blockchains

MDPI

Article

PROACTION: Profitable Transactions Selection Greedy Algorithm in Rational Proof-of-Work Mining †

Mariano Basile ‡, Giovanni Nardini ‡, Pericle Perazzo ‡ and Gianluca Dini *,‡

Department of Information Engineering, University of Pisa, Largo Lucio Lazzarino 1, 56100 Pisa, Italy; mariano.basile@ing.unipi.it (M.B.); giovanni.nardini@unipi.it (G.N.); pericle.perazzo@unipi.it (P.P.)

* Correspondence: gianluca.dini@unipi.it; Tel.: +39-050-221-7549

† Basile, M.; Nardini, G.; Perazzo, P.; Dini, G. A Rational Mining Strategy for Proof-of-Work Consensus Algorithms. In Proceedings of the 4th Conference on Blockchain Research and Applications for Innovative Networks and Services (BRAINS), Paris, France, 27–30 September 2022; pp. 59–66. https://doi.org/10.1109/BRAINS55737.2022.9909327.

‡ The authors contributed equally to this work.

Abstract: Despite the many consensus algorithms being used in blockchains, proof of work (PoW) is still the most common nowadays. The state-of-the-art mining strategy for PoW-based blockchain protocols consists of including as many transactions as possible in a block to maximize the block reward. Unfortunately, this strategy maximizes the block orphaning probability too. Recently, we proposed a rational mining strategy aimed at carefully balancing the trade-off between the block reward and the risk of block orphaning. In this work, we present PROACTION, a PROfitable transACTions selectION greedy algorithm that implements such a strategy. We evaluate the algorithm both analytically and experimentally on Bitcoin by assuming a variable random percentage of winning miners adopting PROACTION. Experiments show that when executing PROACTION, miners gain higher long-term rewards than when using the state-of-the-art strategy. The gain is in the order of the block orphaning probability. This result is particularly relevant for those PoW-based blockchain protocols in which such a probability is significant.

Keywords: blockchain; proof of work; mining

Academic Editors: Keke Gai and Liehuang Zhu

Received: 12 November 2024
Revised: 15 January 2025
Accepted: 16 January 2025
Published: 22 January 2025

Citation: Basile, M.; Nardini, G.; Perazzo, P.; Dini, G. PROACTION: Profitable Transactions Selection Greedy Algorithm in Rational Proof-of-Work Mining. *Blockchains* **2025**, *3*, 2. https://doi.org/10.3390/blockchains3010002

1. Introduction

Despite the introduction of many consensus algorithms, proof of work (PoW) is still the most common algorithm. At the time of writing, 167+ blockchain protocols, i.e., 60% of the total cryptocurrency market capital, are based on PoW [1]. Currently, Bitcoin, Dogecoin, Bitcoin Cash, Litecoin, Ethereum Classic, and EthereumPoW, the top five cryptocurrencies by market capitalization, are based on PoW [1].

Due to its widespread adoption, a blockchain technology based on proof of work (PoW) represents a vibrant area of research [2–4]. Prominent topics of investigation include its sustainability, security, comparisons with Proof-of-Stake (PoS) technology, and the influence of machine learning [5–9]. This paper focuses on the *transaction selection problem*,which pertains to the challenge faced by miners in selecting transactions to include in the next block under construction [10,11]. The remainder of this section provides an overview of the transaction selection problem and outlines our contribution to addressing this challenge.

Informally, the PoW consensus algorithms allow miners to compete to create blocks as follows. Initially, a miner selects some transactions from its own local transaction set (also known as transaction pool) and includes them in a block. Then, the miner solves a

computationally intensive cryptographic hashing problem. Finally, the miner propagates the block in the blockchain network. During propagation, and upon receiving a new block, a miner first validates it and then forwards it as needed. It follows that block propagation delay encompasses block verification delay. Thus, the bigger the block, the more time it takes to propagate and validate the block itself, and therefore, the higher the block propagation delay.

PoW-based blockchain protocols provide economic incentives to miners for proposing blocks that finally become part of the valid consensus chain. These incentives are based on two mechanisms, namely *static reward* and *transaction fees*, which are both measured in terms of the underlying cryptocurrency. The static reward is defined by the blockchain protocol. In general, the static reward decreases or remains constant over time, according to the specific blockchain system [10]. In contrast, a transaction fee is determined by the sender of a transaction. It is an additional incentive paid to miners to encourage them to select the transaction when creating a block. The sum of the static reward and transaction fees, i.e., the fees of all transactions in a block, is referred to as *block reward*. Today, the state-of-the-art mining strategy in PoW aims to maximize the block reward by maximizing the reward that comes from transaction fees [12]. Therefore, upon building a new block, a miner tends to include in the block as many transactions as possible, and therefore, it tends to generate "large" blocks. This is the strategy employed by, among others, the Bitcoin Core software, version 28.1, which is the reference and most used (full-node) implementation of the Bitcoin protocol [13,14].

Increasing the number of transactions within a block results in an increase in the block's size, which, in turn, leads to a longer block propagation delay. This delay has been demonstrated to be a linear function of the block size [15–17]. An increase in block propagation delay consequently raises the likelihood of a blockchain *fork* [18]. A fork occurs when, during the propagation and validation of a newly mined block, another miner solves the cryptographic challenge and proposes a smaller block that propagates more quickly. This leads to the creation of two blocks that reference the same previous block in the chain. Ultimately, the consensus algorithm resolves the conflict by determining which block prevails in the race condition and becomes part of the valid consensus chain. The miner of the winning block, referred to as the winning miner, receives the block reward. Conversely, the block that loses the race is discarded and is referred to as an *orphan* block. The miner who created this block, known as the losing miner, forfeits the block reward. This phenomenon is termed *block orphaning* [15]. Consequently, the larger the block size, the greater the likelihood of the block becoming an orphan [19,20].

A block mining strategy that simultaneously minimizes the block orphaning probability and maximizes the block reward is not achievable. Therefore, a *rational* mining strategy should balance these two conflicting objectives to achieve a trade-off.

In a preliminary work [20], we proposed pursuing such a rational mining strategy through a proper transaction selection, which aims to maximize the expected block reward and accounts for both the block reward and the probability of orphaning. While keeping the traditional proof-of-work consensus protocol, which is unlike the state-of-the-art, the decision to include a given transaction in a block must be based not only on the additional reward deriving from the transaction fee but also on the increased risk of block orphaning due to the transaction's size.

In this paper, we present PROACTION, a greedy transaction selection algorithm that allows the above rational mining strategy. In short, PROACTION processes transactions in the order they are maintained in the local transaction set (i.e., transaction pool), prioritizing those with the highest revenue first. A transaction is included in the block if and only if its addition increases the expected block reward compared to the current value. PROAC-

TION evaluates the expected block reward by exploiting a model of the block orphaning probability based on the past behavior of the network.

Overall, this paper builds upon and significantly expands the preliminary work presented in [20] across several dimensions:

- We extend the state-of-the-art Blocksim simulator by PROACTION's transaction selection algorithm.
- We evaluate the impact of the rational mining strategy on Bitcoin performance evaluation metrics, including network throughput, long-term block reward, and orphan block count, by assuming that a variable random percentage β of winning miners adopt PROACTION.
- We conduct a fair long-term comparison between PROACTION and the state-of-the-art mining strategy, considering the same percentage β and the same set of winning miners.
- We experimentally demonstrate that the average number of orphan blocks produced under the rational mining strategy is always strictly lower than that generated by the state-of-the-art mining strategy.
- We experimentally prove that miners using PROACTION achieve higher long-term rewards, regardless of β. However, this increased long-term reward comes at the expense of reduced network throughput. The results obtained in the context of Bitcoin show that the percentage change in long-term reward is closely aligned with the block orphaning probability.

The rest of the paper is organized as follows. Section 2 discusses related works. Section 3 analyzes the state-of-the-art mining strategy in the framework of PoW-based blockchain protocols, showcasing its properties and limitations. Then, it presents the rational mining strategy. Section 4 presents the PROACTION rational mining algorithm, providing a detailed discussion of its implementation and an analysis of its computational complexity. Section 5 experimentally evaluates the impact of the PROACTION rational mining algorithm. Finally, Section 6 draws final conclusions.

2. Related Works

Efforts have been made to study transaction selection algorithms in the context of PoW-based blockchains through feature extraction techniques. This is particularly true for the Bitcoin blockchain, given the enormous attention and interest it has garnered. Initially, after Bitcoin's inception, the transaction rate was very low, so transactions to be included in the next block were predefined, and there was no need for a specific transaction selection algorithm. However, as Bitcoin gained popularity, the transaction rate increased. Furthermore, with the rise of pool mining, transaction selection algorithms became a key decision for Bitcoin pool servers.

Fiz et al. [21] develop a model for the transaction selection algorithm used by Bitcoin pool servers on the set of transactions. Based on this model, the authors aim to predict whether a transaction will be included in the next block by framing the problem as a supervised classification problem. The authors identify a set of features and demonstrate that, in the transaction selection process, the most important features are transaction size and fee. The latter determines the fee rate, which in turn dictates the order in which transactions get included in a block. This aligns with our approach, where transaction size and fee are also the most significant features.

Later, Fiz et al. extend the work presented in [21]. In [22], the authors propose a method for monitoring the transaction selection algorithm used by Bitcoin pool servers. They frame the selection algorithm as a classification problem, wherein a dataset of transactions is created for each block and separated by a Bitcoin pool server. Each transaction is a vector of features. For each Bitcoin pool server, feature selection techniques are employed to extract

a vector of significant features. As new blocks are incorporated into the valid consensus chain, variations in feature importance are monitored. Finally, the authors demonstrate how a change in the transaction selection algorithm employed by a pool server could be detected.

In addition to examining transaction selection algorithms through feature extraction techniques, several transaction selection algorithms suitable tailored for PoW-based blockchain protocols have been proposed in the literature.

