum_{m \in M} \mathcal{W}(m)$.

DEFINITION 3: The *relative security* of chain X is the fraction of fiat value of PoW applied to that chain, which is given by

$$K_X = \frac{\mathcal{H}_X / \mathcal{S}_X}{\mathcal{H}_A / \mathcal{S}_A + \mathcal{H}_B / \mathcal{S}_B} = \frac{H_X / \mathcal{S}_A}{H_A / \mathcal{S}_A + H_B / \mathcal{S}_A} = \frac{H_X}{H_A + H_B}. \quad (1)$$

In terms of relative security, the regularized allocation vector is given by $\mathbf{w} = (K_A, K_B)$. Notice that the relative security for chain $X \in \{A, B\}$ is equivalent to the fraction of total available regularized hash rate (i.e., in terms of the W_A PoW algorithm) allocated to chain X . Thus, when chains A and B share the same PoW algorithm, \mathbf{w} gives the share of hash rate for each chain.

The target, expected block inter-arrival time for chain $X \in \{A, B\}$ is denoted T_X . In general, blocks from chains A and B will be produced at different times, but we require some method of marking time universally. Let τ be a discrete variable that represents the times when a block is mined on either chain A or B . At time τ , the actual inter-arrival time for the last block from chain X is given by $T_X(\tau)$, and the fiat coinbase value for chain X is given by $V_X(\tau)$.

³<https://www.nicehash.com>

Coinbase value decomposes into $V_X(\tau) = c_X P_X(\tau)$, where c_X is the quantity of X coins paid out per block and $P_X(\tau)$ is the fiat value of each X coin at time τ . Furthermore, define the *hash adjusted reward*⁴ (HAR) for chain X at time τ by $\pi_X(\tau) = \frac{V_X(\tau)}{T_X(\tau)H_X(\tau)}$. The HAR for chain X represents the expected fiat value of each regularized hash on chain X . Finally, define the relative reward of the two chains by $R(\tau) = \frac{V_A(\tau)}{V_A(\tau)+V_B(\tau)}$. Note that in this analysis we ignore the contribution of fees to the coinbase.

DEFINITION 4: A *security adjustment algorithm* (SAA) is any algorithm that adjusts the expected number of hashes required to mine blocks so that their expected inter-arrival time tends toward T ; when the block time reaches T , the SAA is said to be *at rest*. It is further assumed that the SAA is a function of the properties of previously mined blocks, thus it can only update the security after a new block is mined.⁵

DEFINITION 5: The *greedy choice* allocation, denoted \mathbf{w}_g , is one that yields that maximum weighted sum of HAR vector $\boldsymbol{\pi}(\tau) = (\pi_A(\tau), \pi_B(\tau))$, i.e. $\mathbf{w}_g^T \boldsymbol{\pi}$.

DEFINITION 6: Blockchains A and B are said to be at *allocation equilibrium* if when both SAAs are at rest, there exists no greedy choice in allocation, i.e. $\boldsymbol{\pi} = c(1, 1)$ for some constant c .

LEMMA 1: Assume that at time τ both SAAs have come to rest, relative reward R is stable, and the allocation vector is fixed at $\mathbf{w} = (xR, y(1 - R))$, where $x \geq 0$, $y \geq 0$, and $xR + y(1 - R) = 1$. Then the HAR vector is given by

$$\boldsymbol{\pi}(\tau) = \frac{V_A(\tau) + V_B(\tau)}{H} \left(\frac{1}{xT_A}, \frac{1}{yT_B} \right). \quad (2)$$

PROOF: For $X \in \{A, B\}$, if at time τ the SAA for chain X has come to rest, then the actual inter-block time $T_X(\tau)$ is approximately equal to its expected time T_X . Therefore, the HAR vector is given by

$$\begin{aligned} (\pi_A(\tau), \pi_B(\tau)) &= \left(\frac{V_A(\tau)}{T_A(\tau)H_A(\tau)}, \frac{V_B(\tau)}{T_B(\tau)H_B(\tau)} \right) \\ &= \left(\frac{V_A(\tau)}{T_A H_A}, \frac{V_B(\tau)}{T_B H_B} \right) \\ &= \frac{1}{H} \left(\frac{V_A(\tau)}{T_A w_A}, \frac{V_B(\tau)}{T_B w_B} \right) \\ &= \frac{1}{H} \left(\frac{V_A(\tau)}{xT_A R}, \frac{V_B(\tau)}{yT_B (1-R)} \right) \\ &= \frac{V_A(\tau) + V_B(\tau)}{H} \left(\frac{1}{xT_A}, \frac{1}{yT_B} \right). \end{aligned}$$

□

⁴The HAR is analogous to the popular *difficulty adjusted reward index* (DARI) metric, except that the latter normalizes by the blockchain difficulty.

⁵In practice, most PoW blockchains employ a *difficulty adjustment algorithm*, which adjusts a value that is inversely proportional to the mining target t . Because *difficulty* is ambiguously defined between blockchains, we opt for this definition instead.

THEOREM 1: Assume any choice of SAA for chains A and B (not necessarily the same), and further assume that total hash rate is fixed at H . When the relative reward stabilizes, there exists a unique equilibrium allocation

$$\mathbf{w}_e = \left(\frac{T_B R}{T_B R - T_A R + T_A}, \frac{T_A(1-R)}{T_B R - T_A R + T_A} \right), \quad (3)$$

which simplifies to

$$\mathbf{w}_e = (R, 1 - R), \quad (4)$$

if $T_A = T_B$.

PROOF: An equilibrium allocation is one where the HAR vector is homogeneous, i.e. the HAR values for chains A and B are equal. Assuming the SAAs are at rest and relative price is stable, from Lemma 1 we can surmise that HAR values will be equal iff $xT_A = yT_B$. We can solve this equation for x and y along with the simultaneous constraint $xR + y(1 - R) = 1$:

$$x_e = \frac{T_B}{T_B R - T_A R + T_A}, \text{ and } y_e = \frac{T_A}{T_B R - T_A R + T_A}.$$

Substituting x_e and y_e into the identity $\mathbf{w}_e = (x_e R, y_e(1 - R))$ (from the statement of Lemma 1) yields Equation 3, as desired. Moreover, because the constraints constitute a system of two linearly independent equations with two unknowns, \mathbf{w}_e must be the only equilibrium allocation. \square

We next derive results related to how miners behave relative to the equilibrium.

DEFINITION 7: The *distance* between two allocations \mathbf{w}_1 and \mathbf{w}_2 is given by the L1-norm of their difference: $|\mathbf{w}_1 - \mathbf{w}_2|$.

DEFINITION 8: The ϵ -greedy allocation policy moves the current allocation ϵ closer (in terms of Definition 7) to the greedy choice, e.g. if $\pi_A(\tau_0) > \pi_B(\tau_0)$, then $\mathbf{w}(\tau_1) = (w_A(\tau_0) + \frac{\epsilon}{2}, w_B(\tau_0) - \frac{\epsilon}{2})$.

DEFINITION 9: The set of miners *loyal*⁶ to chain $X \in \{A, B\}$, denoted by M_{X^*} , are those that will allocate all hash rate to chain X over chain Y regardless of the value of π_X relative to π_Y .

THEOREM 2: Assume that equilibrium allocation \mathbf{w}_e is fixed over an arbitrarily long period of time and loyal miner hash weights are such that $\mathcal{W}(M_{A^*}) \leq w_{eA}H$ and $\mathcal{W}(M_{B^*}) \leq w_{eB}H$. If the SAAs on both chains have come to rest and reward ratio $V_A(\tau)/V_B(\tau)$ is constant in τ , then from an allocation not at equilibrium and for sufficiently small ϵ , the ϵ -greedy allocation policy converges to the equilibrium allocation.

