

Verifier's Dilemma in Proof-of-Work Public Blockchains: A Quantitative Analysis

DARIA SMUSEVA, Department of Environmental Sciences, Informatics and Statistics, Ca' Foscari University of Venice, Venezia, Italy

ANDREA MARIN, Department of Environmental Sciences, Informatics and Statistics, Ca' Foscari University of Venice, Venezia, Italy

SABINA ROSSI, Department of Environmental Sciences, Informatics and Statistics, Ca' Foscari University of Venice, Venezia, Italy

AAD VAN MOORSEL, School of Computer Science, University of Birmingham, Birmingham, United Kingdom of Great Britain and Northern Ireland

A blockchain is an immutable ledger driven by a distributed consensus protocol. In public blockchains, such as Bitcoin and Ethereum Classic, consensus is established through a computational effort called Proof-of-Work (PoW). Special users called *miners* contribute to the PoW in exchange for a fee and also verify the data stored in blocks mined by the other miners. Here is where the Verifier's Dilemma emerges. Verification of blocks does not receive a reward, and to maximise their profits, miners may be incentivised to forego verifying blocks and to only invest their resources in PoW. In this article, we study the Verifier's Dilemma and a possible countermeasure consisting of the injection of invalid blocks using a quantitative model based on Markovian process algebra. To avoid the state space explosion problem, we study the underlying Markov chain by using a lumping that allows us to derive closed-form solutions for interesting performance indices. The analysis demonstrates the circumstances under which non-verifying miners gain fees higher than those of verifying miners. The model also allows us to derive the optimal rate at which invalid blocks must be injected so that skipping the verifying phase becomes economically disadvantageous whereas the throughput of the blockchain is only minimally reduced. The impact on miners' rewards and overall performance is also assessed.

CCS Concepts: • **Networks** → **Network performance modeling**; • **Mathematics of computing** → **Markov processes**; • **Computer systems organization** → **Peer-to-peer architectures**;

Additional Key Words and Phrases: Blockchain, stochastic process algebra, Verifier's Dilemma

This study was carried out within the PE0000014 - Security and Rights in the CyberSpace (SERICS) and received funding from the European Union Next-GenerationEU - National Recovery and Resilience Plan (NRRP) – MISSION 4 COMPONENT 2, INVESTIMENT 1.3 – CUP N. H73C22000890001. This work has been also partially supported by the Research Project INDAM GNCS 2024 - CUP E53C23001670001 "Modelli composizionali per l'analisi di sistemi reversibili distribuiti (MARVEL)" and by the Project PRIN 2020 - CUP N. 20202FCJMH "NiRvAna - Noninterference and Reversibility Analysis in Private Blockchains". This article reflects only the authors' views and opinions; neither the European Union nor the European Commission can be considered responsible for them.

Authors' Contact Information: Daria Smuseva, Department of Environmental Sciences, Informatics and Statistics, Ca' Foscari University of Venice, Venezia, Venezia, Italy; e-mail: daria.smuseva@unive.it; Andrea Marin, Department of Environmental Sciences, Informatics and Statistics, Ca' Foscari University of Venice, Venezia, Venezia, Italy; e-mail: marin@unive.it; Sabina Rossi, Department of Environmental Sciences, Informatics and Statistics, Ca' Foscari University of Venice, Venezia, Venezia, Italy; e-mail: ssrossi@dais.unive.it; Aad Van Moorsel, School of Computer Science, University of Birmingham, Birmingham, Birmingham, United Kingdom of Great Britain and Northern Ireland; e-mail: a.vanmoorsel@bham.ac.uk.



This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2025 Copyright held by the owner/author(s).

ACM 1558-1195/2025/04-ART12

<https://doi.org/10.1145/3723874>

ACM Reference Format:

Daria Smuseva, Andrea Marin, Sabina Rossi, and Aad Van Moorsel. 2025. Verifier's Dilemma in Proof-of-Work Public Blockchains: A Quantitative Analysis. *ACM Trans. Model. Comput. Simul.* 35, 2, Article 12 (April 2025), 24 pages. <https://doi.org/10.1145/3723874>

1 Introduction

A blockchain is a decentralised distributed network whose records are stored in blocks. When new transactions are submitted to the system, special users called *miners* verify them and include transactions in blockchain blocks. The transactions included in the network are widely acknowledged as being effectively permanent and immutable.

Every blockchain network follows a certain consensus protocol. The original Bitcoin blockchain uses a consensus model referred to as **Proof-of-Work (PoW)** consensus protocol [18]. PoW implies a race among miners to solve a certain computationally intensive problem. Once a new block is announced, other miners can verify its correctness. Only the miner who announces the solution is rewarded for its efforts. One of the main strengths of PoW is the fact that any modification to transactions in the ledger comes with a high computational cost that miners have to face. In particular, since blocks are linked, modifications of the consolidated blocks require malicious participants to re-mine all subsequent blocks. This has such a high computational cost that it is usually considered unfeasible.

The Bitcoin blockchain supports a lightweight scripting language and scripts that are able to specify the conditions required for the validity of transactions. More interestingly, next-generation cryptocurrencies such as Ethereum [6] introduce a Turing-complete script language which allows users to encode any arbitrary computation as scripts. In [17], the computation framework of scriptable cryptocurrencies is referred as a *consensus computer*. Miners have two separate functions in the consensus computer: checking that blocks are correctly constructed and checking the validity of the transactions in each block. It is well known that verifying the correct block construction requires a relatively small amount of work, whereas checking the validity of transactions contained in a block can take much more time since it requires the execution of the corresponding code pieces. Miners are incentivised to verify the scripted transactions for the “common good” of the cryptocurrency. However, verifying scripts consumes computation resources, delaying fair miners in the race to mine the next block. Indeed, while the verification process delays miners in the race for mining the next block, there is no miners’ incentive mechanism behind this process nor a way to check whether miners performed this task. As a consequence, miners face the Verifier’s Dilemma: *use resources for verification to support the blockchain honestly or avoid the verification process to dedicate more resources to mining new blocks in order to maximise the revenues*. The problem is aggravated when the block size or limit increases, as this increases the number of transactions to check.