Santos et al. propose a solution called Size-Density Table (SDT), a data structure designed to efficiently manage the transaction pool [23]. The SDT enables constant-time transaction insertion and deletion operation, eliminating the need for the miners to sort the transaction pool. In conjunction with this, the authors present a transaction selection algorithm that leverages the SDT. Through experimental evaluation, the authors demonstrate that their algorithm enables faster Bitcoin block creation compared to sorting-based algorithms, primarily due to the absence of transaction sorting time. Additionally, the authors show that the total size of the Bitcoin transactions selected by the SDT-based surpasses that of the sorting-based algorithm. However, with the increased block size, this approach also heightens the risk of block orphaning. As a result, the SDT aligns with the state-of-the-art mining strategy by aiming to maximize the block size, thereby increasing the block orphaning probability. In fact, the block orphaning probability associated with the SDT approach exceeds that of the state-of-the-art strategy, as the SDT entirely neglects the consideration of orphaning risk.

Rizun shows how a rational Bitcoin miner should select transactions for inclusion in a block to maximize the block reward in the absence of a block size limit [24]. Rizun introduces the *block space supply curve* and the *transactions demand curve*. The block space supply curve models the cost of providing block space, incorporating the orphaning risk. The transaction demand curve represents the fees offered by the transactions. Rizun further demonstrates the relationship between these and their derivatives, drawing upon classical economic principles. Moreover, he proves that, in the absence of a block size limit, maximizing profit involves producing a block size corresponding to the intersection point of these two curves. Unlike Rizun's framework, our analysis focuses on a real-world scenario where a block size limit exists. We conclude by discussing alternative transaction selection algorithms in comparison to PROACTION and the state-of-the-art strategy. The First-In-First-Out (FIFO) transaction selection algorithm includes transactions into a block in the order they are received, ensuring fairness by avoiding any form of sorting [25]. Consequently, FIFO is computationally more efficient than PROACTION, as it eliminates the need for sorting operations. Moreover, a FIFO algorithm ensures that all transactions are eventually processed, avoiding their starvation. However, it is clearly suboptimal from the point of view of miner profitability because it does not consider fees at all. In contrast, the Highest Fee First (HFF) transaction selection algorithm sorts transactions waiting to be confirmed in descending order of their offered fees [26]. While HFF has the same computational complexity as PROACTION and the state-of-the-art strategy due to its sorting step, it is less profitable than both. This disparity arises because HFF prioritizes transactions based on their fee and not on their *fee rate*, which is the ratio between the fee and the memory occupation in bytes. Consequently, HFF does not distinguish between large and small transactions, treating a large transaction that occupies significant block space the same as a small transaction with the same fee. By considering both the fee and the space a transaction occupies, PROACTION adopts a more profitable approach, given the block size limitation, as it optimizes the selection of transactions that maximize the reward per block.

The review of the existing research presented above highlights several unresolved challenges associated with transaction selection algorithms for proof-of-work (PoW)-based blockchains. Notably, real-world constraints, such as block size limitations and orphaning risks, are often neglected. Traditional algorithms, such as FIFO and HFF, offer simplicity and computational efficiency but similarly fail to account for these critical constraints. This paper addresses these gaps by introducing an efficient method for selecting the most suitable set of transactions to be included in a block, which are tailored for realistic PoW-based blockchain environments. Particular attention is given to the Bitcoin protocol, which remains the most widely adopted PoW-based blockchain.

3. Mining Strategy Analysis

In this section, we analyze the state-of-the-art mining strategy (Section 3.1), and we argue why it is not rational. Then, based on its shortcomings, we propose a rational mining strategy (Section 3.2).

3.1. State-of-the-Art Mining Strategy

We discuss the miner's probability of obtaining a block reward (i.e., the miner's block rewarding probability) under the state-of-the-art mining strategy. This is the strategy employed by, among others, the Bitcoin Core software, which is the reference and most used (full-node) implementation of the Bitcoin protocol [13,14]. We refer to this probability as $\mathcal{P}_{reward,art}$. The time steps required in order for a miner to obtain a block reward (Figure 1) are as follows: the block creation (Step 1), the block mining (Step 2), and the block propagation (Step 3).

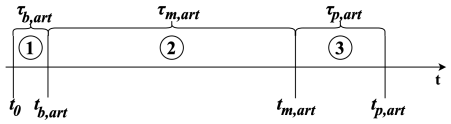

Figure 1. Time steps in order for a miner to obtain a block reward.

In Step 1, the block creation Step takes place as soon as a block has been received, (successfully) validated, and appended to the local copy of the blockchain. In Step 1, the miner includes in a block b as many transactions as possible from the transaction pool until one of the following occurs: (i) The block size limit (${b_s}^{max}$) is reached, i.e., the sum of the transaction size (tx_s) in the block equals the block size limit; (ii) There are no more transactions that can fit into the block. The rationale behind the state-of-the-art mining strategy is to maximize the block reward by maximizing the reward coming from transaction fees. We recall that, by definition, the block reward ($\mathcal{R}$) is the sum between the static reward ($\mathcal{R}_{static}$) and the reward coming from transaction fees (tx_{fee}). So, the state-of-the-art mining strategy aims to solve the following constrained optimization problem:

$$\begin{cases} \max \mathcal{R} = \mathcal{R}_{static} + \max \sum_{tx \in b} \mathrm{tx}_{fee} \\ \sum_{tx \in b} \mathrm{tx}_s \leq {b_s}^{max} \end{cases} \tag{1}$$

We refer to the time interval it takes to create a block as follows:

$$\tau_{b,art} = t_{b,art} - t_0 \tag{2}$$

that is the difference between the end block creation instant $t_{b,art}$ and the start block creation instant t_0.

In Step 2, the block mining step, the miner tries to solve the hashing-based mathematical challenge based on the block created in the previous step. Solving the challenge

can be modeled as a Bernoulli process. Typically, each miner tries so many times that the Bernoulli process can be well approximated by a Poisson process [27]. In this respect, the Poisson rate parameter μ (i.e., the average block mining rate) is blockchain-dependent. The higher the block mining rate, the shorter the block mining time. For instance, with respect to the Bitcoin blockchain, it is $\mu_{\text{Bitcoin}} = \frac{1}{600}$ s. In the pre-Merge Ethereum blockchain, it is $\mu_{\text{Ethereum}} = \frac{1}{13}$ s. Each miner $i \in \mathcal{N}$, where $\mathcal{N}$ is the set of natural numbers, has an individual hash rate x_i. A miner $i \in \mathcal{N}$ with hash rate x_i has a relative hash rate α_i with respect to the network's hash rate defined as follows:

$$\alpha_i = \frac{x_i}{\sum_{j \in \mathcal{N}} x_j}, \quad \alpha_i > 0. \tag{3}$$

The probability that a miner $i \in \mathcal{N}$ solves the mathematical challenge is directly proportional to its relative hash rate α_i [28–30]. From a temporal standpoint, we indicate the mining time with the following:

$$\tau_{m,art} = t_{m,art} - t_{b,art} \tag{4}$$

that is the difference between the end mining instant $t_{m,art}$ and the start mining instant $t_{b,art}$.

Finally, in Step 3, the block propagation step, the miner propagates the block with a valid PoW solution to the blockchain network in order for nodes to reach a consensus on such a block. The time it takes for nodes to reach consensus on a block of size $b_{s,art}$, namely block propagation delay τ_p, is proven to be a linear function $\tau_p(b_{s,art})$ of the block size, hence of the transactions included in the block [15–17,20]. From a temporal standpoint, the propagation delay of a block whose size is $b_{s,art}$ is defined as follows:

$$\tau_p(b_{s,art}) = t_{p,art} - t_{m,art} \tag{5}$$

that is the difference between the end consensus instant $t_{p,art}$ and the start consensus instant $t_{m,art}$. Considering that the number of valid PoW solutions found by the whole network in a window of time follows a Poisson distribution with rate μ, the probability for a block of size $b_{s,art}$ to become orphan is modeled as follows [20,27–29]:

$$\mathcal{P}_{orphan} = 1 - e^{-\mu \cdot \tau_p(b_{s,art})}. \tag{6}$$

Therefore, the block orphaning probability decreases exponentially with a decreasing rate given by the product of the block mining rate and the block propagation delay. Thus, the larger the block mining rate, the higher the block orphaning probability. Likewise, given that the block propagation delay is a linear function of the block size [15–17,20], the more the transactions there are in the block, the higher the block orphaning probability.

In other terms, even though one miner may find the first valid PoW solution, if its mined block is large, meaning with a lot of transactions, then this block will likely get orphaned because of long latency and computations to validate it. So, successfully mining a block actually is only a necessary but not sufficient condition to obtain a block reward: a successfully mined block must also become part of the valid consensus chain, i.e., the block must not turn into an orphan.

According to Steps 1, 2, and 3, which represent three independent events, the probability that a miner with relative hash power α obtains a block reward from a block whose size is $b_{s,art}$ is the following:

$$\mathcal{P}_{reward,art} = \alpha \cdot (1 - \mathcal{P}_{orphan}) = \alpha \cdot e^{-\mu \cdot \tau_p(b_{s,art})} \tag{7}$$

Therefore, the block rewarding probability under the state-of-the-art mining strategy ($\mathcal{P}_{reward,art}$) decreases exponentially with the block propagation delay and the block mining rate. In contrast, the higher the miner's relative hash rate, the higher the miner's probability of obtaining a block reward. Note that each miner might not be willing to have limitless increase in its hash rate due to the associated financial burden [31].

Given the objective function in (1), it is straightforward to note that the state-of-the-art mining strategy maximizes the block size, thus it maximizes the block orphaning probability, hence it minimizes the block rewarding probability.

3.2. Rational Mining Strategy

In this section, we discuss the main concepts regarding the previously proposed rational mining scheme. This section is organized as follows. In Section 3.2.1, we provide the main concepts about the rational mining scheme. Later on, in Section 3.2.2, we present the two-Step method to create a block. Then, in Section 3.2.3, we provide the formula for the miner's block rewarding probability under the rational mining scheme.

3.2.1. Key Concepts

Let us begin by providing the main concepts regarding the previously proposed rational mining scheme.

