Ignore the Extra Zeroes: Variance-Optimal Mining Pools

Tim Roughgarden[1][0000−0002−7163−8306] and Clara Shikhelman[2][0000−0002−0587−7181]

[1] Columbia University, New York, NY, 10027, USA
tim.roughgarden@gmail.com
[2] Chaincode Labs, New York, NY, 10017, USA
clara.shikhelman@gmail.com

Abstract. Mining pools decrease the variance in the income of cryptocurrency miners (compared to solo mining) by distributing rewards to participating miners according to the shares submitted over a period of time. The most common definition of a "share" is a proof-of-work for a difficulty level lower than that required for block authorization—for example, a hash with at least 65 leading zeroes (in binary) rather than at least 75.

The first contribution of this paper is to investigate more sophisticated approaches to pool reward distribution that use multiple classes of shares—for example, corresponding to differing numbers of leading zeroes—and assign different rewards to shares from different classes. What's the best way to use such finer-grained information, and how much can it help? We prove that the answer is *not at all*: using the additional information can only increase the variance in rewards experienced by every miner.

Our second contribution is to identify variance-optimal reward-sharing schemes. Here, we first prove that pay-per-share rewards simultaneously minimize the variance of all miners over all reward-sharing schemes with long-run rewards proportional to miners' hash rates. We then show that, if we impose natural restrictions including a no-deficit condition on reward-sharing schemes, then the pay-per-last-N-shares method is optimal.

Keywords: Blockchains · cryptocurrencies · mining pools · variance-minimization.

1 Introduction

In Bitcoin [13] and many other cryptocurrencies (Ethereum [3], for example), *miners* produce proofs-of-work to authorize blocks of transactions in exchange for rewards. A solo miner controlling a small fraction of the overall hashrate will receive no reward for long stretches of time (e.g., for a Bitcoin miner with 0.1% of the overall hashrate, for roughly a week on average). To spread payouts more evenly over time, many miners join *mining pools* in which multiple miners join forces and work to authorize a block in tandem.

When a mining pool successfully authorizes a block (e.g., finding a nonce so that the block hashes to a number with at least 75 leading zeroes in binary[3]), the reward collected by the pool owner must be distributed to the participating miners (perhaps less a commission) so that they continue to contribute. There are many ways to distribute rewards (as evident already from the early survey of Rosenfeld [16]); here, we isolate two of the design decisions involved.

Design decision #1: What information from miners should be the basis for their rewards?

Typically, rewards are based on the *shares* submitted by each miner over a period of time, where a "share" is a proof-of-work for a difficulty level lower than that required for block authorization (e.g., 65 leading zeroes instead of 75). This difficulty level is chosen to be low enough that a typical miner can produce shares reasonably frequently (thereby receiving a somewhat steady payout) but high enough that neither miners nor the pool are overwhelmed by the number of shares that must be communicated. There is no obvious reason to restrict designs to a simple uniform notion of shares, however. An example of a more sophisticated approach would be to use multiple classes of shares—for example, differing numbers of leading zeroes—and assign different rewards to shares from different classes.

Design decision #2: How should the information submitted by miners determine their rewards?

For example, with a single class of shares, two of the approaches common in practice are *pay-per-share (PPS)*, in which there is a fixed reward for each share (independent of any block authorization events); and *pay-per-last-N-shares (PPLNS)*, in which the reward associated with each successful block authorization is distributed equally among the most recently submitted N shares. Is there a good reason to prefer one of these approaches over the other? Is some other way of distributing rewards "better" than both of them?

> *The goal of this paper is to identify mining pool reward-sharing schemes that are "optimal" in a precise sense.*

1.1 Our Contributions

Model for identifying variance-optimal reward-sharing schemes. Given that the primary raison d'être of mining pools is to reduce the variance in miners' rewards [16], we focus on the objective of minimizing variance.[4] Our first contribution is the definition of a formal model that allows us to identify every

[3] Technically, in Bitcoin this is defined by finding a hash that is smaller than a number that gets adjusted over time. For ease of discussion we will continue to refer to the number of leading zeros.

[4] Rosenfeld [16] computed the variance of some specific reward-sharing schemes and briefly considered multi-class shares (in [16, §7.5]) but did not pursue optimality results. A discussion on variance minimization can also be found in [17]

reward-sharing scheme with a statistical estimator (of the miner hashrate distribution) and formally compare the variance properties of different schemes. We then use this model to investigate the two design decisions above. We focus on schemes that are *unbiased* in the sense that the long-run rewards to a miner are proportional to the fraction of the overall hashrate controlled by that miner (as is the case for all of the most popular reward-sharing schemes).

Single-class shares are optimal. It is intuitively clear that, at least in some scenarios, multi-class shares can lead to higher payoff variance than single-class shares. For example, consider a sequence of t consecutive messages, generated by the miners, that qualify for some type of reward (say, because each starts with at least 65 zeroes). In the extreme case in which there is only a single miner, with 100% of the hashrate, the variance of that miner's reward under standard (single-class) PPS in such a sequence is zero (as all t messages must have been generated by that miner, and each pays the same reward). With multi-class shares, by contrast, there would be positive variance in the miner's payoff across such sequences because of (say) the varying number of zeroes across different messages.