In September 2022, Ethereum had a hard-fork in which part of the users kept using the PoW protocol (Ethereum Classic, ETC) and the rest adopted a Proof-of-Stake (PoS) consensus. In Ethereum PoS, all participating nodes are required to deposit a (financial) stake that can be taken away if they violate the protocol’s rules. The amount of influence given to each node is proportional to its fraction of total stake. PoS reduces the amount of computational work needed to verify blocks and transactions. PoS systems are claimed to increase the throughput of the chain while also decreasing the total capital costs associated with maintaining the security of the chain. However, when incentives are given to speed up operations, the Verifier’s Dilemma persists also for PoS blockchains. Especially for high-throughput blockchains, the verification of a large number of transactions consumes significant computational resources and time. The delay caused by

computationally intensive contracts could be used in a PoS mechanism as an attack: given that miners have a certain time frame to create and broadcast a block, a long enough delay caused by the validation of the previous block could cause them to miss the window and therefore effectively to lose money.

The ongoing debate between PoS and PoW proponents is highly spirited. While several blockchains, such as Cardano [11] and Algorand [7], have opted for a PoS consensus mechanism, PoW remains the choice for the most prominent blockchain, Bitcoin. Even if we narrow our focus to blockchains that support smart contracts, the list is extensive and includes Ethereum Classic, Kadena, Flux, Horizen, Nervos Network, and Conflux, to name just a few, ranked in decreasing order of market capitalisation.

This article focuses on PoW blockchains with smart contracts, although we believe that some of the insights can be extended to study the Verifier's Dilemma in PoS. Indeed, also in this case, there is no clear mechanism to force verifiers to perform their tasks. Moreover, blocks are generated more frequently and a delay in the voting of a block results in a reduction of the rewards.

In PoW blockchains, miners are rewarded for their work according to a probabilistic process. In blockchains with a PoW consensus mechanism, only the miner that first solves the PoW problem receives the reward. When all miners behave fairly, the probability of winning the race for PoW is equal to the proportion of computational power invested by a miner. This feature leads miners to face the Verifier's Dilemma. Some miners tend to skip the verification process to save computational resources for profitable mining instead of supporting the blockchain fairly. Since in many cases the PoW task consists of computing hashes, miners' computational power is called **hash power (HP)**, i.e., it is measured in number of hashes that can be computed per unit of time.

Clearly, in a healthy blockchain, the large majority of miners should be fair, i.e., they verify other miners' blocks. For this reason, methods to incentivise fair miners or disincentivise unfair ones are being investigated in the literature. Among the most promising approaches to the mitigation of the Verifier's Dilemma problem is the intentional production of invalid blocks [1]. The solution refers to the implementation of a special node that generates invalid blocks, i.e., blocks that contain at least one invalid transaction. Such a block will be rejected by fair miners but accepted by others that will immediately start to compute the PoW of the subsequent block. While this unfair behaviour gives an advantage to the unfair miners upon a valid block announcement, it is harmful when an invalid block is announced. In fact, unfair miners will try to consolidate a new block after an invalid one and, even if they succeed, they will never get any reward because all fair miners will never recognise their blockchain branch. If the miner does not verify an invalid block, they realise that the network of peers has not accepted it only when a new valid block is announced. Hence, the waste of time could be much larger than the advantage of skipping the verification.

Injecting invalid blocks has some drawbacks. First, it diverts some computational power of fair miners to the verification of meaningless blocks. Second, the invalid block must still exhibit a valid PoW that will not be used to secure the immutability of the ledger. For these reasons, the rate of injection of invalid blocks is a crucial problem for the system designers. *This should be the minimum rate that makes miners' unfairness non- or disadvantageous.*

In this article, we use the **Performance Evaluation Process Algebra (PEPA)** [13] for modelling the salient aspects of the Verifier's Dilemma in PoW blockchains and the above described mitigation approach. The choice of PEPA is motivated by its compositional nature, allowing us to model complex systems as the composition of various simple, cooperating components. Moreover, it is quite interesting to observe that the solution of the PoW is known to be a memoryless process, meaning that the delays between two subsequent blocks are independent and exponentially distributed, as is the time required by a single miner to solve the PoW (see, e.g., [5]). This justifies the use of a Markovian process algebra such as PEPA.

The main contributions of the article are:

- First, we propose an algebraic, compositional model for the Verifier’s Dilemma in consensus computers. The number of states in the model is proportional to the number of fair miners in the network. We show how the model can be reduced to a more tractable 4-state model by resorting to a bisimulation-based aggregation technique, named *strong equivalence*. The latter has been defined by Hillston in [13] for models expressed as PEPA terms.
- We propose a generalisation of the definition of strong equivalence that still preserves the property of inducing a *strong lumpability* on the underlying Markov chain [3, 14, 16]. A CTMC is strongly lumpable with respect to a partitioning of states if the stochastic process associated with the new partition remains a **Continuous Time Markov Chain (CTMC)**. Simple conditions on the transition rates allow one to decide whether a CTMC is strongly lumpable with respect to a certain partition. This aggregation technique allows us to efficiently compute the exact values of the performance indices of the original model. Therefore, there is a crucial difference between this work and the contribution of [20]: the state space reduction done in the latter is approximate and is *not* a lumping of the CTMC underlying the PEPA model. Intuitively, the approximation was expected to be accurate, but it is very difficult, in general, to give bounds to the performance indices. In this work, we overcome this problem by introducing a new model, more sophisticated in its definition, which is strongly lumpable and, hence, provides exact results.
- The third contribution is the introduction of a PEPA model that takes into account the invalid block injection mechanism. Using this model, we are able to investigate this important countermeasure from the perspective of the single miner and that of the entire network. The model allows us to determine the waste in terms of rewards both of fair miners (that spend more time in verifying) and of unfair miners (that spend time in mining possibly invalid blocks because of being subsequent to an injected one). A counterintuitive observation following this model analysis is that the optimal rate of injection of invalid blocks grows with the percentage of fair miners. We will explain the reason from a system perspective in the following sections.
- The analysis of the model allows us to give a closed-form expression for the optimal injection rate of invalid blocks, which is provably correct under mild conditions. The advantage of this approach is that it can be used to dynamically control the optimal injection rate without requiring the solution of the model for each change in its parameters.