The rational mining scheme aims at carefully balancing the trade-off between the block reward and the block orphaning probability. To achieve this goal, the strategy indicates that the transactions should be selected with the objective to maximize, rather than the mere block reward as for the state-of-the-art strategy, the *expected* block reward. We compute the expected block reward $\hat{\mathcal{R}}$ as a function of the block reward and the block orphaning probability:

$$\begin{aligned}\hat{\mathcal{R}} &= \mathcal{R} \cdot (1 - \mathcal{P}_{orphan}) = \mathcal{R} \cdot e^{-\mu \cdot \tau_p(b_s)} \\ &= (\mathcal{R}_{static} + \sum_{tx \in b} \mathrm{tx}_{fee}) \cdot e^{-\mu \cdot \tau_p(b_s)}\end{aligned} \tag{8}$$

According to (8), to compute the expected block reward for a block of size b_s, a miner must be aware of the corresponding block propagation delay $\tau_p(b_s)$, e.g., through empirical data it has on block propagation delay versus block size [17,24]. Based on the above-mentioned formulation, the rational mining strategy aims at solving the following constrained optimization problem:

$$\begin{cases} \max \hat{\mathcal{R}} = \max \left[(\mathcal{R}_{static} + \sum_{tx \in b} \mathrm{tx}_{fee}) \cdot e^{-\mu \cdot \tau_p(b_s)} \right] \\ \sum_{tx \in b} \mathrm{tx}_s \leq {b_s}^{max} \end{cases} \tag{9}$$

For each block, the rational strategy aims at ensuring that the miner's block rewarding probability $\mathcal{P}_{reward,rat}$ is higher than the corresponding miner's probability under the state-of-the-art strategy $\mathcal{P}_{reward,art}$:

$$\mathcal{P}_{reward,rat} > \mathcal{P}_{reward,art} \tag{10}$$

The strategy is also flexible as it gives the miner a degree of freedom about how much larger $\mathcal{P}_{reward,rat}$ should be.

3.2.2. Block Creation Method: Two-Step Approach

We now provide a high-level overview of the method to create a block under the rational mining strategy. With regards to the rational mining strategy, block creation consists of two different steps, namely Step 1a and Step 1b (Figure 2).

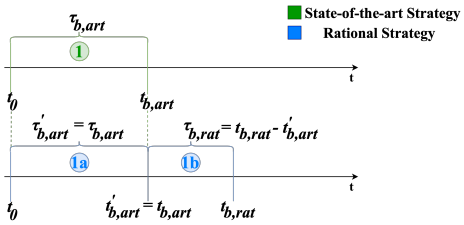

Figure 2. Time steps in order to create a block in the proposed mining strategy (in blue) as compared to the state-of-the-art mining strategy (in green).

In Step 1a, the miner behaves pretty much the same way as in Step 1 of the state-of-the-art strategy (Section 3.1). By decreasing revenue (e.g., fee rate), it extracts as many transactions as possible from the transaction pool until one of the following occurs: (i) The block size limit is reached; (ii) There are no more transactions that can fit into the block. However, as opposed to the state-of-the-art strategy, instead of including the transactions in a block, it places them in a temporary data structure in the same extraction order. Let $|tmp\_struct| > 0$ be the cardinality of the temporary data structure (i.e., the number of transactions in it), where $|\cdot|$ denotes the cardinality operator. So, the time required to carry out Step 1a is equivalent to the block creation time of the state-of-the-art strategy (Step 1). It is the following:

$$\tau'_{b,art} = \tau_{b,art} = t_{b,art} - t_0 = t'_{b,art} - t_0 \tag{11}$$

Through Step 1a, the miner gains knowledge of the transactions it would include in a block if the state-of-the-art strategy was followed, and also of the size of the correspondent block $b_{s,art}$. From this, the miner can estimate the block propagation delay $\tau_p(b_{s,art})$, e.g., through empirical data it has on block propagation delay versus block size [17,24]. By the end of Step 1a, the miner becomes aware of the block rewarding probability under the state-of-the-art strategy (Equation (7)). In Step 1b, consistent with the optimization problem objective function in (9), the miner selects and includes in a block only the transactions that maximize $\hat{\mathcal{R}}$ among those sitting in the temporary data structure. To achieve this goal, a miner is free to run any transaction selection algorithm of its choice that adheres to such a rational mining scheme, i.e., maximizing the expected block reward.

The miner runs the transaction selection algorithm for a certain amount of time based on how much larger the miner wants $\mathcal{P}_{reward,rat}$ to be with respect to $\mathcal{P}_{reward,art}$ (10) or once all the transactions in the temporary data structure have been processed. Particularly, regarding the inequality defined in (10), the rational scheme allows the miner to arbitrarily choose how much larger $\mathcal{P}_{reward,rat}$ should be with respect to $\mathcal{P}_{reward,art}$. The larger it is, the lower the block reward as compared to the one of the state-of-the-art strategy. This is because fewer transactions can be included in the block, and therefore, the lower the reward coming from fees. To adjust the trade-off between the block rewards of the two mining strategies and to ensure that (10) is satisfied, a miner periodically checks a stop condition set on purpose. According to this condition, a miner should immediately stop running the transaction selection algorithm as soon as the percentage change in block rewarding probability gets below (or equal to) a miner defined threshold ($\mathcal{P}^*$):

$$\frac{\mathcal{P}_{reward,rat} - \mathcal{P}_{reward,art}}{\mathcal{P}_{reward,art}} \leq \mathcal{P}^* \quad \mathcal{P}^* \in (0, +\infty) \tag{12}$$

The stop condition defined in (12) is thus the place where the miner leverages the information about the block rewarding probability under the state-of-the-art strategy (Equation (7)). A miner could check (12) at different time intervals, depending on the specific transaction selection algorithm executed, which is equivalent to saying that the

length of the period to check the stop condition can be set flexibly according to the specific transaction selection algorithm.

In the end, the miner should stop including new transactions in a block as soon as Inequality (12) is satisfied or once all the transactions in the temporary data structure have been processed. We indicate with $b_{s,rat}$ the size of the block created according to the rational mining strategy. As to the time it takes to execute the transaction selection algorithm, i.e., the duration of Step 1b, we indicate it with $\tau_{b,rat}$, which is the difference between the end block creation instant $t_{b,rat}$ and the start block creation instant $t'_{b,art}$. Recall that the latter corresponds to the state-of-the-art strategy's end block creation instant $t_{b,art}$. Note that when Inequality (12) is checked, $t_{b,rat}$ corresponds to the current time, i.e., the time when such an inequality is tested, and hence with the block comprising the transactions selected and included up until that time. The missing piece is $\mathcal{P}_{reward,rat}$. We discuss this in the next section.

3.2.3. Block Rewarding Probability

To determine the miner's block rewarding probability under the rational mining strategy $\mathcal{P}_{reward,rat}$ we need to assess whether and how Step 1b affects it. If this is not the case, then $\mathcal{P}_{reward,rat} = \mathcal{P}_{reward,art}$. One way to test whether or not Step 1b affects $\mathcal{P}_{reward,rat}$ is by fairly comparing the rational with the state-of-the-art mining strategy and looking for reward-related implications. To make the comparison fair, let us consider the following scenarios: (a) The two strategies lead to the same block size; (b) The mining time is the same in the two strategies. It follows that the duration of Steps 2 and 3 is the same in the strategies. Furthermore, we know that the duration of steps 1 and 1a is the same (Section 3.2.2). Under these assumptions, the end consensus time instant of the rational strategy $t_{p,rat}$ is shifted more to the right with respect to the state-of-the-art strategy's end consensus time instant $t_{p,art}$. Specifically, it is shifted to the right exactly by a time interval equal to $\tau_{b,rat}$, i.e., by the duration of Step 1b. Therefore, in case of a fork, the block created according to the rational strategy has less chance to end up in the valid consensus chain. This is because, assuming the length of the two chains is the same (fair comparison), miners follow the heuristic of extending the branch with the block they heard first. This means that Step 1b has actually reward-related implications. Figure 3 depicts the described scenario.

We need now to assess how Step 1b actually affects the miner's block rewarding probability under the rational strategy. Intuitively, $\mathcal{P}_{reward,rat}$ should be a function of $\tau_{b,rat}$. In general, $\tau_{b,rat}$ can be interpreted as the time between the creation of two subsequent blocks between any two miners in the network. Note, however, that there is no means for a miner to know whenever another miner in the network has created a block until that block is eventually received. Thus, it is necessary to examine $\tau_{b,rat}$ from the individual miner's perspective. From this standpoint, there are two blocks involved: (i) On the one hand, there is the block the miner would create by $t'_{b,art}$ if it employed the state-of-the-art strategy; (ii) On the other hand, there is the block that the miner would create by $t_{b,rat}$ if it employed the rational strategy starting at the time instant $t'_{b,art}$. Stated in other terms, we are seeking the probability of not creating the next block until a $\tau_{b,rat}$ time interval from $t'_{b,art}$.

We reason as follows. Let E ={"creation of a block"} be the event of interest. Moreover, let M be the average number of miners in the network. Actually, because of pool mining, M can be described as the number of up-and-running mining pool servers in the network. As already anticipated, each pool server coordinates the activities of all the miners in that pool and performs the selection of transactions to be part of the next block. Note that solo miners are by far an exception nowadays: compared to mining pools, solo miners retain an insignificant percentage of the total hash rate in the network. Let $\frac{1}{\mu}$ be the average block mining time. Assume that $\tau_p(\hat{b_s})$ is the propagation delay of the average size block.

Finally, assume $T = \frac{1}{\mu} + \tau_p(\hat{b_s})$ to be a time window. The experiment consists of counting the number of occurrences of the event E over a period of time T. We observe a situation that has the following characteristics: (a) The number of created blocks is random; (b) The creation of a block does not influence the chance to create another block; (c) The average number of created blocks per time window is $\lambda = \frac{M}{T}$; (d) The likelihood of creating a block in a given time window is the same for all the time windows.