⁶Our definition of *loyal* is consistent with Kirly and Lomoschitz [16], but not Kwon et al. [17].

PROOF: We prove this result in two stages. In the first we show that a non-loyal miner, making an ϵ -greedy choice will always move in the direction of \mathbf{w}_e . In the second, we argue that non-loyal miners comprise sufficient hash weight to reach \mathbf{w}_e .

To prove the first stage, it will suffice to show that for a suitably small ϵ , the ϵ -greedy allocation $\mathbf{w}(\tau_1)$ always moves the current allocation $\mathbf{w}(\tau_0)$ closer to the equilibrium allocation \mathbf{w}_e . Without loss of generality, we may assume that $\pi_A(\tau_0) > \pi_B(\tau_0)$. In this case, because $w_B(\tau_0) = 1 - w_A(\tau_0)$, we need only show that $w_A(\tau_0) < w_{eA}$ to prove the theorem. This follows from the fact that our assumption implies that the greedy choice will increase w_A : $w_A(\tau_1) = w_A(\tau_0) + \frac{\epsilon}{2}$, which can only move the allocation closer to w_{eA} provided that $\frac{\epsilon}{2} < w_{eA} - w_A(\tau_0)$.

Before proceeding, note that because the reward ratio is stable, there exists an r such that $V_A(\tau)/V_B(\tau) = r$ for all τ . Similarly, because the SAAs are assumed to have come to rest, we assume that $T_X(\tau) = T_X$ for every τ and $X \in \{A, B\}$. We have

$$\begin{aligned} \pi_A(\tau_0) > \pi_B(\tau_0) &\Rightarrow \frac{V_A(\tau_0)}{H_A(\tau_0)T_A} > \frac{V_B(\tau_0)}{H_B(\tau_0)T_B} \\ &\Rightarrow \frac{V_A(\tau_0)}{Hw_A(\tau_0)T_A} > \frac{V_B(\tau_0)}{Hw_B(\tau_0)T_B} \\ &\Rightarrow \frac{a}{w_A(\tau_0)} > \frac{b}{1-w_A(\tau_0)} \\ &\Rightarrow w_A(\tau_0) < \frac{a}{a+b}, \end{aligned}$$

where $a = r/T_A$ and $b = 1/T_B$. On the other hand, similar reasoning shows that $w_{eA} = \frac{a}{a+b}$. So we have $w_A(\tau_0) < w_{eA}$, as required.

To prove the second stage, it will suffice to argue that the hash weight of non-loyal miners at time τ_0 is non-zero. We again assume without loss of generality that $\pi_A(\tau_0) > \pi_B(\tau_0)$. Let $M = M_A \cup M_B$, and note that by definition $\mathcal{W}(M) = H$. The hash weight of non-loyal miners is given by $\mathcal{W}(M) - \mathcal{W}(M_{A^*}) - \mathcal{W}(M_{B^*})$. In stage 1, we proved that $w_A(\tau_0) < w_{eA}$, which implies that $\mathcal{W}(M_{A^*}) < w_{eA}H$. And by assumption $\mathcal{W}(M_{B^*}) \leq w_{eB}H$. Finally, because $w_A(\tau_0) + w_B(\tau_0) = 1$, we know that $\mathcal{W}(M) - \mathcal{W}(M_{A^*}) - \mathcal{W}(M_{B^*}) > 0$, which implies that the hash weight of non-loyal miners at time τ_0 must be non-zero. \square

COROLLARY 1: If the SAAs on both chains have come to rest and reward ratio V_A/V_B is stable, then for any allocation within distance δ of the equilibrium allocation, following the $(2\delta + \epsilon)$ -greedy allocation policy, $\epsilon > 0$, causes divergence from the equilibrium allocation.

PROOF: Similar to the proof of stage 1 in Theorem 2, it will suffice to show that for any $\epsilon > 0$, the $(2\delta + \epsilon)$ -greedy allocation $\mathbf{w}(\tau_1)$ always moves the current allocation $\mathbf{w}(\tau_0)$ *further* from the equilibrium allocation \mathbf{w}_e . Let $\delta_A = |w_A(\tau_0) - w_{eA}|$ and $\delta_B = |w_B(\tau_0) - w_{eB}|$, which according to Definition 7 must satisfy $\delta_A + \delta_B = \delta$. Again, without loss of generality, we may assume that $\pi_A(\tau_0) > \pi_B(\tau_0)$, which implies that the greedy choice will increase w_A : $w_A(\tau_1) = w_A(\tau_0) + \delta + \frac{\epsilon}{2}$. The proof of stage 1 of Theorem 2 showed that

$w_A(\tau_0) < w_{eA}$, giving $w_{eA} = w_A(\tau_0) + \delta_A$. It follows then that

$$w_A(\tau_1) = w_A(\tau_0) + \delta + \frac{\epsilon}{2} = w_{eA} - \delta_A + \delta + \frac{\epsilon}{2}.$$

Similarly, $w_B(\tau_1) = w_{eB} + \delta_B - \delta - \frac{\epsilon}{2}$. Therefore, the $(2\delta + \epsilon)$ -greedy choice at time τ_0 moves $\mathbf{w}(\tau_1)$ further from \mathbf{w}_e by $|w_A(\tau_1) - w_{eA}| + |w_B(\tau_1) - w_{eB}| = |\delta - \delta_A + \epsilon/2| + |\delta_B - \delta + \epsilon/2| = \epsilon$. \square

DEFINITION 10: The *extreme greedy* policy for a non-loyal miner is to allocate all hash rate entirely to the greedy choice.

COROLLARY 2: Let $\mathcal{W}(M_{A^*})$ and $\mathcal{W}(M_{B^*})$ be the hash weights of miners loyal to coins A and B , respectively, and define

$$\mathbf{w}_1 = (1 - \mathcal{W}(M_{B^*}), \mathcal{W}(M_{B^*})) \frac{1}{H} \text{ and } \mathbf{w}_2 = (\mathcal{W}(M_{A^*}), 1 - \mathcal{W}(M_{A^*})) \frac{1}{H}.$$

Suppose equilibrium allocation \mathbf{w}_e is such that $w_{2A} < w_{eA} < w_{1A}$ and $w_{1B} < w_{eB} < w_{2B}$, and suppose further that reward ratio $V_A(\tau)/V_B(\tau)$ is constant in τ . Then for any choice of SAAs, non-loyal miners following the extreme greedy policy will result in hash rate fluctuations that oscillate between \mathbf{w}_1 and \mathbf{w}_2 . SAAs that come to rest faster will result in higher frequency oscillations.