Our second contribution shows that variance degredation from multiple classes of shares is a fundamental phenomenon and not just an edge case: for every possible miner hashrate vector, every deviation from the dominant-in-practice single-class model can only increase the variance in rewards of *every* miner. For example, a version of PPLNS or PPS that conditions rewards on the number of leading zeroes in a hash (e.g., with a smaller reward for 65 zeroes and a larger reward for 75) would be worse for all miners than PPLNS or PPS (respectively) with all shares treated equally. As shown in Figure 1, the difference in variance between single-class and multi-class shares can be significant.

Pay-per-share is optimal. Our third contribution identifies a sense in which the pay-per-share method is variance-optimal: for every possible miner hashrate vector, it simultaneously minimizes the variance of all miners over all unbiased reward-sharing schemes. We stress that there is no a priori guarantee that a scheme with such a guarantee exists—conceivably, small miners would be better off under one scheme and larger miners under a different one. Our result shows that no trade-offs between different miners are necessary—from the perspective of variance-minimization, all miners prefer pay-per-share. We also provide a second statistical justification of the pay-per-share method by showing that it corresponds to the maximum likehood estimator for the miner hashrate distribution.

Pay-per-last-N-shares is optimal within a restricted class. One drawback of the PPS scheme is that, in the short run, it might be obligated to pay out rewards to miners that exceed the rewards that the pool has actually earned to date. (Whereas, in the long run, the reward per share is set so that the pool can almost surely cover its obligations with its earned rewards.) This issue motivates our fourth contribution, in which we study practically motivated subclasses of

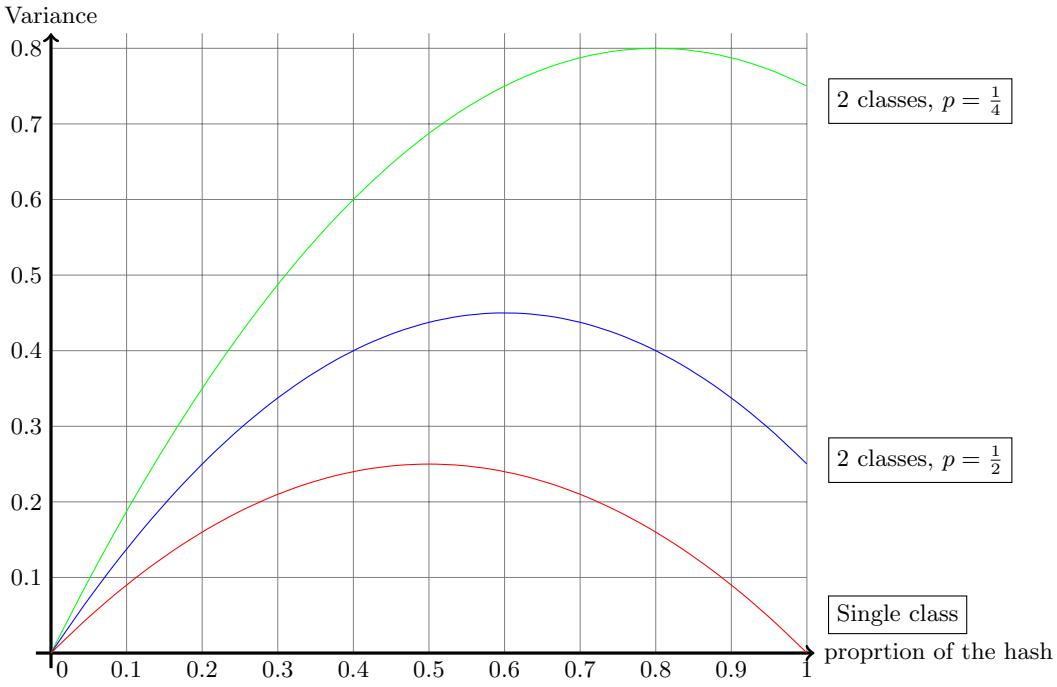

Fig. 1. Example variances in miner reward (over sequences of messages of a fixed length t) under the PSS RSS, as a function of the miner's fraction of the overall hashrate. For the single-class case, we use a reward of 1 per share. For the 2-class cases, we assume every share belonging to the second class (e.g., with at least 66 or at least 67 zeroes) belongs also to the first class (e.g., with at least 65 zeroes).
Let p denote the probability that a share that belongs to the first class also belongs to the second (e.g., $\frac{1}{2}$ or $\frac{1}{4}$). We use a reward of $\frac{1}{2}$ per share in the first class and an additional bonus for each share in the second class. The bonus is set so that the expected value of a share from the first class (which may or may not also belong to the second) is 1 (e.g., a bonus of 1 for $p = \frac{1}{2}$ and a bonus of 2 for $p = \frac{1}{4}$).

reward sharing schemes (RSSes), such as those that never run a deficit and must distribute any block reward immediately. We prove that the pay-per-last-N-shares method is variance-optimal among RSSes in a natural subclass (and not variance-optimal if the subclass restrictions are relaxed).

1.2 Related Work

The goals of this paper are closely related to those of Fisch et al. [7], although the model and conclusions differ. In [7], miner risk aversion is modeled via a concave utility function of the form $u(x) = x^\alpha$ (where x is the reward and $\alpha \in (0, 1)$); here, we assume that each miner's preference is to minimize reward variance

(subject to the expected reward being proportional to their hashrate).[5] Fisch et al. [7] then define "optimality" in terms of a global objective function, namely maximizing the total discounted utility of all miners. Because our optimality results apply to all miners simultaneously (and not just e.g. for the total variance of all miners), we are not forced to choose any method of aggregating benefits across miners. Finally, Fisch et al. [7] consider only what they call "pure" pooling strategies which do not allocate any rewards prior to a block authorization event (which have the advantage of never running a deficit) and thus the PPS scheme is outside of their model. (PPS does not run a deficit in the long run, but it can in the short run.) Because of these differences, the main result in [7] advocates for a geometric reward scheme (with the reward-per-share decaying with the share's distance from the next block authorization) to maximize total discounted utility; our theory singles out the pay-per-share scheme as variance-optimal. (Though our Theorem 5 does incorporate a no-deficit constraint to prove a restricted variance-optimality result for the pay-per-last-N-shares method.)