This article is an extended and revised version of our previous work [20]. With respect to the conference contribution, in this article, we propose a more elegant model and analysis that allow us to resort to the well-known property of Markov chains called *strong lumpability*. This leads to a simplification of the expressions of the stationary performance indices. As a consequence, we are able to give a closed-form expression for the optimal invalid block injection rate, thus improving the numerical approach used in [20]. Furthermore, this allows us to prove an important monotonicity relation between the verification rate and the invalid block generation rate that was previously not observed. While from the numerical point of view, the results of the new model presented here are compatible with those of [20], the fact that we aggregate the states in an exact manner rather than using an approximation allows us to give a simpler and more efficient way of studying the system as well as to unveil some new insights with practical relevance.

The article is structured as follows. In Section 2, we provide a literature review. Section 3 gives the necessary background. Section 4 proposes a model for the Verifier’s Dilemma. Section 5 proposes a model for the invalid block injection approach. In Section 6, we study the optimal configuration of the system using the invalid block injection. Finally, Section 7 concludes the article.

2 Related Work

The Verifier's Dilemma remains the subject of discussion in many Ethereum-related works. The authors of [17] first spotted this problem. They have shown, from a game theoretical perspective, that rational miners would omit the verification process to maximise their rewards. Based on this idea, other authors [12] analysed the case in which malicious miners design dummy smart contracts which are computationally expensive to verify in order to slow down the mining process of all other miners and, hence, increase the possibilities of winning the PoW race. In addition, the authors of [1, 12, 17] showed the profitability of skipping the verification process in scenarios with computationally intensive smart contracts.

Various authors [4, 8, 10, 23] introduced the parallel computation of the smart contract transactions in order to mitigate the drawback of long verification time. Particularly, in [8], the on-chain protocol that allows miners to delay verification of transactions in a block by up to the certain number of blocks ahead was introduced. The authors showed that, thanks to this implementation, the Ethereum-like blockchains become less vulnerable to the problems caused by the growth of the verification time.

Alternatively, in [1, 9, 15, 21], a number of off-chain solutions have been proposed for efficient computation of complex smart contracts. They suggested that the verification of complex contracts is no longer performed by all of the nodes; rather, verification is performed by only a small subset of them. Those nodes earn rewards only if they perform the verification correctly. Furthermore, in [17], the authors proposed to divide complex transactions into a set of transactions with smaller size so that rational validators have less incentive to skip verification. After running and verifying each step as individual scripts, the final outcome matches the result of correctly executing the original script.

A closely related work is [1]. The authors introduced a simulation framework that allows them to investigate the network behaviour when a fraction of the miners do not verify the blocks. They showed that the single non-verifying miner can be rewarded approximately 23% higher than the network's fair share as total gain of all other fair nodes is reduced respectively. They also proposed the injection of invalid blocks as a solution to the Verifier's Dilemma problem. The authors propose to use a special node for intentionally generating invalid blocks as a way to punish non-verifying miners. With respect to our work, we have used the simulation estimates of [1] to validate our PEPA model. Then, we used this model to determine the optimal rate of injection of invalid blocks. In fact, it would be very computationally expensive to run stochastic simulations searching the optimal value of this rate whereas, thanks to our model, the problem is reduced to the computation of a simple expression.

3 Background

3.1 Verifier's Dilemma

In PoW blockchains, consensus is fully distributed and miners contribute to the life of the blockchain in two ways: (i) they verify the transactions stored in the blocks and (ii) they compete to solve the PoW problem.

Smart contracts are important features of blockchains. In Ethereum Classic, they are stored as contract accounts, which contain the code and some storage space to support their execution [6]. Users of the blockchain control the so-called externally owned accounts. External entities and contract accounts cooperate with each other through two types of transactions: financial and contract based. The former is responsible for moving the Ethereum Classic cryptocurrency, ETC, among the accounts. Contract transactions control all interactions with contract accounts: either to publish a new smart contract or to execute an existing one.

While the verification of financial transactions is very quick since its heaviest activity is to verify that no double spending has been done, smart contracts pose more problems. Smart contracts are written with a Turing complete language and the verification of a transaction containing the output of its execution requires all miners to re-execute the same program on the same input [17].

Miners are rewarded for their work according to a probabilistic process. In fact, only the miner that solves the PoW problem first receives the reward. When all miners behave fairly, the probability of winning the race for PoW is equal to the proportion of computational power invested by a miner. This feature leads miners to face the Verifier's Dilemma. Some miners tend to skip the verification process to save computational resources for profitable mining instead of supporting the blockchain fairly.

One of the solutions is to inject invalid blocks in the network to punish unfair miners. The idea is that if an unfair miner does not verify the announced blocks, it would mine its candidate block after an invalid one and, hence, does not gain a fee in the case of success. Thus, the unfair miners must face a trade-off between the advantage of skipping the verification phase and the risk of mining a new block after an invalid one.