PROOF: Without loss of generality, we can assume that at time τ_0 the greedy choice is to allocate all hash rate to chain A , which implies that $\mathbf{w}(\tau_0) = \mathbf{w}_1$. Now assume that both SAAs have come to rest at time τ_1 (if $\mathcal{W}(M_{B^*}) = 0$, then the SAA for chain B will not have had an opportunity to run because it has hash rate zero, but we nominally regard this as being at rest). It will suffice to show that the greedy choice at time τ_1 is to shift allocation to coin B . Suppose, for the purpose of contradiction, that the greedy choice at time τ_1 is to maintain maximum allocation to coin A . In that case, according to Theorem 2, there must exist some $\epsilon > 0$ such that $|w_{1A} + \frac{\epsilon}{2} - w_{eA}| < |w_{1A} - w_{eA}|$. But this is not possible because, by assumption $w_{1A} > w_{eA}$, so the greedy choice at time τ_1 must instead be to shift allocation to coin B , i.e. $\mathbf{w}(\tau_1) = \mathbf{w}_2$. Notice that SAAs that come to rest faster will realize faster fluctuations in the extreme greedy choice, and will therefore result in higher frequency oscillations between extreme allocations. \square

4 Beyond PoW

Fundamentally, the results of Section 3 tie the aggregate relative security of a blockchain to the value of reward given to those who provide security (i.e., PoW). PoW can be seen as proof-of-opportunity-cost for miners, who sacrifice energy and CPU cycles in return for the opportunity to gain native coins and a *vote* on the next block. The HAR measures fiat value per unit of opportunity cost.

And the SAA is simply a means of regulating this value so as to achieve to the desired emission of the native currency.

We can generalize PoW concepts as follows. Each blockchain defines a *cost function* \mathcal{C}_X with which it maps a unit of *native cost* to some quantity of native coin X . In PoW, native cost for chain X is the execution of a single hash using algorithm W_X . Define a *proof of cost* (PoC) voting system as one that allocates votes and native reward to participants proportional to their demonstrated cost. Furthermore, define a *cost adjustment algorithm* (CAA) as an algorithm that adjusts cost function \mathcal{C}_X so as to achieve a desired distribution of coin X over the short-to-medium-term. Total cost per second, H_X , on chain X is the amount of cost levied collectively against all participants in a single second. Regularized hash rate, \mathcal{H}_X is interpreted as the total cost per second on chain X , denominated in units chain A cost. The cost-adjusted-reward (CAR) is the fiat value of reward per unit of regularized cost. Some proof-of-stake (PoS) systems meet the criteria of a PoC voting system, and therefore, there exists the potential for an equilibrium to form relative to a PoW blockchain.

Public blockchains produce blocks as the result of a voting process, where votes are awarded to participants proportional to their opportunity cost. In PoW systems, the set of participants is entirely *open*: anyone with access to hardware capable of running the PoW algorithm can vote. But in PoS system, the set of participants is *restricted*: only those holding native coins can vote. Moreover, most PoS systems make a distinction between *active validators* who actively stake coins and simple coin *owners*. The former set can vote, while the latter cannot. Blocks are produced in *validation rounds*. Delegated PoS or DPoS blockchains are somewhat different still; coin holders vote for *delegates* and it is the delegates that create blocks using an alternative form of consensus such as Byzantine Fault Tolerance [18]. Another difference is that, instead of fixing the number of coins c comprising the block reward, PoS blockchains tend to define c as a function of the number of coins staked by active validators.

4.1 Basic PoS Equilibria

Consider PoS blockchain X . At time τ , there are k_X coins staked on chain X . The total reward for a single validation round is c_X , and each round lasts T_X seconds. Thus, total reward value is given by $V_X = c_X P_X$, where P_X is the fiat value of coin X . The total opportunity cost during a validation round, \mathcal{H}_X , is equal to $r k_X T_X P_X$, where r is the risk-free rate of return for investing 1 unit of fiat for 1 second. In words, \mathcal{H}_X measures the amount of fiat that could be earned by exchanging quantity k_X coins X for a so-called risk-free asset such as the 1-year US Treasury Note. Because the *native* unit of cost for \mathcal{H}_X is fiat, $\mathcal{S}_X = 1$, and $\mathcal{H}_X \mathcal{S}_A = H_X$. The CAR is given by $\pi_X = \frac{V_X}{H_X}$.

EXAMPLE 1: NEO is a DPoS blockchain [6]. There are two native coins on the chain: NEO and GAS. Holding NEO affords the bearer two privileges: the right to vote for delegates and access to a stream of GAS. Exactly 100e6 NEO coins exist; initially 50e6 were distributed during a crowd sale and the

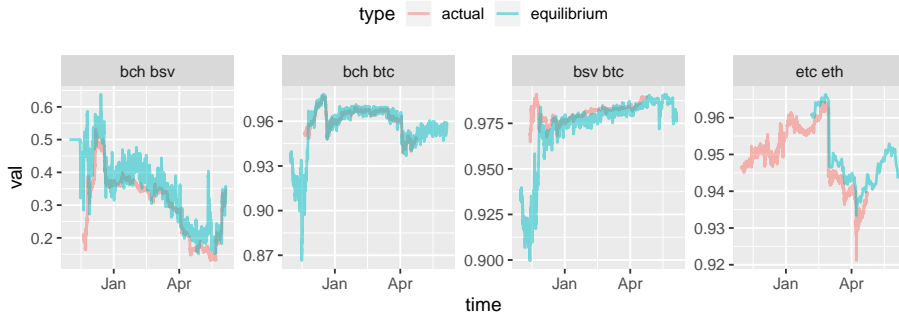


Figure 1: Actual hash rate allocation between various pairs of cryptocurrencies (red) juxtaposed with the equilibrium allocation (blue). The plots show strong agreement between the actual allocation and the allocation predicted by the equilibrium, the latter of which is based entirely on expected block times and coinbase values. The data ranges from December 1, 2018 until June 1, 2019.

remaining 50e6 were reserved by the *NEO council* to be used in the future to pay for development. GAS is awarded to NEO holders every validation round (occurring roughly once every T_{GAS} seconds) according to their percentage of the total available NEO. Initially, 8 GAS per round were awarded total, i.e. $c_{\text{GAS}} = 8$. Every 2e6 validation rounds (roughly 1 year), c_{GAS} is reduced by 1 GAS⁷. As of July 27, 2019, more than 4e6 blocks have been mined⁸, therefore $c_{\text{GAS}} = 6$. With these definitions, we can directly compare the security of the NEO blockchain to that of an arbitrary PoW blockchain using the framework from Section 3.

5 Evaluation

In this section, we validate the theoretical results from Section 3. Recall that the *actual* resource allocation between two blockchains is given by $\mathbf{w} = (K_A, K_B)$, where K_X is the relative security of chain $X \in \{A, B\}$ (see Definition 3). When chains A and B employ the same PoW algorithm W , \mathbf{w} is simply the fraction of aggregate hash rate for algorithm W applied to each of the chains. (Note that by *aggregate* we mean between chains A and B only, so $|\mathbf{w}| = 1$.) We first show that the equilibrium point \mathbf{w}_e , described by Theorem 1, closely matches the actual allocation \mathbf{w} for real historical blockchain data. We then show results from a block mining simulation that corroborate Theorem 2 and Corollary 2.

⁷<https://docs.neo.org/docs/en-us/basic/whitepaper.html>

⁸<https://neotracker.io>

5.1 Historical Convergence to Equilibrium

We collected historical data for several of the largest PoW blockchains by market cap including Bitcoin (BTC), Bitcoin Cash (BCH), Bitcoin Satoshi Vision (BSV), Ethereum (ETH), and Ethereum Classic (ETC). Included in the dataset were hourly prices for ETH, ETC, and BSV from the Bitfinex exchange⁹ and for BTC and BCH from the Binance exchange¹⁰. Difficulty data were collected for each block from Blockchair¹¹ with the exception of ETC, which was collected hourly from Coinwarz¹². From the difficulty, we were able to extract the approximate hash rate for each of the blockchains. Figure 2 plots the actual hash rate allocation \mathbf{w} in red for various pairs of blockchains (one pair per facet) along with the equilibrium allocation \mathbf{w}_e , which is plotted in blue. Agreement between the two curves indicates that there was an observed convergence to equilibrium as predicted by Theorem 2. The plots generally show strong agreement except for times when there were well-documented macro-level disturbances. For example, BSV hard-forked from BCH in November, 2018¹³, and we can see corresponding divergences from the equilibrium at this time. Also, a bug was exploited in the BCH ABC client during a hard fork upgrade in May, 2019, which caused a delay in block production and a chain reorganization¹⁴. Corresponding to this event, we again see divergence from the equilibrium.