Much of the previous theoretical work on mining pools has focused on incentive aspects. For example, there are incentive issues both between different pools (e.g., pool-hopping [16]) and within a single pool (e.g., the miner's dilemma [5] or the delayed reporting of shares [18]). Another game-theoretic analysis of mining pools can be found in Lewenberg et al. [12], where the authors study the dynamics of a network with several mining pools. They show that there exists an instability in the choice of pools by miners, and that the miners will often switch pools, given some natural topological assumptions on the network. Along related lines, Laszka et al. [10] and Johnson et al. [8] examine the incentive of mining pools to attack each other. They show that in certain cases pools can benefit from such attacks.

We stress that while the present work does not focus on incentive issues per se (excepting the discussion in Section 4.4), our main results nevertheless advocate (on the basis of variance-minimization) for rules that happen to possess good incentive properties (such as PPS, see e.g. [18]). That is, our results optimize over all (not necessarily incentive-compatible) schemes and yet they champion schemes with strong incentive-compatibility properties.

Another line of works considers the forces behind and consequences of centralization (either outside of a mining pool or within a mining pool). An empirical study of this issue can be found in a recent paper by Romiti et al. [15]. The results of this study point to centralization tendencies inside pools, with a small number of miners reaping a large portion of the rewards. This raises incentive and security concerns motivated by the power that a small group may hold. In [6] Eyal and Sirer show that the Bitcoin protocol is not incentive compatible, in the sense that colluding miners could gain profits larger than their proportional hash power. As a counterpoint, in [9] the authors analyze mining as a stochastic game and show that as long as all of the miners are small, honest mining is a Nash equilibrium.

[5] Variance-minimization has been regarded as a key objective function for mining pool design since Rosenfeld's seminal analysis of Bitcoin mining pools [16].

Others have studied the setting where there are a few miners with large mining power (one can think of them as mining pools, but this does not have to be the case) and many small miners. For Bitcoin this was studied in [11]. For Ethereum, which has a slightly different reward allocation rule in which a miner can be rewarded for finding a block that does not end up in the main chain (known as "uncle" rewards), it was shown in [21] that powerful miners can attack weak miners.

A different approach taken in [1] takes into account the cost of mining equipment purchased by miners. There it is shown that with time one can expect that there will only be a small set of strong miners. These strong miners again can be interpreted as mining pools.

An axiomatic approach to reward allocation for miners was taken by Chen et al. in [4]. They start by stating the desired properties of reward allocation rule, such as symmetry, sybil-proofness, collusion-proofness, and others. They proceed to study which allocation rules satisfy these properties, showing that Bitcoin's allocation rule is the unique solution that satisfies a strong set of properties and that this is no longer the case for slightly weaker properties or if the miners are risk-averse.

Finally, we point the reader to two surveys that may be of interest. The first is a Systematization of Knowledge paper [2], where the authors examine results in the fields of game theory, cryptography, and distributed systems. The second [20] offers a systematic study of blockchain networks, focusing on the incentive aspects in the design of such systems.

2 Preliminaries

2.1 Model of Miners

We assume there is a finite set of miners, and use $[k] := \{1, 2, \ldots, k\}$ to denote their possible identities (public keys). (Any number of miner identities may belong to the same actor.) We assume that there is a finite message space M, such as $\{0,1\}^{256}$ (e.g., all possible hash function outputs). By a *signed message* (s, m) we mean a miner $s \in [k]$ and a message $m \in M$. The sets $[k]$ and M are known to all in advance. We assume that, due to capacity and communication considerations, a mining pool is willing to accept only a subset $A \subseteq M$ of the possible messages (e.g., those with at least 65 leading zeroes).[6]

Each miner s has a nonnegative hashrate h_s, not known a priori to the designer, which controls the rate at which s can generate signed messages. We model miner s as a Poisson process with rate h_s, with each generated message of the form (s, m) with m drawn uniformly at random from the message space M (e.g., the output of SHA-256 on a block with a specific nonce, under the ran-

[6] Our results remain the same if each miner has its own subset A_i of acceptable messages, provided the A_i's all have the same size.

dom oracle assumption).[7] We assume that a miner s sends one of its generated messages (s, m) to the mining pool if and only if $m \in A$ (e.g., a miner doesn't bother to send hashes with less than 65 zeroes). We can assume without loss of generality that the total hashrate is 1 ($\sum_{s=1}^{k} h_s = 1$) and hence the vector $\mathbf{h}$ of hashrates can be interpreted as a probability distribution, called the *hashrate distribution*.

The *message distribution* $\mathcal{M}(\mathbf{h})$ induced by a hashrate distribution $\mathbf{h}$ is the distribution over signed messages arriving at the mining pool. A sample (s, m) from $\mathcal{M}(\mathbf{h})$ can be generated by independently choosing a miner identity s according to the hashrate distribution and an acceptable message m uniformly at random from A. Each signed message received by the mining pool is an i.i.d. sample from the message distribution.