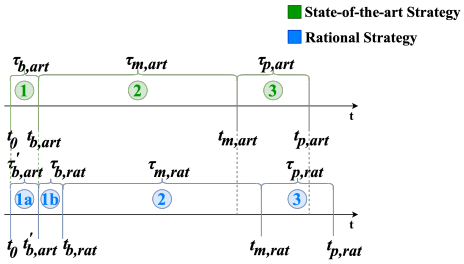

Figure 3. Fairly comparison between the proposed (blue) and the state-of-the-art (green) mining strategies.

Because of the above properties, we model the number of blocks created during a T time window with a Poisson distribution (we are modeling a count). We are interested in the probability distribution of the time until the next block is created. By definition, the probability distribution of the wait time until the next event occurs in a Poisson process is an exponential distribution. Let $\mathcal{X}$ be the random variable modeling the wait time until the next block is created (from any starting point). The probability of not creating a block until a $\tau_{b,rat}$ time interval is computed as follows:

$$\begin{aligned} \mathcal{P}(\mathcal{X} > \tau_{b,rat}) &= \mathcal{P}(\text{no block created before } \tau_{b,rat} \text{ time}) \\ &= \mathcal{P}(\mathcal{Y} = 0) \quad \mathcal{Y} \sim Poisson(\lambda \cdot \tau_{b,rat}) \\ &= \frac{e^{-\lambda \cdot \tau_{b,rat}} \cdot (\lambda \cdot \tau_{b,rat})^0}{0!} = e^{-\lambda \cdot \tau_{b,rat}} \end{aligned} \tag{13}$$

At this point, we are able to define the miner's block rewarding probability under the rational strategy. Assuming the miner relative hash rate is α, observing that Steps 1b, 2, and 3 constitute three independent events, and recalling that the size of the block created according to the rational strategy is denoted with $b_{s,rat}$, it is the following:

$$\mathcal{P}_{reward,rat} = e^{-\lambda \cdot \tau_{b,rat}} \cdot \alpha \cdot e^{-\mu \cdot \tau_p(b_{s,rat})} \tag{14}$$

Compared to the miner's block rewarding probability under the state-of-the-art mining strategy (7), Equation (14) introduces an additional negative exponential factor whose decreasing rate is given by the product of the block creation rate, namely λ, and the time it takes to execute the transaction selection algorithm, namely $\tau_{b,rat}$. Also, the size of the created block $b_{s,rat}$ is not the same as in the reference strategy $b_{s,ref}$.

Therefore, from the individual miner's perspective, for the same block size (i.e., $b_{s,rat} = b_{s,art}$), the block rewarding probability under the rational strategy is lower than the corresponding probability under the state-of-the-art strategy. Mitigation of the adverse effect coming from the additional negative exponential factor is delivered by the rational strategy of creating smaller size blocks. In practice, the rational mining strategy creates smaller size blocks by ensuring that Inequality (10) holds. In particular, the stop condition in (12), which allows (10) to hold, gives rise to blocks containing fewer transactions. Those smaller size blocks will necessarily take less time to reach consensus, meaning that, for any block, it is the following: $\tau_p(b_{s,rat}) < \tau_p(b_{s,art})$. All in all, the rational strategy thwarts the adverse effect of the additional negative exponential factor by decreasing the rate of the right-most exponential.

4. PROACTION Algorithm

We now present the implementation of the rational mining scheme. Specifically, we walk through the implementation of the block creation method undertaken in the rational mining scheme. The method comprises 2 steps (Section 3.2): (i) A common first step, which is aimed to fill in a (temporary) data structure by the same transactions the miner would include in a block if the state-of-the-art mining strategy was followed; (ii) The transaction selection algorithm itself, which is aimed to select and include in the block the subset of transactions, among the ones sitting in the (temporary) data structure, that maximizes the expected block reward. In the following, we investigate the two steps separately. In Section 4.1, we examine the first step. Then, in Section 4.2, we go deeper into the second Step by presenting and analyzing PROACTION.

4.1. Implementation of First Step

4.1.1. Discussion

We discuss the implementation of the block creation method's first step. We take the pseudo-code description presented in Algorithm 1 as a reference.

The algorithm takes as input the transaction pool of cardinality $\mathcal{K}$. Because of the non-negligible computational cost related to the knapsack problem [32], we still assume that the miner uses a max heap to maintain the transaction pool.

At first, the algorithm initializes the temporary data structure tmp_struct aimed to contain those same transactions the miner would include in a block if the state-of-the-art mining strategy was followed (line 1). The algorithm also initializes a couple of variables, tmp_struct_s and $t_{b,art}$ (line 2 and 3). The former variable accounts for the size of the temporary data structure, that is for the size of the block created if the state-of-the-art mining strategy was followed, i.e., the sum of the size of the transactions sitting in the temporary data structure plus the block header size. Thus, tmp_struct_s is initialized to the block header size η. Regarding $t_{b,art}$, which is initialized to 0, it accounts for the time instant in which the execution of the first Step terminates. This corresponds to the end block creation instant if the state-of-the-art mining strategy is followed.

Next, the algorithm scans, in an ordered fashion, the transaction pool and includes the transactions in tmp_struct until either the block size limit is reached or there are no more transactions that can fit in a block (line 4 to 9). Every time the algorithm includes a new transaction in tmp_struct, it also increments tmp_struct_s by the corresponding transaction size value. All in all, apart from including the transactions in tmp_struct, rather than in a block data structure, a miner behaves the same way as in the state-of-the-art mining strategy.

The algorithm stores the current time in $t_{b,art}$ (line 10). Finally, the algorithm returns: (a) tmp_struct containing the transactions to be potentially selected for inclusion in the block, which is a matter of the transaction selection algorithm (second step); and (b) $t_{b,art}$, whose value is going to be exploited by the transaction selection algorithm (second step).

4.1.2. Computational Complexity Analysis

We explore the time complexity of the first Step of the block creation method undertaken in the rational mining scheme. We can distinguish between two cases.

When the transaction pool is empty ($\mathcal{K} = 0$), i.e., the trivial case, it follows that tmp_struct consists of the empty structure. In this case, the time complexity of the first Step is $\mathcal{O}(1)$.

When the transaction pool is non-empty ($\mathcal{K} > 0$), i.e., the non-trivial case, it turns out that tmp_struct contains a subset of all the transactions in the transaction set. In the worst case, tmp_struct contains all the $\mathcal{K}$ transactions, meaning that $|tmp\_struct| = \mathcal{K}$. Thus, in the worst case, the complexity of the first task is $\mathcal{O}(\mathcal{K})$. Instead, when the volume of traffic

in the network is not negligible, due to the constraint given by the block size limit, the number of transactions included in tmp_struct is $\overline{\mathcal{K}} < \mathcal{K} \mid \overline{\mathcal{K}} = \alpha \cdot \mathcal{K}, \alpha \in (0,1)$. In this scenario, it is $|tmp\_struct| = \overline{\mathcal{K}}$.

Algorithm 1: Block Creation Method's First Step

Data:
- Transaction pool (tx) of cardinality $\mathcal{K}$ sorted by decreasing revenue (i.e., fee or fee rate)

Result:
- Temporary data structure (tmp_struct) with the transactions to be potentially selected for inclusion in the block
- Time instant ($t_{b,art}$) in which the first step's execution terminates

```
1  tmp_struct ← ∅
2  tmp_struct_s ← η
3  t_b,art ← 0
4  for (i = 0; i < K; i = i + 1) do
5      if tmp_struct_s + tx[i]_s ≤ b_s^max then
6          add_tx_to_tmp(tx[i], tmp_struct)
7          tmp_struct_s = tmp_struct_s + tx[i]_s
       end
8      if tmp_struct_s = b_s^max then
9          break
       end
   end
10 t_b,art ← now()
11 return tmp_struct, t_b,art
```

4.2. Implementation of Second Step: PROACTION

4.2.1. Discussion

We comment on the implementation of PROACTION, the first transaction selection algorithm we propose in this work. We take the pseudo-code description in Algorithm 2 as a reference.

PROACTION takes as input the temporary data structure tmp_struct containing the transactions to be potentially selected for inclusion in the block to mine, the time instant $t_{b,art}$ in which the block creation method's first-Step execution terminated, and the miner defined threshold $\mathcal{P}^*$.

At the beginning, PROACTION initializes the block data structure b_{rat} aimed to contain the subset of selected transactions among the ones sitting in tmp_struct. At the end of PROACTION's execution, b_{rat} is the block to mine. Also, PROACTION initializes the variable $b_{s,rat}$ (line 2). This variable accounts for the size of the block b_{rat}. Initially, the block consists of the header only; therefore, its size is initialized to η, i.e., the block header size.

Then, PROACTION stores the current and the new expected block reward, respectively $\hat{\mathcal{R}}_{cur}$ and $\hat{\mathcal{R}}_{new}$, according to the Equation (8) in Section 3.2.1. This is accomplished by means of the function "expected_block_reward($\cdot,\cdot,\cdot$)". This function takes in three parameters: the next transaction to consider in the computation of the expected block reward, the block b_{rat} created up until that time, and the size $b_{s,rat}$ of such a block. The purpose of the three parameters is quite evident, namely, to compute the new expected block reward based on the current expected block reward value and the size and the fee of the considered transaction. The first time the above function is invoked (line 3), PROACTION initializes $\hat{\mathcal{R}}_{cur}$ and $\hat{\mathcal{R}}_{new}$ to the exact same value, which is computed by considering no transaction (first argument), the empty block (second argument), and the only block header size (third argument). Indeed, it might be that the expected block reward is maximum when no transaction is part of the block.

Moving on from the above, we note that PROACTION scans tmp_struct following the order in which transactions were included (line 4). PROACTION has a *fine-grained* round. This is because, at each round, PROACTION processes just the *single* next transaction in tmp_struct and computes the new expected block reward (line 5). If (and only if) the considered transaction is such that $\hat{\mathcal{R}}_{new} > \hat{\mathcal{R}}_{cur}$, then the transaction is included in the block b_{rat} (line 6 and 7). Moreover, in this case, $b_{s,rat}$ is incremented by the size of the included transaction and $\hat{\mathcal{R}}_{cur}$ is set to $\hat{\mathcal{R}}_{new}$ (line 8 and 9). So, there are (at most) as many rounds as transactions in tmp_struct.