At the state-of-the-art, there has not been a rigorous analysis of this mitigation solution using quantitative analysis techniques. Our goal is to investigate the impact of the Verifier's Dilemma and to detect the minimum rate of invalid block injection that makes the unfair policy less lucrative than the fair one. Our work applies to any blockchain with PoW and smart contracts, such as ETH classic, Kadena, and similar systems that were mentioned in the Introduction.

3.2 PEPA

We use PEPA [13] to investigate the Verifier's Dilemma in Ethereum and its impact on miners' behaviour. In this section, we give a brief overview of PEPA. The advantages of using PEPA lie primarily in the fact that PEPA is compositional, every PEPA specification has an underlying stochastic process and, under given assumptions, this stochastic process is a continuous time Markov process. Moreover PEPA is supported by a tool that can be applied to Eclipse, namely, the PEPA Eclipse Plug-in [22].

The primitives of the language PEPA are components and activities. Each activity is represented by a pair (α, r) where α is a label, or action type, and r is its activity rate, that is, the parameter of an exponential distribution determining its duration. The language terms describe the behaviour of components via the activities they undertake as well as the interactions between them which are expressed through a small set of combinators.

The syntax for PEPA terms is defined as follows:

$$P ::= (\alpha, r).P \mid P + Q \mid P \bowtie_L Q \mid P/L \mid A$$

Prefix: $(\alpha, r).P$

The component $(\alpha, r).P$ carries out the activity (α, r) with action type α and rate r and subsequently behaves as P . The activity completion takes some time Δt , drawn from the distribution.

Choice: $P + Q$

The component $P + Q$ represents a system which may behave either as P or as Q . It enables all the activities of P and all the activities of Q . The first enabled activity determines one of the components whereas the other component is discarded. The probability that P and Q both complete an activity at the same time is zero due to the continuous nature of the probability distributions.

Cooperation: $P \bowtie_L Q$

The cooperation combinator represents an interaction between two components which is determined by the set L of action types called a *cooperation set*. The cooperation set defines the action types on which the components are forced to synchronise. Activities with action types in the set L require the simultaneous involvement of both components whereas activities whose types do not occur in the cooperation set L are assumed to proceed independently. The former are called *shared activities* whereas the latter are called *individual activities*. Shared activities will only be enabled in $P \bowtie_L Q$ when they are enabled in both P and Q . Thus, one component may become blocked, waiting for the other component to be ready to synchronise. The duration of a shared activity is reflected by the rate of the slower participant. A component in a cooperation is called *passive* if it has an activity with an unspecified rate τ .

When the cooperation set L is empty, the combinator $\bowtie_L$ represents the parallel composition which allows components to proceed concurrently without any interaction between them.

Hiding: P/L

The component P/L behaves as P except that any activities of types within the set L are hidden, i.e., their type appears as an unknown type, denoted by the special symbol τ , and can be regarded as an internal delay by the component. Such an activity cannot be carried out in cooperation with any other component.

Constant: $A \stackrel{\text{def}}{=} P$

We assume that there is a countable set of *constants*. Constants are components whose meaning is given by a defining equation, e.g., $A \stackrel{\text{def}}{=} P$, which associates the constant A to the behaviour of the component P . This is how we assign names to components.

4 Analysis of the Verifier's Dilemma

In this section, we present a PEPA model for the Ethereum network with fair and unfair miners. The validation of the model is done with the estimates obtained by stochastic simulation in [1]. More precisely, the analysis of the PEPA model of the base system with fair and unfair miners produces results for the rewards that are consistent with the closed-form expression of [1]. In the same work, the authors show the comparison of the rewards obtained with the closed-form expression and the data-driven simulation estimates obtained with BlockSim [2]. The accuracy of this expression and, hence, of the proposed PEPA model is therefore assessed. The novelty of this work consists in proposing a PEPA model for the system with fake block injection. Although we cannot validate this model with trace-driven simulations since this policy has not been implemented yet in any blockchain, its underlying assumptions are the same as those of the base model.

We assume that all miners have the same computational power γ , and verification is performed with rate β that corresponds to the parameter of an exponentially distributed random time. Another assumption is about the constant number of miners. While, in general, this is untrue for public blockchains, here we have to interpret this assumption as the request that the speed at which the system reaches its stationary condition is higher than the speed at which the number of miners changes. For example, according to [blockchain.com](https://www.blockchain.com/explorer/charts/hash-rate),¹ between August 10 and September 10th, 2023, the HP of the network changed at most 4% in 1 day. Since the network generates a block every 10 minutes, we can safely assume that the number of miners remains constant for a sufficiently long time to justify a stationary analysis. We emphasise that although we present a model for an ideal Proof-of-Work blockchain supporting smart contracts, it should be adjusted to

¹<https://www.blockchain.com/explorer/charts/hash-rate>

Table 1. PEPA Model of a Network with $K \geq 3$ Fair Miners

M_i	$\stackrel{def}{=} (m_i, \gamma).M_i + \sum_{j \neq i} (m_j, \tau).V_i$
V_i	$\stackrel{def}{=} \sum_{j \neq i} (v_j, \beta).M_i + \sum_{j \neq i} (m_j, \tau).V_i$
$Network$	$\stackrel{def}{=} (..((M_1 \boxtimes_{L \cup L_{12}} M_2) \boxtimes_{L \cup L_3} \dots) \boxtimes_{L \cup L_K} M_K$
where	$i, j \in \{1, \dots, K\}$ and $L = \{m_1, \dots, m_K\}, L_{12} = \{v_3, \dots, v_K\}, L_j = \{v_1, \dots, v_K\} \setminus \{v_j\}$ for $j \geq 3$

fit the specific real-world scenario under investigation. Nonetheless, the core conclusions of the model remain valid: the block injection rate must be calculated under the assumption that all miners are fair, as the most profitable strategy arises when a single miner switches from fair to unfair behaviour.