There are two reasons why allocations might diverge from equilibrium at these times. First, the equilibrium defined by Theorem 1 assumes that blocks arrive exactly at their targeted times (every 10 minutes for BCH and BSV). However, during the events discussed above, block times were significantly slower than 10 minutes for a period of time, which means the plotted equilibrium is not quite accurate. Second, prices tend to fluctuate wildly during hard-forks and when bugs are encountered. (Indeed, there was no trading of BCH or BSV for several days on most exchanges during the November, 2018 hard-fork.) At those times it is difficult to correctly formulate an equilibrium with inaccurate price data, and also it is possible that miners will cease to mine greedily in order to ensure a given chain continues to produce blocks.

5.2 Convergence to Equilibrium in Simulation

We implemented a block mining simulation where miners were given the choice between chain A or B . For simplicity, it was assumed that both chains used the same PoW algorithm, had the same target inter-block time, and issued the same number of coins per block. The ratio of prices was initially 0.5, but we allowed it to vary according to a random walk with mean 0 and standard deviation 5e-3. We simulated 15 months of block generation total, but discarded the first 90

⁹<https://www.bitfinex.com>

¹⁰<https://www.binance.com>

¹¹<https://blockchair.com>

¹²<https://coinwarz.com>

¹³https://en.wikipedia.org/wiki/Bitcoin_Cash#November_2018_split

¹⁴<https://cointelegraph.com/news/bitcoin-cash-experiences-bug-during-scheduled-hard-fork-upgrade>

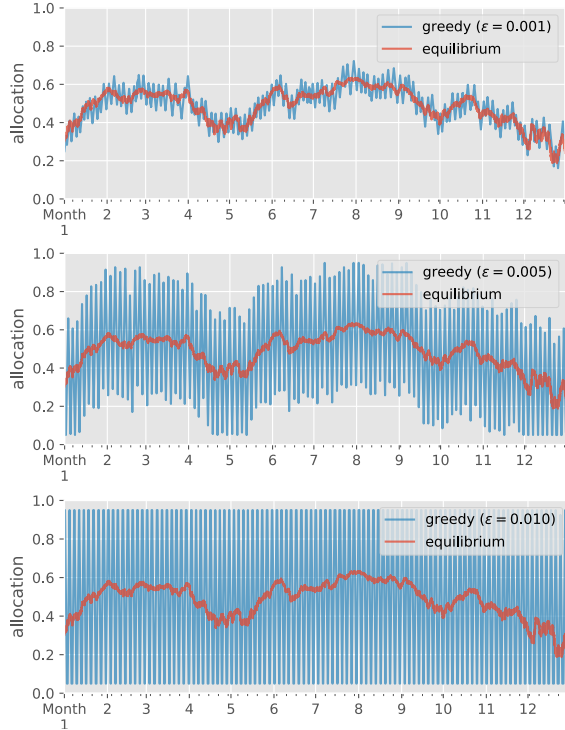


Figure 2: Results from 3 block mining simulation runs where miners chose between chains A and B . Each curve shows the equilibrium (red) and actual (blue) allocations to chain B . Each facet varies ϵ and 90% of miners follow the ϵ -greedy policy. Of the remaining miners, 5% are loyal to chain A and 5% are loyal to chain B . The plots show close adherence to the equilibrium for small ϵ and wild oscillations between extreme allocations as ϵ increases.

days to ensure the system had reached a steady state. Both chains A and B were assumed to use the difficulty adjustment algorithm (DAA) of Bitcoin Cash (BCH) [26], which uses a rolling average of the ratio of chain work to block time over the last 144 blocks (roughly 1 day). For each run of the simulation, we assumed that 5% of the miners were loyal to chain A , 5% were loyal to chain B , and the remaining 90% followed the ϵ -greedy strategy where ϵ was allowed to vary between runs.

Figure 2 shows the results of a single simulation run for each choice of $\epsilon \in \{1e-3, 5e-3, 1e-2\}$. The plots show the second component of the equilibrium vector (w_{eB}) in red, juxtaposed with the actual aggregate allocation to chain B in blue. As predicted by Theorem 2, sufficiently small ϵ (top facet) ensures convergence to the equilibrium. On the other hand, larger choices for ϵ result in divergence from the equilibrium (lower two facets), as predicted by Corollary 1. Moreover, as predicted by Corollary 2, the divergence from the equilibrium results in oscillations between the extremes defined by the fraction of loyal miners: $\mathcal{W}(M_{B^*})$ at one extreme and $1 - \mathcal{W}(M_{A^*})$ at the other.

6 Applications

6.1 Trustless Price-Ratio Oracle

Price feeds are a fundamental tool for many popular smart contract applications including prediction market Augur [1], stable coin issuer MakerDAO [4], hedge fund Numerai [7], and loan initiator Dharma [2]. Existing price feed solutions range from crowd-sourced [3,28] to trusted/whitelisted sources [5,8]. In this section, we present an application of the allocation equilibrium presented in Section 3 that delivers an estimate of the fiat *price ratio* of the coins native to blockchains A and B who share the same PoW algorithm.

We now describe how chain A can implement a price ratio oracle, but it should be noted that chain B could also do the same. Smart contract **Oracle** runs on chain A and returns an estimate of the price ratio P_B/P_A when the two chains are each at a given block height. Essentially, **Oracle** runs a light client for blockchain B , which contains all the headers since the chain’s genesis block. There are just two public methods exposed: **Update**(h_B) and **Query**(b_A, b_B). Method **Update**(h_B) allows any user or other contract to update the chain of headers with a new header h_B having the following properties: (i) the block height of h_B is exactly one greater than the previous header; (ii) the previous block hash of h_B points to the block hash of the previous header; and (iii) the PoW associated with the the hash of h_B meets the difficulty implied by earlier headers and chain B ’s protocol. If any of the conditions are not met, then it returns an error. Method **Query**(b_A, b_B) returns an estimate of the price ratio P_B/P_A at the time when chain A was at block height b_A and chain B was at height b_B . If either (i) the header at block height b_B is unknown to **Oracle** or (ii) the block on chain A at height b_A has not yet been mined, then an error is returned. Any party interested in maintaining the validity of the oracle will be sure to quickly run **Update**(h_B) for all new headers h_B for chain B .