2.2 Reward Sharing Schemes

A *reward sharing scheme (RSS)* ascribes a (possibly random) reward to the sender of each signed message, given the messages received thus far. Formally, an RSS is a random function φ from finite sequences $(s_1, m_1), \ldots, (s_t, m_t)$ of signed messages to real-valued rewards (for the miner s_t). An RSS is *memoryless* if its output is independent of all but the most recent signed message. We write $\varphi(s, m)$ for the (random) output of a given memoryless RSS φ on a given signed message (s, m).

For example, the *pay-per-share (PPS)* RSS deterministically pays a fixed reward for each message received. That is,

$$\mathrm{PPS}(s, m) = c$$

for some $c > 0$.[8]

For a more involved example, consider the *pay-per-last-N-shares (PPLNS)* RSS, which distributes a fixed reward to the most recent N messages leading up to the pool's successful authorization of a block. Here, a miner's reward for a message depends on the future—on the number of blocks mined by the pool over the course of the next N messages. (Note, however, the miner's reward is independent of the past.) We can model this uncertainty in our RSS framework using random rewards:

$$\mathrm{PPLNS}(s, m) = c \cdot X,$$

where $c > 0$ is a constant and $X \sim Bin(N, p)$ is a binomial random variable, where the number of trials is N and the success probability p is the probability that a sample from the message distribution leads to a block authorization (e.g.,

[7] The Poisson assumption is for convenience. The important property is that the identity of the sender of a new signed message is distributed proportionally to the hashrate distribution, independent of the past.

[8] The constant $c > 0$ would typically be chosen so that the rate at which rewards are granted to miners equals the rate at which the pool accrues block rewards (and possibly transaction fees), less a commission.

if the acceptable messages A have 65 leading zeroes and 75 are necessary to authorize a block, then $p = 2^{-10}$).[9]

An example of a natural RSS that is not memoryless is the *proportional* RSS, which upon a block authorization distributes the corresponding reward to miners proportionally to the number of signed messages each miner sent since the most recent successful block authorization. Here, the reward to a miner for a signed message depends on the past, and specifically on how many signed messages the pool has received since the most recent successful block authorization.

2.3 Message-Independence and Symmetrization

An RSS φ is *message-independent* if $\varphi((s_1, m_1), \ldots, (s_t, m_t))$ is independent of $m_1, m_2, \ldots, m_t$, that is, the RSS does not take into account the content of the message $m_1, \ldots, m_t$.[10] Message-independence corresponds to the notion of "single-class shares" from the introduction—the RSS does not consider the contents of a message beyond its acceptability (i.e., membership in A). Thus our results about the optimality of single-class shares will be formalized as optimality results for message-independent RSSes.

The PPS, PPLNS, and proportional RSSes are all message-independent. (To avoid confusion, remember that every message reaching the RSS belongs to A; non-acceptable messages are filtered out beforehand.) Conditioning a reward on, for example, the number of leading zeroes in an acceptable message would lead to a non-message-independent RSS.

For a (not necessarily message-independent) RSS φ, define its *symmetrization* φ^{sym} by

$$\varphi^{sym}((s_1, m_1), \ldots, (s_t, m_t)) = \mathbb{E}_{u_1, \ldots, u_t \sim A}[\varphi(s_i, m_i)],$$

where the u_i's are i.i.d. uniformly random messages from A. For the special case of a memoryless RSS φ, we can write

$$\varphi^{sym}(s, m) = \mathbb{E}_{u \sim A}[\varphi(s, m)].$$

We immediately have:

Proposition 1. *For every RSS φ, its symmetrization φ^{sym} is message-independent.*

[9] Other schemes with future-dependent rewards can be similarly modeled. The key requirement is that the probability distribution over the reward associated with a share (with respect to future samples from the message distribution) is independent of the hashrate distribution $\mathbf{h}$. This is the case for most of the well-studied RSSes (including e.g. the geometric reward schemes studied in [7]).

[10] All of the common RSSes that motivate this work are also *anonymous*, meaning that $\varphi((s_1, m_1), \ldots, (s_t, m_t))$ is independent of $s_1, s_2, \ldots, s_t$. While anonymity is natural (and arguably unavoidable) in a permissionless blockchain setting, our positive results do not require that assumption. In any case, the RSSes advocated by our results are anonymous.

2.4 A Reward-Sharing Scheme as a Hashrate Estimator

To compare the statistical properties (such as variance-minimization) of different RSSes, it is useful to view an RSS as a statistical estimator of the hashrate distribution. By an *estimator*, we mean a function that associates each sequence $(s_1, m_1), \ldots, (s_t, m_t)$ with a probability distribution over the miners $[k]$.

Specifically, given an RSS φ, the corresponding estimator f_φ associates each sequence $(s_1, m_1), \ldots, (s_t, m_t)$ of signed messages with the k-vector $\mathbf{p}$ in which the jth component p_j is the fraction of the rewards awarded to miner j:

$$p_j = \frac{1}{Z} \cdot \sum_{i \in [t] \,:\, s_i = j} \varphi((s_1, m_1), \ldots, (s_i, m_i)), \tag{1}$$

where $Z = \sum_{i=1}^{t} \varphi((s_1, m_1), \ldots, (s_i, m_i))$ is a normalizing factor. For a memoryless rule φ one can write $\varphi(s_i, m_i)$ instead of $\varphi((s_1, m_1), \ldots, (s_i, m_i))$ in (1). If the RSS φ is randomized, so is the corresponding estimator f_φ (even for a fixed sample).