Algorithm 2: Block Creation Method's Second Step—PROACTION

Data:
- Temporary data structure (tmp_struct) with the transactions to be potentially selected for inclusion in the block
- Time instant ($t_{b,art}$) in which the first step's execution terminated
- Miner defined threshold ($\mathcal{P}^*$)

Result: Block to mine (b_{rat}) created by PROACTION

1 $b_{rat} \leftarrow \varnothing$
2 $b_{s,rat} \leftarrow \eta$
3 $\hat{\mathcal{R}}_{cur} \leftarrow \hat{\mathcal{R}}_{new} \leftarrow$ expected_block_reward$(\varnothing, b_{rat}, b_{s,rat})$
4 **for** $(i = 0; i < |tmp\_struct|; i = i + 1)$ **do**
5 $\quad \hat{\mathcal{R}}_{new} \leftarrow$ expected_block_reward$(tmp_struct[i], b_{rat}, b_{s,rat})$
6 $\quad$ **if** $\hat{\mathcal{R}}_{new} > \hat{\mathcal{R}}_{cur}$ **then**
7 $\quad\quad$ include_tx_in_the_block$(tmp_struct[i], b_{rat})$
8 $\quad\quad b_{s,rat} = b_{s,rat} + tmp\_struct[i]_s$
9 $\quad\quad \hat{\mathcal{R}}_{cur} \leftarrow \hat{\mathcal{R}}_{new}$
$\quad$ **end**
10 $\quad \tau_{b,rat} \leftarrow now() - t_{b,art}$
11 $\quad \overline{\mathcal{P}} \leftarrow$ change_in_reward_prob$(tmp_struct_s, b_{s,rat}, \tau_{b,rat})$
12 $\quad$ **if** $\overline{\mathcal{P}} \leq \mathcal{P}^*$ **then**
13 $\quad\quad$ **return** b_{rat}
$\quad$ **end**
end
14 **return** b_{rat}

PROACTION's execution lasts until all the transactions in tmp_struct have been processed or until the percentage change in block rewarding probability gets below (or equal to) the miner defined threshold $\mathcal{P}^*$ (Section 3.2.2). About the latter stop condition, a miner checks it with a period length equivalent to the time it takes to execute one single round. Given that each round consists of processing *just the single next* transaction, such a stop condition is checked just after processing each transaction.

Particularly, the change in block rewarding probability is computed by means of the corresponding function (line 11). This function takes in three parameters: tmp_struct_s, $b_{s,rat}$, and PROACTION's current execution time interval $\tau_{b,rat}$. As an aside, the latter parameter is obtained (line 10) as the difference between the current execution time instant and PROACTION's starting execution time instant, namely $t_{b,art}$.

Given empirical data on block propagation delay versus block size, the function's first parameter is all that is needed to compute $\mathcal{P}_{reward,art}$ (Equation (7)). The same applies to $\mathcal{P}_{reward,rat}$ as far as the second and third parameters are concerned (Equation (14)). If the computed value $\overline{\mathcal{P}}$ is lower than (or equal to) the miner defined threshold $\mathcal{P}^*$ (line 12), PROACTION's execution stops and the block b_{rat} is returned (line 13). Otherwise, PROACTION's execution goes on to process the next transaction. Eventually, PROACTION's execution terminates, and the block b_{rat} is returned (line 14).

4.2.2. Computational Complexity Analysis

We investigate PROACTION's time complexity. We can distinguish between two cases. When tmp_struct consists of the empty structure ($|tmp\_struct| = 0$), i.e., the trivial case, PROACTION's time complexity is $\mathcal{O}(1)$.

When tmp_struct is non-empty ($|tmp\_struct| > 0$), i.e., the non-trivial case, in the worst-case scenario, PROACTION's time complexity is linear with the number of transactions sitting in tmp_struct, and thus it is $\mathcal{O}(|tmp\_struct|)$. We now analytically prove the above claims regarding PROACTION's time complexity. The cardinality of the solution space $\mathcal{S}$ of all combinations of $|tmp\_struct|$ transactions taken $i = 1, ..., |tmp\_struct|$ at a time, namely $C(|tmp\_struct|, i)$, which a miner could include in b_{rat} by running PROACTION, is as follows: $|\mathcal{S}| = \sum_{i=1}^{|tmp\_struct|} C(|tmp\_struct|, i) = 2^{|tmp\_struct|} - 1$. Let $\mathcal{E}$ be the exploration space, that is space of transactions explored during PROACTION's execution. Let $\mathcal{E}_i$ be the exploration space at round $i, i = 1, ..., |tmp\_struct|$. Given that, at each round, one single transaction is processed, it is the following: $|\mathcal{E}_i| = 1 \ \ \forall i, i = 1, ..., |tmp\_struct|$. Therefore, in the *worst-case* scenario, it is as follows: $\mathcal{E} = \bigcup_{i=1}^{|tmp\_struct|} \mathcal{E}_i$. This means that the following holds:

$$|\mathcal{E}| = \sum_{i=1}^{|tmp\_struct|} |\mathcal{E}_i| = \sum_{i=1}^{|tmp\_struct|} 1 = |tmp\_struct| \tag{15}$$

This confirms that, in the *worst-case* scenario, PROACTION's time complexity actually increases linearly with the cardinality of the temporary data structure. On the other hand, note that the cardinality of the solution space $\mathcal{S}$ grows exponentially with the cardinality of tmp_struct, i.e., $|\mathcal{S}| \in \mathcal{O}(2^{|tmp\_struct|})$. Therefore, even in the *worst-case* scenario, it is $|\mathcal{E}| \ll |\mathcal{S}|$.

5. Performance Evaluation

To demonstrate the functionality of the proposed transaction selection algorithm, we proceed with simulation [33,34]. Specifically, we consider the *BlockSim* simulator [35,36], one of the top 5, state-of-the-art blockchain simulators [37,38], and extend it by implementing the rational mining scheme in accordance to Algorithms 1 and 2. Moreover, a new metric representing the *total reward per block* (i.e., the sum of the static block reward and transaction fees for each block) has been added, and the parametrization of the input data has been aligned to a more recent version of the Bitcoin network. The BlockSim model of the Bitcoin protocol being extended has been validated in one of our previous works [17]. In that work, we determined a realistic model of the Bitcoin protocol, including the block propagation process and exploiting about 2 years of real-world Bitcoin data from 1 January 2020 to 30 November 2021. The modified simulator's code and raw data are publicly available at [39].

We experimentally validate that the rational mining scheme is effective, i.e., it helps to solve the issue underlying the state-of-the-art mining strategy, which is the maximization of the block orphaning probability (Section 3.2.1). Specifically, we evaluate the impact of the rational mining strategy on Bitcoin performance evaluation metrics, namely network throughput, long-term block reward, and orphan block count, assuming a variable random percentage β of winning miners employing PROACTION. So, we perform a long-term comparison between the rational mining strategy's impact and that of the state-of-the-art mining strategy. For the comparison to be fair, we consider the same percentage β and those same winning miners following the state-of-the-art mining scheme. Additionally, we assume the transaction pool to be the same in the two mining strategies.

The reason why we focus on Bitcoin rather than any other PoW-based blockchain protocol follows. Due to the extremely long block mining time (i.e., on average 600 s), Bitcoin's block orphaning probability is extremely low. We exploit this property of Bitcoin to perform a worst-case analysis, meaning that we use Bitcoin to establish a lower bound for the impact of the rational mining strategy powered by PROACTION on PoW-based blockchain protocols.

To compute the expected block reward, the block propagation delay corresponding to the current block size must be known. In this regard, it is proven the block propagation delay is a linear function of the block size [15–17,20]. This is where the *linear* regression model developed in one of our preliminary paper [17] comes into play. The regression model is based on almost 2 years of real-world Bitcoin block size and block delay data. The model is able to estimate the Bitcoin median block propagation delay given the Bitcoin block size. Note that the 50th percentile of block propagation delay (i.e., the median) is the time it takes for a block to reach 50% of the network from the generation node. Thus, it is also the minimum time it takes for the majority of network nodes to reach consensus on a block [20,30]. As with any regression model, it is *parametric*. So, the slope and the offset of the regression line can be updated in time, adding new observations to the original set, possibly weighting older observations differently from new ones or, alternatively, eliminating them and iterating the procedure as shown in [17]. Note that the more accurate the regression model the better it is from a reward perspective. Miners have full freedom regarding how frequently the model is updated, which, therefore, improves the coefficient of determination. Intuitively, it is a(n inevitable) trade-off between accuracy of the estimation and computation and memory overhead. For full details about the regression model please refer to [17]. The rest of this section is organized as follows. In Section 5.1, we present BlockSim's experimental parametrization. In Section 5.2, we discuss validation results.

5.1. Simulator Parametrization

Validation dictates BlockSim's input parameters to reflect the status of the real-world Bitcoin blockchain. In this section, we present the BlockSim's experimental parametrization. Values for the parameters are statistics computed on real-world Bitcoin blockchain data. We gathered an entire month of data. In Table 1, we present the resulting parametrization. The transaction fees and sizes in the simulation are random, and they follow an exponential distribution having the mean values shown in the table.

Each of the statistics in Table 1 is computed as the sample mean of the data. A couple of remarks about the methodology applied to compute some of the statistics. As to the transaction rate, it is computed from [40] as the mean number of unique INVentory (INV) entries [41] received by a Bitcoin node per second. Concerning the Node Count parameter's value, it is computed from [40] as the mean number of reachable Bitcoin nodes. In this regard, to identify Bitcoin mining nodes, we analyze each raw block data structure. We identify 17 unique mining pools and, thus, just as many unique mining nodes (excluding replica ones). Thus, we compute each mining node's hash rate as the node's share of blocks generated during the month. Lastly, the details regarding the simulated block propagation process. First, the block propagation delay is estimated by means of real-world Bitcoin block delay data. Furthermore, the block delay is computed on a per-simulated-block basis. Last but not least, the simulated nodes block receiving time is computed based on the percentile function of the exponential distribution, which is the actual Bitcoin block delay probability distribution [40].