We assume that **Oracle** will have native access to the header of the current block on chain A , h_A . Using headers h_A and h_B , **Oracle** will estimate P_B/P_A using Definition 1 and Theorem 1. Let $H(h_X)$ be an estimate of the hash rate for chain X derived from header h_X . On Bitcoin-like blockchains, the hash rate is simply $2^{32}/D$ where D is the difficulty, which is included in the block header. Hash rate can be extracted via similar transformations on other blockchains. From Definition 3, we have that

$$w_A \approx \frac{H(h_A)}{H(h_A) + H(h_B)},$$

where w_A denotes the fraction of total hash rate H shared between chains A and B that is devoted to chain A . According to Theorem 1,

$$w_A = \frac{T_B R}{T_B R - T_A R + T_A}$$

at equilibrium, where T_A and T_B are expected block times for chains A and B , $R = \frac{V_A}{V_A + V_B}$, $V_X = c_X P_x$, and c_X is the number of coins rewarded per block on

chain $X \in \{A, B\}$. Therefore,

$$\frac{H(h_A)}{H(h_A)+H(h_B)} \approx \frac{T_B R}{T_B R - T_A R + T_A} \Rightarrow$$

$$\frac{P_B}{P_A} \approx \frac{c_A}{c_B T_A} \left(\frac{T_B (H(h_A) + H(h_B))}{H(h_A)} - T_B + T_A \right) - \frac{c_A}{c_B}.$$

Figure 2 demonstrates that the equilibrium (shown in blue) typically agrees strongly with the security implied by the relative hash rate (shown in red). For this reason, we expect that price-ratio predictions will often be quite good. Of course an approximation of this sort is likely never to be as good as a centralized price feed. Thus, we envision the consumers of `Oracle` to be users or smart contracts that require a fully decentralized oracle, or perhaps require a safety check on the trust placed in a centralized price feed.

EXAMPLE 2: Suppose that we wish to introduce fully decentralized *futures contracts* to blockchain A intended to be negotiated between two parties: guarantor \mathcal{G} and beneficiary \mathcal{B} . To do so, a smart contract can be developed that leverages `Oracle`. Each futures contract, or *future* transfers from guarantor to beneficiary a quantity of coins A equivalent to the value of a quantity of coin B at a future date. Specifically, a future issued at the time when chains A and B are at block heights b_A and b_B , allows the beneficiary to trade the contract to the guarantor for a quantity of coins A equivalent to 1 coin B on the *expiry date*. We define expiry as the latter of block heights b'_A and b'_B , anticipated to be some time in the future (for example 90 days). Contract `Future` implements four methods: `Deposit(a)`, `Recover(a)`, `Issue(b_A, b_B, b'_A, b'_B, a)`, and `Redeem(b'_A, b'_B)`. `Deposit` is signed by \mathcal{G} ; it deposits quantity a of coin A into `Future`. This will be used to pay \mathcal{B} at expiry. Prior to calling `Issue`, the funds can be redeemed by \mathcal{G} if he signs `Recover`. The call to `Issue` must be signed by both \mathcal{G} and \mathcal{B} ; signifying that they agree to the initial and expiry block times and fee of a coins, which is paid by \mathcal{B} and immediately transferred to an account owned by \mathcal{G} . Once headers h'_A and h'_B at height b'_A and b'_B have been generated, \mathcal{B} first calls `Update(h'_B)` on `Oracle` and then signs `Redeem`. In response to this method, contract `Future` deposits into an account controlled by \mathcal{B} a quantity of A coins that are equivalent to the value of 1 coin B as determined by calling `Query(b'_A, b'_B)` on contract `Oracle`.

6.2 Increasing Security

Consider two blockchains A and B that have the same PoW algorithm W and target inter-block time T . BTC and BCH constitute an example where W is the SHA256 algorithm and $T = 600$ seconds. Recall from Section 3 that P_A and P_B are the fiat coin values for A and B , respectively, and that, ignoring fees, coinbase value $V_X = c_X P_X$, where c_X denotes the number of coins issued per block on chain X . Finally, recall that $R = V_A / (V_A + V_B)$. Suppose that coin B is consistently less valuable than coin A ; i.e., $P_B / P_A = \alpha$ for some $\alpha < 1$. If $c_A = c_B$, then because they share the same PoW algorithm and inter-block time,

Theorem 1 predicts that the equilibrium allocation will be

$$\begin{aligned}
\mathbf{w}_e &= (R, 1 - R) \\
&= \left(\frac{V_A}{V_A + V_B}, \frac{V_B}{V_A + V_B} \right) \\
&= \left(\frac{P_A}{P_A + P_B}, \frac{P_B}{P_A + P_B} \right) \\
&= \frac{1}{1 + \alpha} (1, \alpha).
\end{aligned} \tag{5}$$

Thus, chain A will tend toward having $1/\alpha$ more hash rate than chain B , which constitutes lower security for chain B . This can lead to a negative feedback loop where lower security leads to lower coin price, which in turn leads to even lower security. One way to break this loop is for chain B to simply increase the issuance per block, c_B . Of course P_B will be reduced in value as a result, but somewhat surprisingly, the net effect is not necessarily zero sum.

The *market capitalization* (CAP) for coin X , m_X , is a measure of the aggregate future value of the corresponding blockchain in the same sense that the CAP of an equity is a measure of the capacity for the underlying corporation to deliver returns to investors in the future. We do not attempt to economically justify the CAP of blockchain coins, but rather we treat the CAP as an objective measure of overall blockchain value that is emergent from the coin market. At time τ , CAP is related to circulating coin I_X and coin price by $m_X = I_X(\tau)P_x(\tau)$. Because no new value is generated for a blockchain by circulating more coin, an increase in issuance alone should not increase the CAP. But since more coin has been issued, the fiat price of each coin must decrease. Therefore, an increase of ΔI coins for chain B during time $\Delta\tau$ must decrease the value of coin B by

$$\begin{aligned}
\Delta P_B &= m_B \left(\frac{1}{I_B(\tau) + \Delta I} - \frac{1}{I_B(\tau)} \right) \\
&= \frac{m_B}{I_B(\tau) + \Delta I} \left(1 - \frac{I_B(\tau) + \Delta I}{I_B(\tau)} \right) \\
&= - \frac{m_B}{I_B(\tau) + \Delta I} \frac{\Delta I}{I_B(\tau)} \\
&= - \frac{P_B(\tau) \Delta I}{I_B(\tau) + \Delta I}
\end{aligned} \tag{6}$$

Suppose that until time τ , chains A and B have each issued $I_A(\tau) = I_B(\tau) = I$ total coins and have each issued the same number of new coins per block: $c_A(\tau) = c_B(\tau) = c$. Because $P_B/P_A = \alpha$, we also have $m_B/m_A = \alpha$. At time τ , chain B decides to increase its issuance per block by factor $k > 1$ for a period of time $\Delta\tau$, i.e. $c_B(\tau') = kc$ while $c_A(\tau') = c$ for $\tau' \in [\tau, \tau + \Delta\tau]$. At time $\tau + \Delta\tau$, βI total A coins and γI total B coins will have been issued, where $\gamma > \beta > 1$. As a result, according to Equation 6,

$$\begin{aligned}
P_A(\tau + \Delta\tau) &= P_A(\tau) - \frac{(\beta - 1)IP_A(\tau)}{\beta I} \\
&= P_A(\tau) \left(1 - \frac{\beta - 1}{\beta} \right) \\
&= \frac{1}{\beta} P_A(\tau).
\end{aligned}$$

while

$$\begin{aligned}
P_B(\tau + \Delta\tau) &= P_B(\tau) - \frac{(\gamma-1)IP_B(\tau)}{\gamma I} \\
&= P_B(\tau) \left(1 - \frac{\gamma-1}{\gamma}\right) \\
&= \alpha P_A(\tau) \left(1 - \frac{\gamma-1}{\gamma}\right) \\
&= \frac{\alpha}{\gamma} P_A(\tau).
\end{aligned}$$

At time $\tau + \Delta\tau$, noting that $R = P_A/(P_A + kP_B)$, the equilibrium allocation becomes

$$\begin{aligned}
\mathbf{w}_e(\tau + \Delta\tau) &= (R, 1 - R) \\
&= \left(\frac{P_A}{P_A + kP_B}, 1 - \frac{P_A}{P_A + kP_B}\right) \\
&= \left(\frac{\frac{1}{\beta}}{\frac{1}{\beta} + \frac{k\alpha}{\gamma}}, 1 - \frac{\frac{1}{\beta}}{\frac{1}{\beta} + \frac{k\alpha}{\gamma}}\right) \\
&= \left(\frac{1}{1 + \frac{k\alpha\beta}{\gamma}}, \frac{\alpha}{\alpha + \frac{k\beta}{\gamma}}\right).
\end{aligned} \tag{7}$$

Comparing the new equilibrium in Equation 7 to the equilibrium prior to the increase in issuance given by Equation 5, we see that chain B will have increased its share of the hash rate so long as $\gamma/(k\beta) < 1$.