The *likelihood* of a sequence $(s_1, m_1), \ldots, (s_t, m_t)$ of signed messages with a hashrate distribution $\mathbf{h}$ is the probability that t i.i.d. draws from the message distribution $\mathcal{M}(\mathbf{h})$ induced by $\mathbf{h}$ is $(s_1, m_1), \ldots, (s_t, m_t)$. A *maximum likelihood estimator (MLE)* maps each sequence $(s_1, m_1), \ldots, (s_t, m_t)$ to a hashrate distribution maximizing the likelihood of that sequence. There is no a priori requirement that an MLE is induced by an RSS, though we'll see in Theorem 2 that the PPS RSS induces an MLE.

Some kind of unbiasedness assumption is required for meaningful variance-minimization results (otherwise, a constant function can achieve zero variance). Formally, we call an estimator f *unbiased* if, for every positive integer t and hashrate distribution $\mathbf{h}$,

$$\mathbb{E}[f((s_1, m_1), \ldots, (s_t, m_t))] = \mathbf{h}, \tag{2}$$

where the expectation is over t i.i.d. samples from $\mathcal{M}(\mathbf{h})$ and any randomization internal to the estimator. For example, the PPS, PPLNS, and proportional RSSes all induce unbiased estimators. Also, because the message m_i in a sample (s_i, m_i) from $\mathcal{M}(\mathbf{h})$ is chosen uniformly at random from A:

Proposition 2. *For every unbiased RSS φ, its symmetrization φ^{sym} is also unbiased.*

An estimator is *unbiased for miner s* if the identity in (2) holds in the sth coordinate. (An estimator is thus unbiased if and only if it is unbiased for every miner.)

For a given estimator f, positive integer t, and hashrate distribution $\mathbf{h}$, we can define its *miner-s t-sample variance* as

$$\mathbb{E}[(f((s_1, m_1), \ldots, (s_t, m_t)))_s - h_s)^2],$$

where the expectation is again over t i.i.d. samples from $\mathcal{M}(\mathbf{h})$ and any randomization internal to the estimator. The *t-sample variance* of an estimator f for a hashrate distribution $\mathbf{h}$ is the vector of all such variances (ranging over the miner s).

2.5 When t Is Random

The t-sample variance refers to a fixed number t of samples, corresponding to a fixed number of miner shares. If one instead fixes an amount of *time*, then t itself is a random variable (the number of shares found during that time window, which is distributed according to a Poisson distribution). All of our t-sample variance-optimality results (such as Theorems 4 and 5) hold simultaneously for all positive integers t. Thus, by the law of total variance (see equation (3)), these variance-optimality results carry over to the case in which t is a random variable (e.g., the case of a fixed time window).

3 Warm-Up: Maximizing Likelihood

We begin with an observation that champions PPS from a statistical prediction perspective—the corresponding estimator is in fact a maximum likelihood estimator for the hashrate distribution $\mathbf{h}$ (given t i.i.d. samples from the message distribution $\mathcal{M}(\mathbf{h})$).[11,12] The next section describes the main results of this paper, on variance-optimality.

Theorem 1 (PPS Is an MLE). *The estimator f_{PPS} induced by the PPS reward-sharing scheme is an MLE.*

Before we prove Theorem 1, we need to state the following result that can be found, for example, in [19]. Let $X_1, ..., X_n$ be i.i.d random variables with a discrete support $[k]$. For $s \in [k]$ let $n_s = |\{X_i = s\}|$. We say that $\mathbf{p} = (p_1, ..., p_k)$ is the *maximum likelihood estimate* if

$$\mathbf{p} = argmax_{\mathbf{q} \in Q} \prod_{s=1}^{n} q_s^{n_s}$$

where $Q = \{\mathbf{q} \in \mathbb{R}^k : \sum_{s=0}^{k} q_s = 1, \forall s \; q_s \geq 0\}$ denotes the simplex.

Theorem 2. *The maximum likelihood estimate is given by the empirical distribution defined by*

$$p_s = \frac{n_s}{n}$$

for all $s \in [k]$.

We are now ready to prove Theorem 1.

Proof (Proof of Theorem 1). First note that the values of the s_i's are i.i.d. For every i, s_i is chosen by the hash distribution, and for any $i \neq j$, s_i is independent of s_j as the Poisson process is memoryless. Furthermore, the estimator induced

[11] MLEs are deterministic (up to tie-breaking). Thus no randomized RSS (such as PPLNS or the estimator induced by the proportional rule) can be a MLE.

[12] A similar result can be found in [14] under the reasonable assumption that the shares follow the Poisson distribution.

by PPS is the empirical distribution. Indeed, when focusing on PPS (with a fixed reward of c per share), the identity in (1) specializes to

$$\begin{aligned} p_s &= \frac{1}{c \cdot t} \cdot \sum_{i \in [t] \,:\, s_i = s} c \\ &= \frac{|\{i \,:\, s_i = s\}|}{t}. \end{aligned}$$

Theorem 1 now follows from Theorem 2.

While we believe Theorem 1 is a novel way to single out one RSS among many, the primary purpose of mining pools is variance-minimization, not prediction per se. We therefore proceed to our main results on variance-optimality, in which the PPS RSS will continue to play a central role.