Table 1. Experimental parametrization of the BlockSim blockchain simulator.

Parameter	Value
Block Size	1.497337 MB
Block Weight	3.989249 MWU
Tx Rate	5.225 tx/s
Average Tx Fee	0.00000451 BTC
Average Tx Size	0.00000455 MB
Node Count	8912

With respect to the miner defined threshold ($\mathcal{P}^*$), we do not assign any value, that is, we let PROACTION execute until all the transactions in the temporary data structure have been processed. Setting no threshold allows the difference between the block rewards of the rational and the state-of-the-art mining strategies to be at a minimum. Yet, such a setting also maximizes the time interval it takes to execute the transaction selection algorithm, which exponentially decreases the block rewarding probability under the rational mining strategy (Section 3.2.3).

Based on the number of transactions in the temporary data structure and the time complexity of PROACTION, such a setting for $\mathcal{P}^*$ may or may not turn the rational mining scheme to be less effective than the state-of-the-art strategy. If the former is actually the case, a value for the miner threshold $\mathcal{P}^*$ will then be necessary. Additionally, if the former is actually the case, a value for the (average) block creation rate (λ) will also be needed. This is required to compute the block rewarding probability under the rational strategy, which in turn is required to check whether the percentage change in block rewarding probability is lower than (or equal to) to $\mathcal{P}^*$. Recalling that $\lambda = \frac{M}{T}$ (Section 3.2.3), estimating the number of Bitcoin pool servers M can be accomplished in the same manner described in [42]. Estimating T is straightforward by leveraging empirical data [40,43].

5.2. Validation Results and Discussion

We are interested in the trend of the expected block reward and in the following blockchain performance evaluation metrics, which are evaluated as a function of the percentage β of winning mining nodes employing PROACTION: (i) *Mean network throughput*, i.e., the mean number of transactions per block; (ii) *Mean percentage change in throughput*, i.e., the mean percentage variation in network throughput between the state-of-the-art mining strategy and the rational mining scheme; (iii) *Yearly mean gain reward*, i.e., the yearly difference between the block reward under the rational mining strategy and the state-of-the-art strategy's one; (iv) *Yearly mean percentage of gain reward*, i.e., the yearly mean percentage difference in block reward between the state-of-the-art mining strategy and the rational mining scheme; (v) *Yearly mean orphan blocks produced by the β winning mining nodes*.

We aim to evaluate the effectiveness of the rational mining strategy powered by PROACTION in the very long term, i.e., when the time goes to infinity. Consequently, we simulated 6 months of activity (i.e., 180 days), and we performed 40 replicas of each day. Therefore, we performed a total of 7200 simulations, each simulating 1 day, spanning a total period of about 20 years.

We start by considering a scenario in which, to create a block, a miner employs the rational mining strategy powered by PROACTION. We compare this scenario with another one whereby the same miner employs a state-of-the-art mining strategy. We investigate the trend of the expected block reward as a function of the number of transactions included in the block created by following each of the two mining schemes.

Intuitively, under the rational mining strategy, the trend of the expected block reward should monotonically increase. Moreover, the number of included transactions should

be less than or, at most, equal to the number of transactions included under the state-of-the-art mining strategy. As to the latter strategy, the expected block reward should not be a monotonic function. Let us verify the intuition. Worthy of particular attention are the curves' peak (Figure 4).

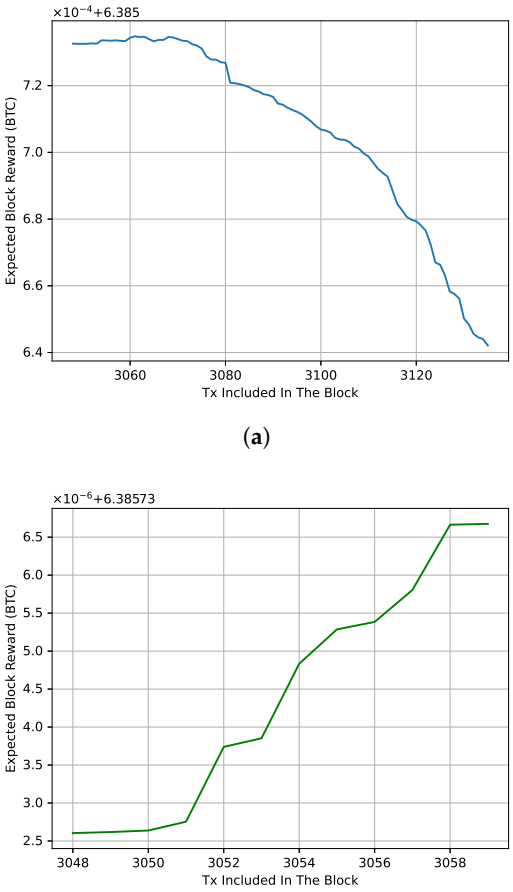

Figure 4. The expected block reward vs. the transaction no. included in the block as a function of the employed mining strategy. We zoom in on the curve's peak. (**a**) State-of-the-art Mining Strategy. (**b**) Rational Mining Strategy powered by PROACTION.

In the case of the state-of-the-art mining strategy (Figure 4a), after some initial fluctuations, a decrease in the expected block reward is appreciable at the right-hand side of the absolute maxima of the curve. On the contrary, in the case of the rational mining strategy and the PROACTION transaction selection algorithm (Figure 4b), the expected block reward is indeed a monotonic increasing function. Last but not least, the number of included transactions is slightly smaller than the one under the state-of-the-art scheme. This is all consistent with the early intuition.

We now present and discuss the simulation results as far as the relevant blockchain performance evaluation metrics introduced above are concerned. Figure 5 summarizes the key findings. Simulation results are shown as mean values and 95%-confidence intervals.

Figure 5a shows the mean network throughput. In general, Bitcoin mean network throughput decreases linearly as the percentage β of winning mining adopting the rational mining strategy powered by PROACTION increases. When $\beta = 0$, all the winning mining nodes follow the state-of-the-art mining strategy. In this case, the average number of Bitcoin transactions per block resembles the value in Figure 4a. Conversely, when $\beta = 100\%$, all the winning mining nodes follow the rational mining strategy powered by PROACTION.

In this other case, instead, the average number of Bitcoin transactions per block resembles the value in Figure 4b, which is just slightly smaller, i.e., less than a hundred transactions.

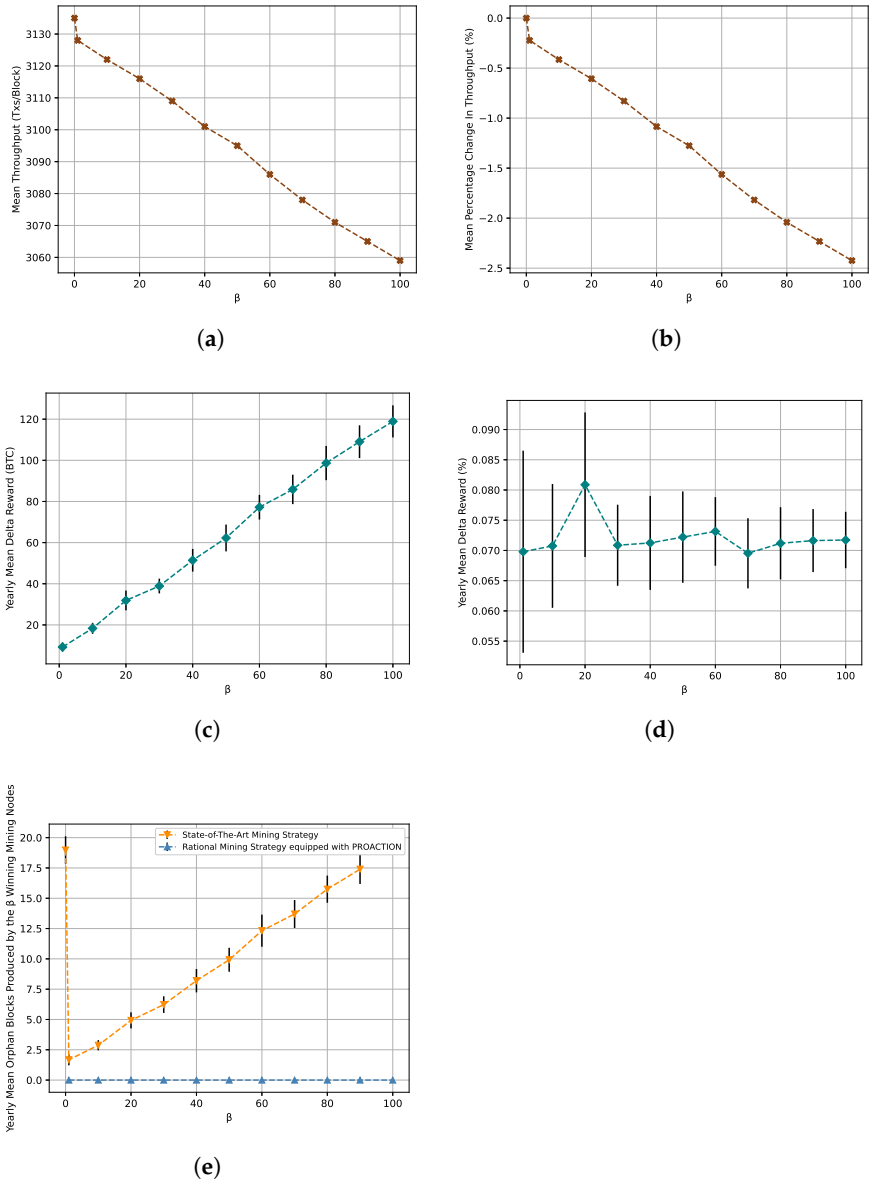

Figure 5. Simulation results as a function of the percentage of winning mining nodes (β) employing the rational mining strategy powered by PROACTION vs. the same percentage (β) and those same winning mining nodes following the state-of-the-art mining strategy in the context of the Bitcoin protocol. (**a**) Mean network throughput. (**b**) Mean percentage change in throughput. (**c**) Yearly mean reward gain. (**d**) Yearly percentage mean reward gauin. (**e**) Yearly mean prphan blocks produced by β winning miners.