EXAMPLE 3: In 2020, both BTC and BCH are expected to have mined 18.375e6 total coins and will reduce their coinbase rewards from $c = 12.5$ down to $c = 6.25$ coins in what is called a *halving* event¹⁵. At that point, both chains will have completed fraction 0.875 of their total planned issuance of 21e6 coins. Suppose that at the time of the planned halving, BCH decides instead to continue issuing 12.5 coins per block for an additional 4 years, and then resume with the convention of halving the coins every four years after that. This practice ensures that BCH will always award twice as many coins as BTC (that is until both coins eventually cut issuance to 0), i.e. $k = 2$. BCH will also emit approximately 2e6 additional coins, which amounts to slightly less than 10% more than the originally planned issuance. Based on the new issuance, at all times $\beta/\gamma > 18.375e6/(2e6 + 18.375e6) > 0.9$. Using Equation 7, we find that in the worst-case, this increases the allocation for BCH to $\alpha/(\alpha + 0.55)$ of the total hash rate from $\alpha/(\alpha + 1)$ before issuance was increased. On July 2, 2019, BTC traded for approximately 11,000 USD and BCH traded for roughly 400 USD¹⁶. This implies that $\alpha \approx 0.036$, which means that the current equilibrium hash rate allocation for BCH is approximately 0.034 of the total, but it would increase to 0.061 of the total after extending its coin issuance, nearly a two-fold increase.

EXAMPLE 4: Expanding on Example 3, we can imagine a blockchain B that tunes its issuance in the extreme to achieve a chosen hash rate equilibrium relative to another chain A . Suppose that $P_B/P_A = \alpha < 1$, but chain B chooses $c_B = c_A/\alpha$ so that $V_B/V_A = 1$. Theorem 1 predicts that as long as $P_B > 0$, $\mathbf{w}_e^* = (0.5, 0.5)$. Of course, $c_B > c_A$, so B coins are issued more rapidly than

¹⁵https://en.bitcoin.it/wiki/Controlled_supply

¹⁶<https://coinmarketcap.com>

A coins. This means that chain B will have achieved **parity in security** with chain A at the expense of more rapidly devaluing its coin relative to coin A . A natural question is, will the market reward chain B for this increase in security with an increase in market cap? Note that the mechanism proposed here is more complicated than the one proposed in Example 3 because here we require that the chain B protocol has knowledge of the price ratio P_B/P_A . One way that this can be accomplished is for chain B to implement the `Oracle` contract as described in Section 6.1.

6.3 Cost of Loyal Mining

In this section, we attempt to quantify the cost for miners who are loyal to mining a single chain B when there exists an option to mine either chain A or B . At times, this choice can be profitable, but at other times, there exists an opportunity cost in the form of higher profits associated with mining on chain A . Equation 2 gives the hash adjusted reward or HAR vector $\boldsymbol{\pi}$ associated with the allocation vector $\boldsymbol{w} = (xR, y(1 - R))$, where x and y are arbitrary positive constants such that $xR + y(1 - R) = 1$. Vector $\boldsymbol{\pi}$ indicates the expected reward per hash performed on each blockchain. Theorem 1 identifies a unique choice for x and y that gives an *equilibrium allocation*, \boldsymbol{w}_e , which is the only point where the HAR values for each chain are equal. Finally, Theorem 2 establishes that when the allocation to chain B is less than the equilibrium; i.e. $w_B < w_{eB}$, the greedy choice is to increase allocation to B and therefore the HAR value is higher on chain B than on chain A . Thus, in this regime a miner loyal to chain B will profit. However, once $w_B > w_{eB}$, the opposite is true.

For miner m , define hash rate vector $\boldsymbol{\phi}$ as the allocation for m expressed as a fraction of total hash rate H ; e.g. $H\boldsymbol{\phi}_X$ gives the hash rate for m on chain X . The *utility vector* for m is defined as

$$\boldsymbol{U}(\boldsymbol{\phi}, \boldsymbol{\pi}) = H\boldsymbol{\phi} \cdot \boldsymbol{\pi}, \quad (8)$$

Component $\boldsymbol{U}(\boldsymbol{\phi}, \boldsymbol{\pi})_X$ gives the total fiat value captured by m mining for one second on chain X . Note that $\boldsymbol{U}(\boldsymbol{w}, \boldsymbol{\pi})$ gives the aggregate utility for all miners collectively. The *opportunity cost* to m for shifting from allocation $\boldsymbol{\phi}$ to $\boldsymbol{\phi}'$ is given by

$$\kappa(\boldsymbol{\phi}, \boldsymbol{\pi}; \boldsymbol{\phi}', \boldsymbol{\pi}') = \boldsymbol{U}(\boldsymbol{\phi}, \boldsymbol{\pi}) - \boldsymbol{U}(\boldsymbol{\phi}', \boldsymbol{\pi}'). \quad (9)$$

6.3.1 Utility between highly similar blockchains

In the special case where $T_A = T_B = T$, $\mathcal{S}_A = \mathcal{S}_B$, and $c_A = c_B = c$, the utility for miner m is given by

$$\boldsymbol{U}(\boldsymbol{\phi}, \boldsymbol{\pi}) = \frac{c(P_A + P_B)}{T} \boldsymbol{\phi} \cdot \left(\frac{1}{x}, \frac{1}{y} \right).$$

At equilibrium, $x = y = 1$ and utility becomes

$$\boldsymbol{U}(\boldsymbol{\phi})_e = \frac{c\phi(P_A + P_B)}{T},$$

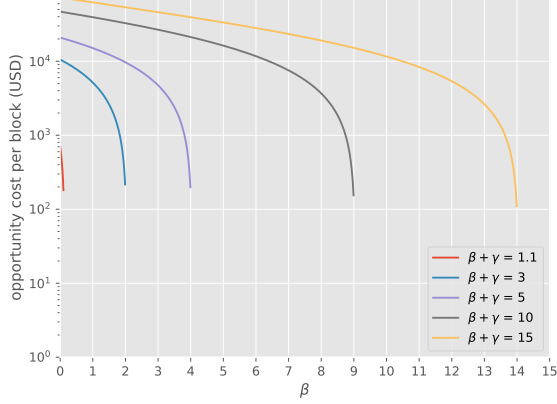


Figure 3: Opportunity cost in USD (Equation 11) to miners from BTC who divert hash rate to BCH in order to cause blockchain reorganization. Attacker diverts multiple γ times the equilibrium BCH hash rate (H_{BCH}), where $\gamma > 1$, in order to create a fork of the BCH chain. Attacker also leaves βH_{BCH} to mine on BTC. For hash rate $\beta + \gamma$, opportunity cost is lowest as γ approaches 1, but this requires much longer to reorganize the BCH chain. $\beta + \gamma = 15$ corresponds to roughly 55% of the total hash rate.