4 Main Results: Variance-Optimality

4.1 Single-Class Shares Are Optimal

Our first result proves that single-class shares are optimal, in the sense that symmetrization can only reduce the variance of every miner.

Theorem 3 (Message-Independence Minimizes Variance). *For every unbiased RSS φ, hashrate distribution $\mathbf{h}$, positive integer t, and miner $j \in [k]$, the miner-j t-sample variance under the estimator f_φ is at least as large as under its symmetrization $f_{\varphi^{sym}}$.*

Proof. The law of total variance states that for X and Y, random variables over the same probability space,

$$\mathrm{Var}[X] = \mathbb{E}[\mathrm{Var}[X \mid Y]] + \mathrm{Var}[\mathbb{E}[X \mid Y]]. \tag{3}$$

For an unbiased RSS, we have that $\mathbb{E}[(f((s_1, m_1), \ldots, (s_t, m_t)))_j - h_j)^2] = \mathrm{Var}[(f_\varphi((s_1, m_1), \ldots, (s_t, m_t)))_j]$, and so we can apply the above. Here we take $X = f_\varphi(\mathbf{s}, \mathbf{m})_j$, where $\mathbf{s} = (s_1, ..., s_t)$ and $\mathbf{m} = (m_1, ...m_t)$. Note that $\mathbb{E}_{\mathbf{m}}[f_\varphi(\mathbf{s}, \mathbf{m})_j \mid \mathbf{s}] = f_{\varphi^{sym}}(\mathbf{s}, \mathbf{m})_j$.

Plugging this into (3) we find that

$$\mathrm{Var}[f_\varphi(\mathbf{s}, \mathbf{m})_j] = \mathbb{E}_{\mathbf{s}}[\mathrm{Var}[f_\varphi(\mathbf{s}, \mathbf{m})_j \mid \mathbf{s}]] + \mathrm{Var}[f_{\varphi^{sym}}(\mathbf{s}, \mathbf{m})_j].$$

The first term on the right-hand side is always nonnegative, and it equals 0 if $f_\varphi = f_{\varphi^{sym}}$. This completes the proof.

For example, consider a version of the PPLNS scheme that uses two classes of shares (e.g., corresponding to at least 65 and at least 75 leading zeroes), with different rewards (e.g., more for 75 zeroes). Theorem 3 implies that all miners would enjoy lower variance (and the same expectation) if instead every share of either type was rewarded according to the expected reward of a share in one of the two classes (where the expectation is over a uniformly random message from A).

4.2 PPS Is Variance-Optimal

Theorem 3 is agnostic to all aspects of an RSS other than message-independence—for example, it offers no opinion on which of PPS or PPLNS is "better." Our next result singles out PPS as variance-optimal among all unbiased RSSes. We discuss this point further and revisit PPLNS after the proof of Theorem 5.

Theorem 4 (PPS Is Variance-Optimal). *For every hashrate distribution* $\mathbf{h}$*, positive integer* t*, and miner* $j \in [k]$*, the PPS RSS minimizes the miner-*j t*-sample variance over all estimators that are unbiased for miner* j*.*

To prove this, we show the following general statement:

Lemma 1. *Let* $x = (m, \mathbf{b})$ *be such that* m *is chosen uniformly from some set* $\mathcal{F}$ *and* $\mathbf{b}$ *are the results of* l *coin flips for some constant* l*. Let* $\{F_i\}$ *be pairwise disjoint subsets of* $\mathcal{F} \times \{0,1\}^l$*. Let*

$$X = \sum_i a_i \mathbf{1}_{x \in F_i}$$

and assume that $\mathbb{E}[X] = R$ *for some constant* R*. Then the choice of* a_i *that minimizes the variance of* X *is* $a_i = \frac{R}{\Pr[x \in \cup F_i]}$*.*

Intuitively we can think about the elements of Lemma 1 as follows. Assume that a bitcoin mining pool runs a version of PPS where the reward for a miner is a function of the number of leading zeroes in the hash. Then each F_i will be the set of messages with a given number of leading zeroes, and for such a message the pool will give a reward of a_i.

If there is some extra randomness beyond the sampling of the miner, say as in PPLNS, then this randomness will appear in the coin flips $\mathbf{b}$. In this case, a family F_i could be, for example, all the messages with the number of leading zeros between 65 and 70, for which the coin flips sum up to exactly 5.

The reward given for the family F_i can be a function also of the history $(s_1, m_1), ..., (s_{t-1}, m_{t-1})$.

Lemma 1 essentially shows that the minimum variance is obtained by giving the same reward for every family F_i, that is, discarding any information given by the message or the randomness of the RSS.

Proof (of Lemma 1). Let $X = \sum_i a_i \mathbf{1}_{x \in F_i}$. Remembering that $\mathrm{Var}[X] = \mathbb{E}[[X - \mathbb{E}[X]]^2]$ we have that the smallest possible variance $\mathrm{Var}[X] = 0$ is obtained if and only if $X = \mathbb{E}[X]$.

In our notation, this means that for every i and i' we must have that $a_i = a_{i'}$, and as $\mathbb{E}[X] = R$ this gives us that for every i, $a_i = \frac{R}{\Pr[x \in \cup F_i]}$ as needed.

It is left to deduce Theorem 4 from Lemma 1.

Proof (of Theorem 4). To show this, we first choose $\mathcal{F} = A$, the family of all messages that can be sent to the pool. Second, for a given RSS we define as F_i

the set of all the messages and result of the extra randomness that get the reward a_i. The expectation R is given by the assumption that the RSS is unbiased.