In percentage terms (Figure 5b), when $\beta = 100\%$, the Bitcoin mean network throughput is only about 2.42% lower than the state-of-the-art mining strategy. This, of course, could cause a degradation of the user experience in terms of the average confirmation delay. However, this degradation should be limited as it is proportional to the 2.42% throughput reduction. Therefore, in the context of Bitcoin, as far as network throughput is concerned, the rational mining strategy powered by PROACTION is only barely more harmful than the state-of-the-art mining scheme.

Figure 5c shows the yearly mean gain reward. Whatever β, following the rational mining strategy powered by PROACTION leads miners to gain higher long-term rewards

than following the state-of-the-art scheme. The yearly mean gain reward increases linearly with β.

A question arises: how can it be that adopting the rational mining strategy powered by PROACTION allows miners to obtain higher long-term rewards than following the state-of-the-art scheme? The question is legitimate since we have just shown that the number of transactions per block is lower than in the state-of-the-art scheme, thus it is the fraction of block reward coming from transaction fees. The only possible way is that the *static* reward is higher than that of the state-of-the-art strategy. In turn, this is possible if (and only if) the number of orphan blocks is less than in the state-of-the-art strategy. Figure 5e proves that this is the case. In fact, the yearly mean orphan blocks produced by the β mining nodes under the rational mining strategy powered by PROACTION is always strictly lower than that of the state-of-the-art strategy.

Note that when $\beta = 0$, i.e., when all the winning mining nodes follow the state-of-the-art mining strategy, the yearly mean orphan block count (dark orange triangle in the upper left corner in Figure 5e) actually resembles the real-world Bitcoin blockchain value [44]. Therefore, if we consider the yearly mean *difference* between the orphan blocks produced by the β winning mining nodes under the state-of-the-art and the rational mining strategies, it reflects the dark orange curve in Figure 5e. This explains the linear trend of the teal curve in Figure 5c, where each diamond is the product of the Bitcoin current static reward (i.e., 6.25 BTC) by the corresponding yearly mean orphan block count (i.e., the dark orange triangles in Figure 5e).

In percentage terms (Figure 5d), the yearly mean gain reward is about 0.07%, which is consistent with the Bitcoin orphaning probability as one would expect. Consequently, from a Bitcoin miner's economic perspective, setting no value for the threshold parameter ($\mathcal{P}^*$) emerges as the most profitable choice. Stated differently, under the current Bitcoin transaction rate (i.e., an average of 5.225 transactions per second) and network throughput (i.e., in the order of a few thousand transactions per block), it is economically inconvenient to possibly stop PROACTION's execution before having elaborated all the transactions in the temporary data structure. In effect, by considering the stop condition regarding the percentage change in block rewarding probability, a miner would reduce (even more) the number of transactions included per block. This would reduce the fraction of reward per block coming from transaction fees, and so the long-term reward coming from fees. Still, this does not mean that a value for $\mathcal{P}^*$ is generally not required. Whether a value for $\mathcal{P}^*$ is required is a matter of the specific PoW consensus algorithm and of the corresponding blockchain network throughput.

The results achieved in the context of Bitcoin demonstrate that the percentage change in long-term reward is indeed in the order of the block orphaning probability. This is something expected since PROACTION is an implementation of a rational mining scheme. In general, the higher the block orphaning probability the higher the percentage change in long-term reward will be. The general idea of PROACTION is applicable to a generic PoW-based cryptocurrency, provided it experiences a non-negligible amount of orphan blocks. Of course, the effect of PROACTION on profitability, throughput, and orphaning rate should be evaluated on a case-by-case basis for different blockchains.

On PROACTION vs. Security

Hassija et al. provide a five-class classification of different blockchain attacks: blockchain network attacks, user-wallet attacks, smart contract attacks, transaction verification attacks, and mining pool attacks [45]. In order to reinforce the ideas, Hassija et al. classify Sybil attacks as blockchain network attacks and double-spending as transaction verification attacks. Proof of work (PoW) helps mitigate Sybil attacks by making

it costly and resource-intensive to control multiple identities. In a PoW system, miners must solve complex cryptographic puzzles to validate transactions and create new blocks. This requires significant computational power and energy, which makes it economically unfeasible for an attacker to create and maintain multiple identities to gain control over the network. Additionally, PoW effectively addresses the problem of double-spending by making it computationally and economically prohibitive for an attacker to alter the transaction history. The combination of chain integrity and network consensus ensures that once a transaction is confirmed, it cannot be reversed or duplicated. As to chain integrity, each block in the blockchain contains a cryptographic hash of the previous block. This chaining of blocks ensures that altering any single block would require solving the (complex) cryptographic puzzles for all subsequent blocks. This is computationally infeasible for an attacker. Regarding network consensus, the majority of the network must agree on the validity of transactions and blocks. An attacker would need to control more than 50% of the network's computational power, which is known as a 51% attack, to successfully double-spend. This level of control is extremely difficult and costly to achieve. Therefore, the transaction selection algorithm itself is not related either to Sybil attacks or to double-spending. It is the PoW mechanism that provides a robust defense against those two types of blockchain attacks. As PROACTION is solely a transaction selection algorithm, it only affects how a block is filled, and it does not concern how transactions, or blocks, are verified, propagated, and made stable. It follows that PROACTION is orthogonal to the resilience to common blockchain attacks, including double-spending and Sybil attacks.

Similar considerations are also valid for the threat of transaction malleability attacks, where an attacker exploits the flexibility in modifying transaction fields without invalidating them, leading to changes in transaction identifiers (TXIDs). While the attack does not alter the actual outcome of transactions, it can cause operational disruptions, enable fraud, and undermine trust in blockchain systems [46]. Modern blockchains, particularly Bitcoin with SegWit, have largely mitigated this issue by making transaction structures more robust.

Of course, PROACTION relies upon the miner's trustworthiness. While the incentive structure in PoW aims to align miners' interests with the network's integrity, threats like selfish mining, selective transaction inclusion (censorship), and miner extractable value highlight the potential for abuse [47–49]. To mitigate these issues, blockchain systems must continually evolve through technical upgrades, decentralization efforts, and robust community governance.

6. Conclusions

With reference to proof-of-work (PoW) consensus algorithms, we have presented PROACTION, a transaction selection algorithm that implements our previously proposed PoW-based rational mining strategy [20]. We demonstrated PROACTION's feasibility by properly extending BlockSim, a state-of-the-art Bitcoin simulator [35,36].

We have also experimentally validated that the rational mining scheme is effective, i.e., it carefully balances the trade-off between the risk of block orphaning and the block reward. So, we experimentally proved that it solves the issue underlying the state-of-the-art mining strategy, i.e., the maximization of the risk of block orphaning. We evaluated the impact of the rational mining scheme powered by PROACTION on Bitcoin blockchain performance evaluation metrics, such as network throughput, long-term block reward, and orphan block count. The simulations showed that, in the long term, the mean orphan blocks produced under the rational mining strategy are always strictly lower than those produced under the state-of-the-art strategy.

In addition, the experimental results indicated that when miners execute PROACTION, they gain higher long-term rewards, regardless of the value of β. The results show that the percentage improvement in long-term reward is in the order of the block orphan probability. Experiments also show that the integration of PROACTION in a Bitcoin protocol implementation is straightforward and does not require any kind of fork.

As a final consideration, PROACTION is of great interest for any of the 167+ PoW-based blockchain protocols, which have significant block orphaning because the higher the block orphaning probability, the higher the percentage improvement in the long-term reward. Of course, PROACTION is applicable together with layer-2 solutions, like the Lightning Network or sidechains, because it only affects the transaction selection process, which is orthogonal to such technologies. In future work, we plan to evaluate PROACTION with different blockchains and with different network latencies, which vary over time or node-by-node.

Author Contributions: Conceptualization, M.B., G.N., P.P. and G.D.; methodology, M.B., G.N., P.P. and G.D.; software, M.B.; validation, M.B., G.N., P.P. and G.D.; formal analysis, M.B., G.N., P.P. and G.D.; investigation, M.B., G.N., P.P. and G.D.; resources, G.D.; data curation, M.B., G.N., P.P. and G.D.; writing—original draft preparation, M.B.; writing—review and editing, P.P. and G.D.; visualization, M.B., G.N., P.P. and G.D.; supervision, G.D.; project administration, G.D.; funding acquisition, P.P. and G.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been partially financed by the Italian Ministry of Education and Research (MUR) within the framework of the Forelab project (Departments of Excellence), by the European Union-NextGenerationEU within the National Sustainable Mobility Center (CN00000023, Italian Ministry of University and Research Decree n. 1033-17/06/2022, Spoke 10), and by the European Union-NextGenerationEU within the MUR National Recovery and Resilience Plan project SERICS (PE00000014)—AQuSDIT: Advanced and Quantum-safe Solutions for Digital Identity and digital Tracing, CUP H73C22000880001.