where $\phi = |\phi|$. Therefore, the opportunity cost to m for mining with allocation vector ϕ' is given by

$$\kappa(\phi', x', y')_e = \frac{c(P_A + P_B)}{T} \left(\phi - \phi' \cdot \left(\frac{1}{x'}, \frac{1}{y'} \right) \right). \quad (10)$$

EXAMPLE 5: Consider blockchains A and B that are similar in the sense of Section 6.3.1, and assume that $P_B/P_A = \alpha$ such that $w_e = \left(\frac{1}{1+\alpha}, \frac{\alpha}{1+\alpha} \right)$. Assume further that the allocation is initially at equilibrium, i.e. $w = w_e$. Now suppose that a group of miners m loyal to coin B , and having total hash rate $\phi = kw_eB$, wish to increase chain B 's share of the hash rate by a factor k , such that $w'_B = kw_eB$. Since w'_B exceeds the equilibrium allocation to chain B , Theorem 2 shows that greedy miners will abandon chain B and therefore **only** loyal miners will mine on chain B , i.e. $w'_B = \phi'_B$. By definition, $R = 1/(1 + \alpha)$, and from Lemma 1, we have that $Rx' = 1 - \frac{k\alpha}{1+\alpha}$ and $y'(1 - R) = \frac{k\alpha}{1+\alpha}$. Hence, according to Equation 10, we find the opportunity cost per block to be

$$\begin{aligned} T\kappa(\phi', x', y')_e &= c(P_A + P_A) \left(\phi - \phi' \cdot \left(\frac{1}{x'}, \frac{1}{y'} \right) \right) \\ &= c(P_A + P_B) \left(\frac{k\alpha}{1+\alpha} - \left(0, \frac{k\alpha}{1+\alpha} \right) \cdot \left(\frac{R(1+\alpha)}{1-\alpha(k-1)}, \frac{(1-R)(1+\alpha)}{k\alpha} \right) \right) \\ &= c(P_A + P_B) \left(\frac{k\alpha}{1+\alpha} + R - 1 \right) \\ &= c(P_A + P_B) \left(\frac{(k-1)\alpha}{1+\alpha} \right) \\ &= c(P_A + P_B)\alpha R(k-1) \\ &= c(k-1)P_B. \end{aligned}$$

In words, the opportunity cost for miners m to increase the hash rate of chain B by a factor k beyond the equilibrium allocation for 1 block is exactly equal to $k - 1$ times the expected coinbase reward from chain B .

EXAMPLE 6: Continuing with Example 5, suppose that a group of miners m from chain A , having aggregate hash weight $(\beta + \gamma)\frac{\alpha}{1+\alpha}$ where $\gamma > 1$ and $\beta + \gamma < \frac{1}{\alpha}$, conspire to *reorganize*, i.e. orphan, the last z blocks on chain B . They will do this by diverting hash rate $\gamma\frac{\alpha}{1+\alpha}$ from chain A to a **fork** of chain B . Thus, the existing hash rate on chain B , $\frac{\alpha}{1+\alpha}$, will be lost entirely. For simplicity, we assume that both DAAs come to rest immediately (which incurs negligible error when z is large). It follows then that, during the attack, the new hash allocation will be

$$\mathbf{w}' = (1 + \alpha) \left(\frac{1 - \alpha\gamma}{1 + \alpha}, \frac{\alpha\gamma}{1 + \alpha} \right) = (1 - \alpha\gamma, \alpha\gamma).$$

with fraction $\frac{\alpha\beta}{1+\alpha}/(\frac{1}{1+\alpha} - \frac{\alpha\gamma}{1+\alpha}) = \frac{\alpha\beta}{1-\alpha\gamma}$ of w'_A and the entirety of w'_B being controlled by miners m . In the parlance of Section 6.3.1, we have $\phi' = (\alpha\beta, \alpha\gamma)$, $\phi = \alpha(\beta + \gamma)$, $x' = (1 + \alpha)(1 - \alpha\gamma)$, and $y' = \gamma(1 + \alpha)$. Therefore, using the same reasoning as in Example 5, the opportunity cost for miners m is equal to

$$\begin{aligned} T\kappa(\phi', x', y')_e &= c(P_A + P_B) \left(\phi - \phi' \cdot \left(\frac{1}{x'}, \frac{1}{y'} \right) \right) \\ &= c(P_A + P_B) \left(\alpha(\beta + \gamma) - (\alpha\beta, \alpha\gamma) \cdot \left(\frac{1}{(1+\alpha)(1-\alpha\gamma)}, \frac{1}{\gamma(1+\alpha)} \right) \right) \\ &= c(P_A + P_B) \left(\alpha(\beta + \gamma) - \frac{\alpha\beta}{(1+\alpha)(1-\alpha\gamma)} - \frac{\alpha}{1+\alpha} \right). \end{aligned} \tag{11}$$

Figure 3 shows the opportunity cost to BTC miners who attempt to carry out a reorganization attack on the BCH chain. We assume here that $P_{\text{BCH}} = 400$, $P_{\text{BTC}} = 11000$, and $\alpha = P_{\text{BCH}}/P_{\text{BTC}} \approx 0.36$ as was the case on July 2, 2019¹⁷. Prior to attack, BCH has fraction $\alpha/(1 + \alpha)$ of the hash rate. The attacker diverts $\gamma\alpha/(1 + \alpha)$, $\gamma > 1$, hash rate to a fork of the BCH chain and leaves fraction $\beta\alpha/(1 + \alpha)$ to mine on BTC. The plot shows that opportunity cost is lowest as γ approaches 1, but this also means that the attacker has roughly the same hash rate as honest miners on the other BCH fork. Thus, a cheaper attack will take much longer to reorganize the BCH chain for fixed reorganization depth z .

7 Discussion

7.1 Greedy is obvious, but why cautious?

Section 5 showed empirically that the hash rate allocations among several pairs of the largest blockchain projects by market cap closely follow the equilibrium described in Theorem 1. There are exceptions, where the allocation diverges from equilibrium, but they tend to be short-lived and align closely with events like hard forks. Also in that section, similar results were observed in simulation when the majority of miners follow an ϵ -greedy policy (see Definition 8) for sufficiently small ϵ . This suggests that much of miner behavior can be explained by a preference for improving immediate reward, but not to an extreme. Specifically,

¹⁷<https://coinmarketcap.com>

the simulation also showed that for a choice of ϵ that is too large, allocations oscillate wildly, a phenomenon not typically observed in practice.

So if mining on a particular chain is currently more profitable than mining on another, why don't miners fully allocate to that chain, i.e. follow the extreme greedy policy? Prior works discussed previously in Section 2 provide possible explanations. Chatzigiannis et al. [11] suggested that miners incur less risk in the form of variance in block reward by mining simultaneously in a mixture of pools and across blockchains. Therefore, there exists incentive to mine at least partially on the less profitable chain in order to enjoy lower variance in payout. Most chains also impose a *cool-down period*¹⁸ for newly awarded coins during which they cannot be spent. Bissias et al. [10] argued that this imparts risk to the miner in the form of price volatility during the cool-down period. They showed that miners can minimize risk by allocating their hash rate to a mixed *portfolio* of blockchains. Thus, again, the extreme greedy policy may be inferior to a mixed strategy that reduces miner risk

7.2 Implications for Minority Hash Rate Chains

A major conclusion from Kwon et al. [17] is that minority hash rate blockchains such as Bitcoin Cash (BCH) might be doomed to fail due to a lack of genuine miner interest. Their reasoning is that, if there exists a loyal miner base devoted to BCH that exceeds the equilibrium allocation, then no profit seeking miners will also mine BCH. Thus, the loyal miners will be alone in propping up the blockchain. While we do not dispute the possibility of this scenario, it is also not clearly a likely outcome. First, Theorem 2 proves that for greedy but cautious miners, there exists a tendency to move toward equilibrium. In Section 5, we demonstrated empirically that this tendency is typically manifested in the real world. And at equilibrium, there exists no *preference* to mine one chain over the other. Thus, there is typically no need for loyal mining to maintain hash rate. Second, Example 5 shows that loyal mining beyond the equilibrium point incurs a cost linear in the value of the coinbase reward of the minority chain. Therefore, loyal miners who are actively propping up the hash rate of a blockchain are financially disincentivized from continuing this practice over the long-term, which will also tend to move hash rate allocation back to equilibrium.