4.1 Fair Miners

We first inspect the behaviour of miners who follow the blockchain consensus protocol by verifying all new blocks in the network. Let $M_1, \dots, M_K$ be a network of K fair miners. The specification of a single fair (verifying) miner, say, M_1 , can be expressed in PEPA as

$$\begin{aligned} M_1 &\stackrel{def}{=} (m_1, \gamma).M_1 + (m_2, \tau).V_1 + \dots + (m_K, \tau).V_1 \\ V_1 &\stackrel{def}{=} (v_2, \beta).M_1 + \dots + (v_K, \beta).M_1 + (m_2, \tau).V_1 + \dots + (m_K, \tau).V_1 \end{aligned}$$

Miner M_1 mines a new block with action type m_1 and rate γ . Then, it returns to its initial state and it starts to mine another block. Whenever M_1 gets a block from the network, through an activity (m_j, τ) with $j \in \{2, \dots, K\}$, it starts the verification process and moves to state V_1 . We assume that all miners have an equal proportion of computational power; thus, they perform verification synchronously. In other words, while the verification of the block requires a random amount of time, the delay is the same for all verifiers. This is captured in the model by the synchronisation operation between miners. The verification of a block mined by the miner M_i occurs with rate β and type of action v_i . For instance, if M_3 created a block, then the rest of the network synchronises on verification with action type v_3 . It is clear that miners verify all blocks from the network with exception of the block that was produced by themselves. During the verification phase, M_1 can still get new blocks from the rest of the network. Notice that this corresponds to the assumption that the amount of gas² associated with a block is exponentially distributed and independent. Therefore, while all blocks have a verification time that is independent and exponentially distributed, all miners spend exactly the same amount of time for the same block verification as a consequence of the assumption on their computational power.

In Table 1, we report the specification of a network composed by $K \geq 3$ fair miners. Notice that, according to the operational semantics of PEPA, if miner M_i announces a block, m_i is received by all miners except M_i . Thus, all miners move to state V_j with $j \neq i$, while M_i is still mining. Now, according to the synchronisation operation specified in Table 1, all miners in V_j synchronise on the intersection of all synchronising sets that is exactly equal to $\{v_i\}$. Hence, once the verification is completed, they all simultaneously move to the mining phase due to action v_i , as required.

The derivation graph of the model in Table 1 is depicted in Figure 1.

²Gas is the computational effort required by a smart contract. This is proportional to the fee paid by the user for the inclusion of the result in a block.

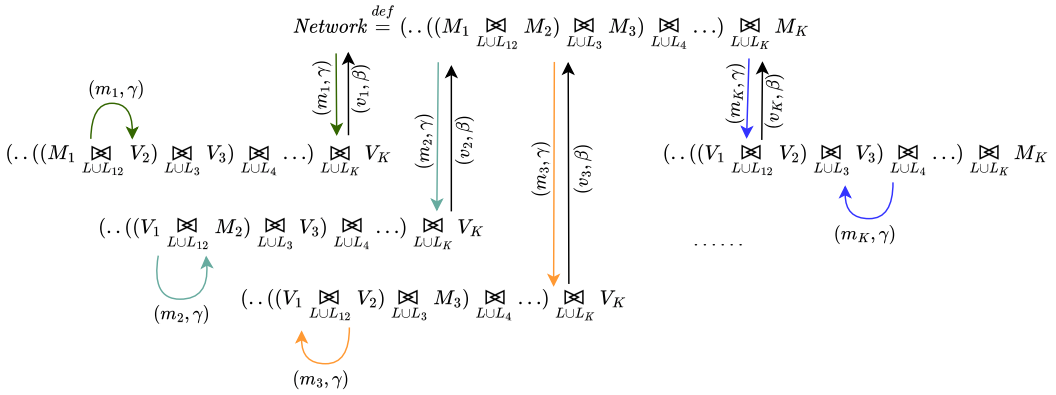


Fig. 1. Derivation graph of the model in Table 1.

Table 2. PEPA Model of a Network with K Fair Miners and N Unfair Miners

M_{U_n}	$\stackrel{\text{def}}{=} (m_{U_n}, \gamma).M_{U_n} + \sum_{l \neq n} (m_{U_l}, \tau).M_{U_n} + \sum_i (m_{F_i}, \tau).M_{U_n}$
M_{F_i}	$\stackrel{\text{def}}{=} (m_{F_i}, \gamma).M_{F_i} + \sum_{j \neq i} (m_{F_j}, \tau).V_i + \sum_n (m_{U_n}, \tau).V_{iU}$
V_i	$\stackrel{\text{def}}{=} \sum_{j \neq i} (v_j, \beta).M_{F_i} + \sum_{j \neq i} (m_{F_j}, \tau).V_i + \sum_n (m_{U_n}, \tau).V_{iU}$
V_{iU}	$\stackrel{\text{def}}{=} (v_U, \beta).M_{F_i} + \sum_{j \neq i} (m_{F_j}, \tau).V_i + \sum_n (m_{U_n}, \tau).V_{iU}$
Network	$\stackrel{\text{def}}{=} (M_{U_1} \boxtimes_L \dots \boxtimes_L M_{U_N}) \boxtimes_L (((M_{F_1} \boxtimes_{L \cup L_{12}} M_{F_2}) \boxtimes_{L \cup L_3} M_{F_3}) \boxtimes_{L \cup L_4} \dots \boxtimes_{L \cup L_K} M_{F_K})$
where	$i, j \in \{1, \dots, K\}, \quad n, l \in \{1, \dots, N\} \text{ and } L = \{m_{U_1}, \dots, m_{U_N}, m_{F_1}, \dots, m_{F_K}\}, L_{12} = \{v_U, v_3, \dots, v_K\},$ $L_3 = \{v_U, v_1, v_2, v_4, \dots, v_K\}, L_4 = \{v_U, v_1, v_2, v_3, v_5, \dots, v_K\}, \dots, L_K = \{v_U, v_1, \dots, v_{K-1}\}$