Note that in an unbiased RSS a miner j cannot get paid for the a message sent by another miner (e.g., if $h_j = 0$ then any payment to j would make the RSS biased). To minimize the variance of miner j it is left to decide on the reward of messages sent by them. By Lemma 1 we have that the smallest variance will be obtained by giving the same reward for every message and flipping no extra coins. This is exactly the definition of PPS.

Finally, we note that Theorem 4 considers minimizing the variance of some fixed miner $s \in [k]$. As the optimal RSS is PPS (independent of s), it is actually variance-optimal for all of the miners simultaneously. More precisely, we have the following:

Corollary 1 (Miner-Optimality). *For every hashrate distribution* $\mathbf{h}$ *and positive integer* t*, the PPS RSS simultaneously minimizes the variance of the miner-*s t*-sample for all* s *over all unbiased estimators.*

It is interesting to note that Corollary 1 holds even if each miner chooses their own difficulty of shares (as is common in some mining pools). A miner that is free to choose its difficulty in the PPS RSS can lower the variance further by choosing the smallest possible difficulty.

We emphasize that there is no a priori guarantee that a statement like Corollary 1 should hold for any RSS—for example, it is conceivable that small miners would fare better under one scheme and large miners under a different one. Corollary 1 shows that no trade-offs between different miners are necessary. This is particularly interesting if we add other restrictions on the RSS.

4.3 Variance-Optimality of PPLNS

A shortcoming of PPS is that for any fixed reward per share, there is a constant probability that at some point the pool will not have the funds to pay the miners. The PPLNS method is common in practice and does not suffer from this drawback. Does PPLNS become variance-optimal if we impose additional constraints on an RSS?

We consider four constraints on an RSS.

(P1) *No-deficit.* An RSS can only distribute block rewards that have been earned to date.
(P2) *Liquidating.* An RSS must distribute each block reward as soon as the block is found.
(P3) N*-bounded.* The reward given for a share depends on at most the next N shares found.
(P4) *Past-agnostic.* The distribution of the reward for a share should not depend on the realizations of past rewards or on past blocks found.

The PPLNS method is variance-optimal for all miners, subject to (P1)–(P4).

Theorem 5. *For every hashrate distribution* $\mathbf{h}$ *and positive integer* t*, the PPLNS RSS simultaneously minimizes the variance of the miner-*s t*-sample for all* s *over all unbiased RSSes that satisfy properties (P1)–(P4).*

Before proceeding to the proof, we note that PPLNS is *not* variance-optimal if any of the properties (P1)–(P4) are relaxed. Most of the RSSes that demonstrate this are not particularly attractive, however, as they suffer from serious incentive problems (see Section 4.4 for further discussion). The point of Theorem 5 is not so much to argue that PPLNS is the only reasonable RSS for variance-minimization, but rather to clarify the types of transgressions required by any RSS that does better.

Proof. (of Theorem 5) Let f be an unbiased RSS that follows constraints (P1)–(P4) and has minimum variance. By (P1) we know that it can distribute only block rewards that were already obtained, and by (P2) we know that it has to distribute the reward immediately when a block is found. Thus, it is enough to determine its behavior at the appearance of a block. Furthermore, by (P3) we know that the reward can only be distributed among the last N shares, and so f takes as an input only the last N messages before the block, even if $t > N$. Thus, we can focus on a function that given the fact that share j is a block, distributes the reward found among the shares $j - N - 1, j - N - 2, ..., j$. Call this function f_j and note that $f = \sum_{j \text{ is block}} f_j$.

By (P4), f_j cannot depend on whether any of the other N shares is a block or if the share already received a reward from a different block. The information available for f_j is the message sent with the shares, the arrival time of the share with respect to the block, and the identity of the sender. This makes f_j independent of any $f_{j'}$, for $j \neq j'$, and so it is enough to minimize the variance of each f_j separately.

After considering (P1)–(P4) we see that f_j is an unbiased RSS over an N-sample. By Corollary 1, PPS minimizes the variance for all of the miners simultaneously, and so in the context of f_j this means that each of the N shares receives the same reward. By (P2) all of the block reward needs to be distributed, so each shares gets $1/N$ of the block reward, which is exactly the definition of PPLNS.

4.4 Relaxing the Constraints

All the properties (P1)–(P4) are required for the variance-optimality result in Theorem 5. For starters, if the no-deficit condition, (P1), is dropped, the PPS method has smaller variance.

Suppose the liquidating constraint (P2) is dropped. That is, an RSS need not allocate the full block reward. With partial reward distributions, we can again find an RSS with a smaller variance: For each share, reward a fixed amount if in the next N shares at least one block is found. (The reward is the same, whether 1 or 17 blocks are found over the next N shares.) This RSS has smaller variance than PPLNS as the reward does not depend on the number of blocks found. The variance of this RSS becomes smaller as a function of N, but a

direct consequence of this is that a large portion of funds will not be distributed. Furthermore, it might incentivize miners to delay the publication of blocks until they have published enough shares.

If N-boundedness (P3) is dropped, consider the following RSS. For a fixed N let M be the expected number of blocks within N shares. For each share, reward a sum proportional to M/N. If there are not enough funds in the pool, wait for the next block to be found and start paying shares, ordered from the oldest to the newest. This results in an RSS very similar to PPS, but with the risk of funds delaying significantly. This RSS suffers from incentive issues, however: As the delays inevitably grow, miners are incentivized to leave the pool for greener pastures.