The examples in Section 6.2 illustrate that the allocation equilibrium point itself is quite fluid, depending mainly on the value of the coinbase reward. As discussed in Section 2, this is a concept that was first suggested in abstract by Spiegelman et al. [27], and we have extended it by quantifying the change in equilibrium given a specific change in coinbase reward. The implication of these results is that minority hash rate blockchains can significantly increase their security relative to the majority hash rate blockchain by simply adjusting their coinbase reward. We further demonstrated that this increase in security can be accomplished for BCH without significantly devaluing the currency.

Finally, for minority hash rate blockchains, there exists a danger that miners

¹⁸<https://bitcoin.org/en/blockchain-guide#transaction-data>

from the majority hash rate chain will force a long reorganization of z previously confirmed blocks. In Example 6, we derived an expression for the opportunity cost to attackers from the heavier weight blockchain. For BCH versus BTC (Figure 3), every reorganization costs at least 100 USD per block. However, the lowest cost attacks only allow the attacker to match the hash rate on BCH. This means that the reorganization of many blocks will likely take a long time since it is required that the attacker mine $n + z + 1$ blocks in the time the honest miners mine n . Cost rises exponentially as the attacker increases hash rate beyond honest BCH miners. For example, if the attackers double the honest hash rate on BCH, then the opportunity cost jumps to at least 3,000 USD per block for any set of attackers with less than 50% of the total BTC + BCH hash rate.

8 Conclusion

In this paper, we have shown formally that a singular hash rate equilibrium arises for miners who split their hash rate among two blockchains assuming that the miners are both greedy and cautious. If they become overly greedy, then their hash rate will oscillate in the extreme between the two chains. Assuming an efficient market for buying and selling hash rate, the results also hold between two blockchains with different PoW algorithms, and even between PoW and PoS blockchains where hash rate is replaced by the opportunity cost associated with locking up stake in the PoS system. We demonstrated these theoretical results empirically using historical data from real world blockchains and data from a block mining simulator. Finally, we presented several applications including a trustless price-ratio oracle, enhanced security for minority hash rate blockchains, and quantification of loyal mining costs.

9 Acknowledgements

We would like to thank David Jensen and Akanksha Atrey for many thought provoking discussions, which helped us to focus our investigation. We would also like to thank Rainer Böhme for his insights in the discussions we had with him.

References

- [1] Augur. <https://www.augur.net>.
- [2] Dharma. <https://blog.dharma.io>.
- [3] Dutchx. <https://dutchx-rinkeby.d.exchange>.
- [4] MakerDAO. <https://makerdao.com>.
- [5] MakerDAO Price Feed. <https://developer.makerdao.com/feeds>.
- [6] Neo. <https://neo.org>.
- [7] Numerai. <https://numer.ai>.

- [8] Provable. <http://provable.xyz>.
- [9] ALTMAN, E., ALEXANDRE, R.-M., MENASCHÉ, D. S., DATAR, M., DHAMAL, S., AND TOUATI, C. Mining competition in a multi-cryptocurrency ecosystem at the network edge: A congestion game approach. <https://hal.inria.fr/hal-01906954>, October 2018.
- [10] BISSIAS, G., LEVINE, B., AND THIBODEAU, D. Using Economic Risk to Model Miner Hash Rate Allocation in Cryptocurrencies. In *Workshop on Cryptocurrencies and Blockchain Technology (CBT)* (2018).
- [11] CHATZIGIANNIS, P., BALDIMTSI, F., GRIVA, I., AND LI, J. Diversification Across Mining Pools: Optimal Mining Strategies under PoW. In *Workshop on the Economics of Information Security (WEIS)* (2019).
- [12] CONG, L. W., HE, Z., AND LI, J. Decentralized Mining in Centralized Pools. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3143724, February 2018.
- [13] EYAL, I., AND SIRER, E. G. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security* (2014), Springer, pp. 436–454.
- [14] GERVAIS, A., O. KARAME, G., WUST, K., GLYKANTZIS, V., RITZDORF, H., AND CAPKUN, S. On the Security and Performance of Proof of Work Blockchains. <https://eprint.iacr.org/2016/555>, 2016.
- [15] HAN, R., SUI, Z., YU, J., LIU, J., AND CHEN, S. Sucker punch makes you richer: Rethinking Proof-of-Work security model. <https://eprint.iacr.org/2019/752>, June 2019.
- [16] KIRÁLY, T., AND LOMOSCHITZ, L. Profitability of the coin-hopping strategy. <http://web.cs.elte.hu/egres/www/qp-18-03.html>, March 2018.
- [17] KWON, Y., KIM, H., SHIN, J., AND KIM, Y. Bitcoin vs. Bitcoin Cash: Coexistence or Downfall of Bitcoin Cash? <https://arxiv.org/abs/1902.11064>, February 2019.
- [18] LAMPORT, L., SHOSTAK, R., AND PEASE, M. The Byzantine generals problem. In *ACM Transactions on Programming Languages and Systems* (1982), vol. 4, pp. 382–401.
- [19] MA, J., GANS, J. S., AND TOURKY, R. Market Structure in Bitcoin Mining. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3103104, June 2019.
- [20] MESHKOV, D., CHEPURNOY, A., AND JANSEN, M. Revisiting Difficulty Control for Blockchain Systems. In *Cryptocurrencies and Blockchain Technology (CBT)* (2017).
- [21] MONDERER, D., AND SHAPLEY, L. S. Potential Games. In *Games and Economic Behavior* (1996), vol. 14, pp. 124–143.
- [22] NAKAMOTO, S. Bitcoin: A Peer-to-Peer Electronic Cash System, May 2009.
- [23] PRAT, J., AND WALTER, B. An Equilibrium Model of the Market for Bitcoin Mining. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3143410, February 2018.
- [24] SAI, A. R., BUCKLEY, J., AND LE GEAR, A. Assessing The Security Implication Of Bitcoin Exchange Rates, 2019.
- [25] SAPIRSHEIN, A., SOMPOLINSKY, Y., AND ZOHAR, A. Optimal Selfish Mining Strategies in Bitcoin. <https://arxiv.org/pdf/1507.06183.pdf>, July 2015.
- [26] SECHET, A. Implement simple moving average over work difficulty adjustment algorithm. <https://reviews.bitcoinabc.org/D601>, October 2017.

- [27] SPIEGELMAN, A., KEIDAR, I., AND TENNENHOLTZ, M. Game of Coins. <https://arxiv.org/abs/1805.08979>, May 2018.
- [28] WHGEORGE. Decentralized price oracle. <https://ethresear.ch/t/decentralized-price-oracle/1941>, May 2018.