4.2 Unfair Miners

Suppose that the network contains some unfair miners that do not verify incoming blocks. The PEPA specification of a network with K fair miners and N unfair miners is described in Table 2. Any non-verifying, or unfair, miner M_{U_n} with $n \in \{1, \dots, N\}$ skips the verification step and adds the incoming block directly to its copy of the ledger. Once fair miners M_{F_i} with $i \in \{1, \dots, K\}$ receive a block from an unfair component, they perform the verification v_U and synchronise on it. Fair and unfair miners cooperate (using the PEPA operator $\bowtie$); we call the resulting component a *Network*. Figure 2 represents the derivation graph of the model in Table 2, whereas Figure 3 represents its underlying Markov chain.

4.3 Lumped Model

The model with the detailed representation of the state of each miner is not scalable since the total number of miners in a blockchain is usually very large. Thus, we resort to a representation based on aggregation through lumping of all fair and unfair miners into environments. In order to be able to analyse the behaviour of one fair miner and of one unfair miner, we leave one of them outside the aggregation.

Let K be the number of fair miners and N be the number of unfair miners. The PEPA specification of the lumped model is represented in Table 3, its derivation graph is depicted in Figure 4, and the underlying CTMC is presented in Figure 5.

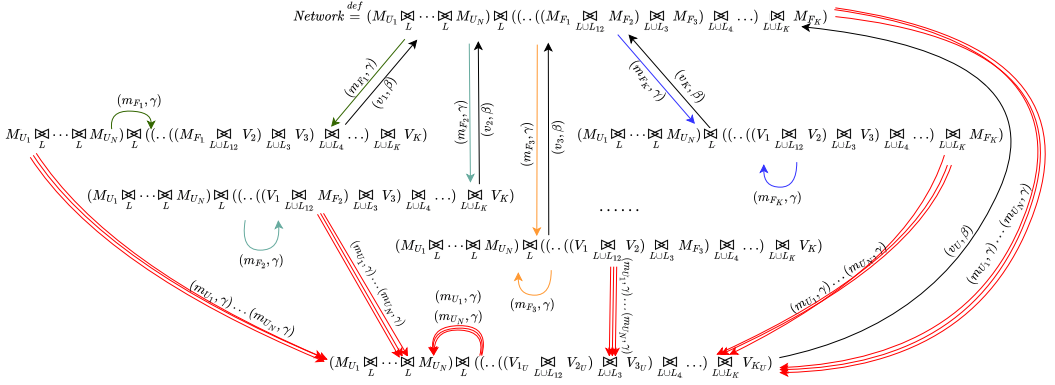


Fig. 2. Derivation graph of the model in Table 2.

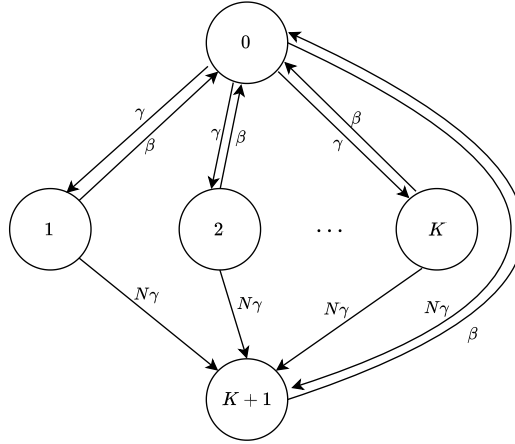


Fig. 3. Markov chain of the model in Table 2.

The fair (i.e., correctly operating) environment is defined as E_F and represents the behaviour of $K - 1$ verifying miners. E_U corresponds to the unfair environment where $N - 1$ miners avoid the verification part. Notice that $(K - 1)\gamma$ and $(N - 1)\gamma$ are the fractions of the hash power of verifying miners and non-verifying miners, respectively. The *Lumped_Network* component is obtained as the cooperation of all components over the set of actions L .

Let us inspect the behaviour of single fair and unfair miners as well as the behaviour of those of the environments (see Table 4). Component M_F represents a single fair (verifying) miner. The miner mines a new block with action type m_F and rate γ . Then, it returns to its initial state and it starts to mine another block. Once verifying miner M_F gets a block from the network, it starts the verification process V_F . The non-verifying, or unfair, miner M_U skips this step and adds the incoming block directly to its copy of the ledger. The verification of a block occurs with rate β .

It is worth noting that, while the race policy of PoW requires us to sum the rates of all miners belonging to an environment, the verification phase is not governed by a race policy. Hence, it is performed with rate β as it happens for the single miner. Intuitively, the amount of work required for the verification is assumed to be exponentially distributed but, since all miners have the same computational power and must perform the same operations, we can say that the aggregated speed remains β . From the system perspective, this is connected to the fact that the verification time

Table 3. Lumped PEPA Model of a Network with K Fair Miners and N Unfair Miners with Distinguished M_F and M_U