Finally, an RSS which is not past agnostic (P4) and has a smaller variance is the following. Assume, again, that we expect to find M blocks for every N shares. Then, for every block found, look back at the last N shares and reward each one with a sum that will make their reward as close to M/N as possible. If not all of the reward was distributes or if the reward is not enough to bring the shares to M/N, distribute the reward in a way that will make the reward of all of the N shares as even as possible. Although this RSS has a smaller variance, it creates a negative incentive to mine if no block was found for a while.

Although the examples above have obvious incentive problems, it is not clear that any such relaxation will create these issues. Studying this further may be of interest.

5 Conclusions and discussion

In this work, we have proposed a model for investigating the variance-minimization properties of different mining pool reward-sharing schemes. We focused on two design decisions: (i) What information from miners should be the basis for their rewards?; and (ii) How should the information submitted by miners determine their rewards? Our results strongly support the common practice of using a single class of shares, as the use of finer-grained information can only increase the variance experienced by every miner. This holds true across different ways of translating single-class shares into miner rewards (PPS, PPLNS, etc.). Our results also strongly support the pay-per-share scheme, which can be justified both as a maximum likelihood estimator for the miner hashrate distribution and as the scheme that minimizes the variance of all miners simultaneously, over all unbiased estimators.

This work focused single-mindedly on variance-minimization. This tunnel vision is deliberate, both because it enables a tractable theory with particularly crisp and interpretable results, and because in many cases it only makes our results stronger. For example, our main results do not restrict consideration to reward-sharing schemes with desirable incentive properties, but nevertheless advocate (as variance-optimal) schemes that do have such properties.

Needless to say, there are many other scientifically interesting and practically relevant dimensions along which one can compare reward-sharing schemes, all

of which should be taken into account in a real design. For example, in some settings the variance-minimization benefits of the pay-per-share scheme may be outweighed by the risk that would be taken on by the pool owner.

References

1. Nick Arnosti and S Matthew Weinberg. Bitcoin: A natural oligopoly. *arXiv preprint arXiv:1811.08572*, 2018.
2. Sarah Azouvi and Alexander Hicks. Sok: Tools for game theoretic models of security for cryptocurrencies. *arXiv preprint arXiv:1905.08595*, 2019.
3. Vitalik Buterin et al. Ethereum: A next-generation smart contract and decentralized application platform. *URL https://github. com/ethereum/wiki/wiki/% 5BEnglish% 5D-White-Paper*, 2014.
4. Xi Chen, Christos Papadimitriou, and Tim Roughgarden. An axiomatic approach to block rewards. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 124–131, 2019.
5. Ittay Eyal. The miner's dilemma. In *2015 IEEE Symposium on Security and Privacy*, pages 89–103. IEEE, 2015.
6. Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer, 2014.
7. Ben Fisch, Rafael Pass, and Abhi Shelat. Socially optimal mining pools. In *International Conference on Web and Internet Economics*, pages 205–218. Springer, 2017.
8. Benjamin Johnson, Aron Laszka, Jens Grossklags, Marie Vasek, and Tyler Moore. Game-theoretic analysis of ddos attacks against bitcoin mining pools. In *International Conference on Financial Cryptography and Data Security*, pages 72–86. Springer, 2014.
9. Aggelos Kiayias, Elias Koutsoupias, Maria Kyropoulou, and Yiannis Tselekounis. Blockchain mining games. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 365–382, 2016.
10. Aron Laszka, Benjamin Johnson, and Jens Grossklags. When bitcoin mining pools run dry. In *International Conference on Financial Cryptography and Data Security*, pages 63–77. Springer, 2015.
11. Nikos Leonardos, Stefanos Leonardos, and Georgios Piliouras. Oceanic games: Centralization risks and incentives in blockchain mining. In *Mathematical Research for Blockchain Economy*, pages 183–199. Springer, 2020.
12. Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, and Jeffrey S Rosenschein. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 919–927. Citeseer, 2015.
13. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.
14. Ewa Paszek. Introduction to statistics. 2007.
15. Matteo Romiti, Aljosha Judmayer, Alexei Zamyatin, and Bernhard Haslhofer. A deep dive into bitcoin mining pools: An empirical analysis of mining shares. *arXiv preprint arXiv:1905.05999*, 2019.
16. Meni Rosenfeld. Analysis of bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.

17. Meni Rosenfeld et al. A short note about variance and pool payouts. *URL https://bitcointalk.org/index.php?topic=5264.0*, 2011.
18. Okke Schrijvers, Joseph Bonneau, Dan Boneh, and Tim Roughgarden. Incentive compatibility of bitcoin mining pool reward functions. In *International Conference on Financial Cryptography and Data Security*, pages 477–498. Springer, 2016.
19. Amnon Shashua. Introduction to machine learning: Class notes 67577, 2009.
20. Wenbo Wang, Dinh Thai Hoang, Peizhao Hu, Zehui Xiong, Dusit Niyato, Ping Wang, Yonggang Wen, and Dong In Kim. A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access*, 7:22328–22370, 2019.
21. Alexei Zamyatin, Katinka Wolter, Sam Werner, Peter G Harrison, Catherine EA Mulligan, and William J Knottenbelt. Swimming with fishes and sharks: Beneath the surface of queue-based ethereum mining pools. In *2017 IEEE 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 99–109. IEEE, 2017.