Data Availability Statement: Data supporting reported results are in the manuscript. Any inquiry can be addressed to the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. CryptoSlate. CryptoSlate—PoW Cryptos. 2022. Available online: https://cryptoslate.com/cryptos/proof-of-work/ (accessed on 19 January 2025).
2. Albrecher, H.; Finger, D.; Goffard, P.O. Empirical risk analysis of mining a Proof-of-Work blockchain. *Decis. Econ. Financ.* **2024**, 1–28. [CrossRef]
3. Garay, J.; Kiayias, A.; Shen, Y. Proof-of-work-based consensus in expected-constant time. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt 2024), Zurich, Switzerland, 26–30 May 2024; Springer: Berlin/Heidelberg, Germany, 2024; pp. 96–125.
4. Nakai, T.; Sakurai, A.; Hironaka, S.; Shudo, K. A Formulation of the Trilemma in Proof of Work Blockchain. *IEEE Access* **2024**, *12*, 80559–80578. [CrossRef]
5. Abellán Álvarez, I.; Gramlich, V.; Sedlmeir, J. Unsealing the secrets of blockchain consensus: A systematic comparison of the formal security of proof-of-work and proof-of-stake. In Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing, Avila, Spain, 8–12 April 2024; pp. 278–287.
6. Gemajli, A.; Patel, S.; Bradford, P.G. A Low Carbon Proof-of-work Blockchain. In Proceedings of the Computer Science & Information Technology (CS & IT) Conference, London, UK, 23–24 November 2024; Volume 14.
7. Kayikci, S.; Khoshgoftaar, T.M. Blockchain meets machine learning: A survey. *J. Big Data* **2024**, *11*, 1–29. [CrossRef]
8. Li, S.N.; Campajola, C.; Tessone, C.J. Statistical detection of selfish mining in proof-of-work blockchain systems. *Sci. Rep.* **2024**, *14*, 6251. [CrossRef] [PubMed]
9. Rukhiran, M.; Boonsong, S.; Netinant, P. Sustainable Optimizing Performance and Energy Efficiency in Proof of Work Blockchain: A Multilinear Regression Approach. *Sustainability* **2024**, *16*, 1519. [CrossRef]

10. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 19 January 2025)
11. Wood, D.G. Ethereum: A Secure Decentralised Generalised Transaction Ledger. 2015. Available online: https://ethereum.github.io/yellowpaper/paper.pdf (accessed on 19 January 2025)
12. Etherscan-Ethereum Blockchain Explorer. 2022. Available online: https://etherscan.io/blocks (accessed on 19 January 2025).
13. Antonopoulos, A.M. *Mastering Bitcoin: Programming the Open Blockchain*, 2nd ed.; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2017.
14. BitcoinCoreDevelopers. Bitcoin Core. 2024. Available online: https://github.com/bitcoin/bitcoin/blob/master/src/node/miner.cpp (accessed on 19 January 2025).
15. Houy, N. The Bitcoin Mining Game. *Ledger* **2016**, *1*, 53–68. [CrossRef]
16. Robla, E.S. Analysis of Reward Strategy and Transaction Selection in Bitcoin Block Generation. Ph.D. Thesis, University of Washington, Seattle, WA, USA, 2015.
17. Basile, M.; Nardini, G.; Perazzo, P.; Dini, G. SegWit extension and improvement of the BlockSim Bitcoin simulator. In Proceedings of the 2022 IEEE International Conference on Blockchain (Blockchain), Espoo, Finland, 2–5 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 115–123.
18. Conti, M.; Sandeep Kumar, E.; Lal, C.; Ruj, S. A Survey on Security and Privacy Issues of Bitcoin. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 3416–3452. [CrossRef]
19. Decker, C.; Wattenhofer, R. Information propagation in the Bitcoin network. In Proceedings of the 2013 IEEE Thirteen Internation Conference on Peer-to-Peer Computing (P2P 2013), Trento, Italy, 9–11 September 2013; pp. 1–10. [CrossRef]
20. Basile, M.; Nardini, G.; Perazzo, P.; Dini, G. A Rational Mining Strategy for Proof-of-Work Consensus Algorithms. In Proceedings of the 2022 4th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS), Paris, France, 27–30 September 2022; pp. 59–66. [CrossRef]
21. Fiz, B.; Hommes, S.; State, R. Confirmation Delay Prediction of Transactions in the Bitcoin Network. In Proceedings of the Advances in Computer Science and Ubiquitous Computing, Taichung, Taiwan, 18–20 December 2017; Park, J.J., Loia, V., Yi, G., Sung, Y., Eds.; Springer: Singapore, 2018; pp. 534–539.
22. Pontiveros, B.B.F.; Norvill, R.; State, R. Monitoring the transaction selection policy of Bitcoin mining pools. In Proceedings of the NOMS 2018—2018 IEEE/IFIP Network Operations and Management Symposium, Taipei, Taiwan, 23–27 April 2018; pp. 1–6. [CrossRef]
23. Dos Santos, S.; Chukwuocha, C.; Kamali, S.; Thulasiram, R.K. An Efficient Miner Strategy for Selecting Cryptocurrency Transactions. In Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 14–17 May 2019. [CrossRef]
24. Rizun, P.R. A Transaction Fee Market Exists Without a Block Size Limit. 2016. Available online: https://blog.bitmex.com/wp-content/uploads/2018/01/feemarket.pdf (accessed on 19 January 2025).
25. Li, Z.; Pournaras, E. SoK: Consensus for Fair Message Ordering. *arXiv* **2024**, arXiv:2411.09981.
26. Ramezan, G.; Schneider, M.; McCann, M. A Survey on Coin Selection Algorithms in UTXO-based Blockchains. *arXiv* **2023**, arXiv:2311.01113.
27. Wang, W.; Hoang, D.T.; Hu, P.; Xiong, Z.; Niyato, D.; Wang, P.; Wen, Y.; Kim, D.I. A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks. *IEEE Access* **2019**, *7*, 22328–22370. [CrossRef]
28. Xiong, Z.; Feng, S.; Wang, W.; Niyato, D.; Wang, P.; Han, Z. Cloud/Fog Computing Resource Management and Pricing for Blockchain Networks. *IEEE Internet Things J.* **2019**, *6*, 4585–4600. [CrossRef]
29. Qiu, C.; Yao, H.; Jiang, C.; Guo, S.; Xu, F. Cloud Computing Assisted Blockchain-Enabled Internet of Things. *IEEE Trans. Cloud Comput.* **2022**, *10*, 247–257. [CrossRef]
30. Basile, M.; Nardini, G.; Perazzo, P.; Dini, G. On Improving SimBlock Blockchain Simulator. In Proceedings of the 2021 IEEE Symposium on Computers and Communications (ISCC), Athens, Greece, 5–8 September 2021; pp. 1–6. [CrossRef]
31. Poxleitner, G. Reducing Mining Costs and Value Optimization. 2016. Available online: https://dxi97tvbmhbca.cloudfront.net/upload/user/image/GPoxleitner_OperatingCostEstimationForMiners_20162019112818544344 6.pdf (accessed on 19 January 2025).
32. Toth, P. Dynamic programming algorithms for the Zero-One Knapsack Problem. *Computing* **2005**, *25*, 29–45. [CrossRef]
33. Smetanin, S.; Ometov, A.; Komarov, M.; Masek, P.; Koucheryavy, Y. Blockchain Evaluation Approaches: State-of-the-Art and Future Perspective. *Sensors* **2020**, *20*, 3358. [CrossRef] [PubMed]
34. Fan, C.; Ghaemi, S.; Khazaei, H.; Musilek, P. Performance Evaluation of Blockchain Systems: A Systematic Survey. *IEEE Access* **2020**, *8*, 126927–126950. [CrossRef]
35. Alharby, M.; van Moorsel, A. BlockSim: A Simulation Framework for Blockchain Systems. *Sigmetrics Perform. Eval. Rev.* **2019**, *46*, 135–138. [CrossRef]
36. Alharby, M.; van Moorsel, A. BlockSim: An Extensible Simulation Tool for Blockchain Systems. *Front. Blockchain* **2020**, *3*, 28. [CrossRef]

37. Paulavičius, R.; Grigaitis, S.; Filatovas, E. A Systematic Review and Empirical Analysis of Blockchain Simulators. *IEEE Access* **2021**, *9*, 38010–38028. [CrossRef]
38. Paulavičius, R.; Grigaitis, S.; Filatovas, E. An Overview and Current Status of Blockchain Simulators. In Proceedings of the 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Sydney, Australia, 3–6 May 2021; pp. 1–3. [CrossRef]
39. Basile, M.; Nardini, G.; Perazzo, P.; Dini, G. Available online: https://github.com/Unipisa/BlockSim (accessed on 19 January 2025).
40. Grundmann, M. Bitcoin Network Monitor. 2022. Available online: https://www.dsn.kastel.kit.edu/bitcoin/data/invstat.gpd (accessed on 19 January 2025).
41. Project, B. Bitcoin Developer—P2P Network. 2022. Available online: https://developer.bitcoin.org/reference/p2p_networking.html (accessed on 19 January 2025).
42. Todirica, I.A.; Le Jamtel, E. Trick or Treat? Unveil the Stratum of the Mining Pools. 2022. Available online: https://www.first.org/resources/papers/amsterdam2019/FIRST-TC-pres-v1.1.pdf (accessed on 19 January 2025).
43. Bitcoin Average Block Time. 2022. Available online: https://bitinfocharts.com/comparison/bitcoin-confirmationtime.html#alltime (accessed on 19 January 2025).
44. Fork Monitor—BTC Stale Block Candidates. 2022. Available online: https://forkmonitor.info/feeds/stale_candidates/btc.rss (accessed on 19 January 2025).
45. Hassija, V.; Zeadally, S.; Jain, I.; Tahiliani, A.; Chamola, V.; Gupta, S. Framework for determining the suitability of blockchain: Criteria and issues to consider. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4334. [CrossRef]
46. Decker, C.; Wattenhofer, R. Bitcoin transaction malleability and MtGox. In Proceedings of the 19th European Symposium on Research in Computer Security (ESORICS'14), Wroclaw, Poland, 7–11 September 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 313–326.
47. Wahrstätter, A.; Ernstberger, J.; Yaish, A.; Zhou, L.; Qin, K.; Tsuchiya, T.; Steinhorst, S.; Svetinovic, D.; Christin, N.; Barczentewicz, M.; et al. Blockchain Censorship. In Proceedings of the ACM on Web Conference 2024 (WWW'24), Singapore, 13–17 May 2024; pp. 1632–1643.
48. Madhushanie, N.; Vidanagamachchi, S.; Arachchilage, N. Selfish mining attack in blockchain: A systematic literature review. *Int. J. Inf. Secur.* **2024**, *23*, 2333–2351. [CrossRef]
49. Daian, P.; Goldfeder, S.; Kell, T.; Li, Y.; Zhao, X.; Bentov, I.; Breidenbach, L.; Juels, A. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 18–20 May 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 910–927.