M_U	$\stackrel{def}{=}$	$(m_U, \gamma).M_U + (m_F, \tau).M_U + (m_{E_F}, \tau).M_U + (m_{E_U}, \tau).M_U$
E_U	$\stackrel{def}{=}$	$(m_{E_U}, (N-1)\gamma).E_U + (m_F, \tau).E_U + (m_{E_F}, \tau).E_U + (m_U, \tau).E_U$
M_F	$\stackrel{def}{=}$	$(m_F, \gamma).M_F + (m_{E_F}, \tau).V_F + (m_U, \tau).V_U + (m_{E_U}, \tau).V_U$
V_F	$\stackrel{def}{=}$	$(v_{E_F}, \beta).M_F + (m_{E_F}, \tau).V_F + (m_U, \tau).V_U + (m_{E_U}, \tau).V_U$
V_U	$\stackrel{def}{=}$	$(v_U, \beta).M_F + (m_U, \tau).V_U + (m_{E_U}, \tau).V_U$
E_F	$\stackrel{def}{=}$	$(m_{E_F}, (K-1)\gamma).V_{E_2} + (m_F, \tau).V_{E_1} + (m_U, \tau).V_{E_U} + (m_{E_U}, \tau).V_{E_U}$
V_{E_1}	$\stackrel{def}{=}$	$(v_F, \beta).E_F + (m_F, \tau).V_{E_1} + (m_U, \tau).V_{E_U} + (m_{E_U}, \tau).V_{E_U}$
V_{E_2}	$\stackrel{def}{=}$	$(v_{E_F}, \beta).E_F + (m_{E_F}, \gamma).V_{E_2} + (m_U, \tau).V_{E_U} + (m_{E_U}, \tau).V_{E_U}$
V_{E_U}	$\stackrel{def}{=}$	$(v_U, \beta).E_F + (m_U, \tau).V_{E_U} + (m_{E_U}, \tau).V_{E_U}$
$Lumped_Network$	$\stackrel{def}{=}$	$(M_U \bowtie_L E_U) \bowtie_L (M_F \bowtie_{L \cup V} E_F)$
where		$L = \{m_U, m_{E_U}, m_F, m_{E_F}\}$ and $V = \{v_{E_F}, v_U\}$

depends on the amount of gas in the block and on the computational power of the miners. Since the block under verification and the computational power are the same for all miners in the verification phase, this delay must be the same for everyone. Another interesting aspect is that since the time for solving PoW is independent and exponentially distributed by design, the aggregation of the rates is exact.

The underlying derivation graph of the model is depicted in Figure 4. Each fair miner that is in the verification phase can still get new blocks from the rest of the network. It means that, in real systems, miners must maintain a queue of incoming blocks. However, the required time for verification is usually much lower than the block interval time. This observation allows us to simplify V_F and V_{E_F} by abstracting out the queue of blocks to verify. However, according to [1], the verification time for an 8MB block is around 0.23 seconds, which is substantially smaller than the time between two blocks when PoW is used (e.g., 13s in Ethereum classic). Experiments confirmed that this simplification does not significantly affect the type of analyses that we perform. Indeed, the model can be easily extended to maintain a finite queue of validated blocks but it loses its analytical tractability. The model with a short queue is numerically tractable and for realistic parametrisations is consistent, from a quantitative point of view, with the model which validates only one block.

The PEPA models *Network* and *Lumped_Network* reported in Tables 2 and 3, respectively, are *strongly equivalent* with respect to the following definition, which is a generalisation of the notion of strong equivalence presented in [13]. Let $L_1, \dots, L_h$ be a partition of the set of action types. We denote by $q[P, S, L]$ the total conditional transition rate from the PEPA process P to the set of PEPA processes in S through activities with action type in L , i.e., $q[P, S, L] = \sum_{r \in H} r$, where $H = \{r \mid P \xrightarrow{(\alpha, r)} P' \text{ and } P' \in S \text{ and } \alpha \in L\}$.

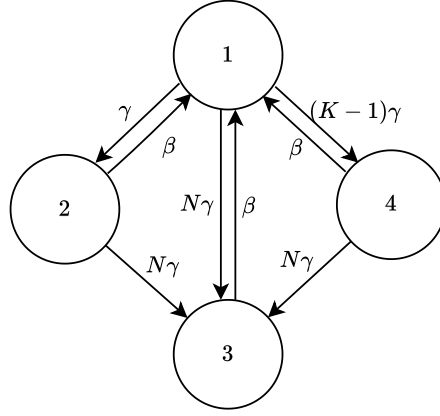


Fig. 5. Markov chain of the model in Table 3.

The relation $\sim$ induces a partition on the derivative set of P into equivalence classes with the condition of *strong lumpability* [16] for the underlying Markov chain.

Definition 4.2 (CTMC Strong Lumpability). Given a CTMC on the state space $\mathcal{S}$, we say that this is strongly lumpable with respect to the partition $\mathcal{S}/\sim$ if and only if for any two states in the same equivalence class, the total transition rates to any other class are the same.

An important property of lumped processes is that the steady-state probability of each aggregated macro-state is equal to the sum of the steady-state probabilities of the corresponding equivalent states. Referring to our models, it is easy to prove that *Network* and *Lumped_Network* are strongly equivalent with respect to the partition $\{m_{F_1}, \dots, m_{F_K}, m_F, m_{E_F}\}, \{m_{U_1}, \dots, m_{U_N}, m_U, m_{E_U}\}, \{v_1, \dots, v_K, v_F, v_{E_F}\}, \{v_U\}$.

Moreover, the symbolic expression of the steady-state probabilities for the Markov chain in Figure 3 underlying the process *Network* is defined as follows for $i \in \{1, \dots, K\}$:

$$\pi_0 = \frac{\beta}{\beta + (K + N)\gamma} \quad \pi_i = \frac{\beta\gamma}{(\beta + N\gamma)(\beta + (K + N)\gamma)} \quad \pi_{K+1} = \frac{N\gamma}{\beta + N\gamma}.$$

For instance, if we use the parameters of Table 5, we obtain $\pi_0 \approx 0.790932$, $\pi_i \approx 0.00203684$, $\pi_{K+1} \approx 0.0257524$.

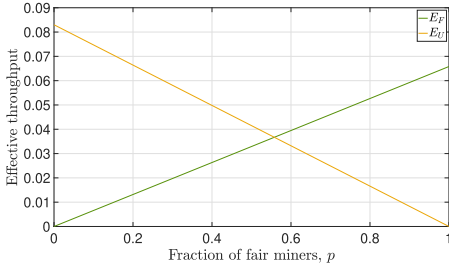
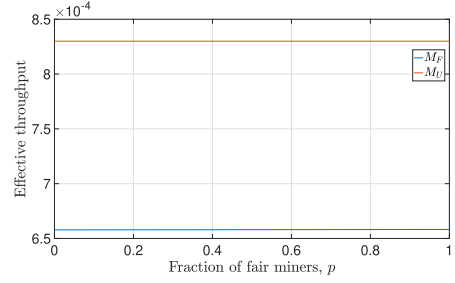
On the other hand, the steady-state probabilities of the Markov chain underlying the *Lumped_Network* depicted in Table 5 are

$$\begin{aligned} \tilde{\pi}_0 &= \frac{\beta}{\beta + (K + N)\gamma} & \tilde{\pi}_1 &= \frac{\beta\gamma}{(\beta + N\gamma)(\beta + (K + N)\gamma)} & \tilde{\pi}_2 &= \frac{(K - 1)\beta\gamma}{(\beta + N\gamma)(\beta + (K + N)\gamma)} \\ \tilde{\pi}_3 &= \frac{N\gamma}{\beta + N\gamma}. \end{aligned}$$

If we use the parameters of Table 5 to compare the results of the model in Table 2, we obtain the following values: $\tilde{\pi}_0 \approx 0.790932$, $\tilde{\pi}_1 \approx 0.00203684$, $\tilde{\pi}_2 \approx 0.181279$, $\tilde{\pi}_3 \approx 0.0257524$. Note that $\pi_2 \approx 0.181279$ is the sum of the probabilities of the 89 states in which one of the fair miners mined a block and all the others are verifying it, i.e., $0.00203684 * 89 \approx 0.181279 = \tilde{\pi}_2$.

Table 5. Rate Parameter Assignment

Parameter	Value
K	90 fair miners
N	10 unfair miners
γ	8.3×10^{-4} blocks/s
β	0.314 s^{-1}

(a) Effective throughput of the fair environment E_F and the unfair environment E_U (b) Effective throughput of a single fair miner M_F and a single unfair miner M_U Fig. 6. Effective throughput as function of the number of verifying miners, where γ , β , and $M = N + K$ are taken from Table 5.

4.4 Model Assessment

We can now use the lumped model to evaluate the effective throughput of fair and unfair miners, which is the throughput of action types m_F , m_U , m_{E_F} and m_{E_U} and, hence, the profits of fair and unfair miners from the aggregated and individual perspective. Note that our model abstracts out the network delays. However, it is worthy of notice that the goal of the model is not that of providing an accurate estimate of the absolute reward for miners, rather it is that of comparing the rewards obtained by unfair miners compared with fair ones. Therefore, it is safe to assume that the impact of network delays are the same for fair and unfair miners.

Figure 6(a) shows the total throughput of all fair and all unfair miners as a function of the fraction of fair miners (green and orange lines, respectively). Indeed, the higher number of fair miners implies a reduction of the total throughput of unfair miners, which consequently leads to a rise of the throughput for those who follow the protocol. Observe that the equilibrium point, where the two lines intersect each other, corresponds to 55% of verifying miners in the network. Therefore, in the case of an equal proportion of the two classes of miners, the non-verifying miners receive a greater reward. Figure 6(b) compares the effective throughput of one verifying (blue line) and one non-verifying miner (red line) as a function of the fraction of verifying miners in the network. This plot illustrates that the unfair miner receives $\approx 25\%$ more reward than the fair miner. Note that the fraction of fair miners is $p = \frac{K}{M}$, where $M = K + N$. In this example, M is constant and is equal to 100. The other model configuration settings are used from Table 5. The performance of the single non-verifying miner remains equal to γ , as it skips the verification step using available resources only for mining. As shown in Figure 7(a), the effective throughput of the single fair miner is only slightly reduced in relation to the network with all fair miners.

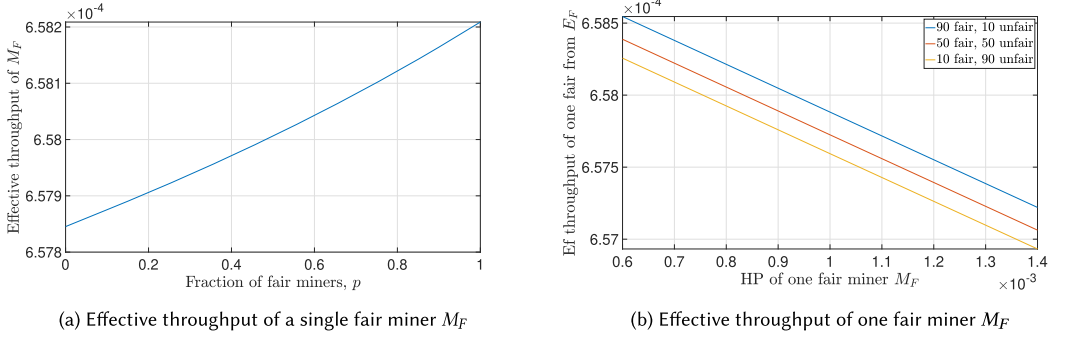


Fig. 7. Effective throughput as function of the number of fair miners (a) and HP of one unfair miner (b), where γ , β , and $M = N + K$ are taken from Table 5.

Suppose that one fair miner has different computational power from the rest of the network. The effective throughput for unfair components will not change; however, it will affect the effective throughput of the other fair components. Figure 7(b) shows the impact of different γ on the total throughput of one fair miner in the fair environment. Clearly, increasing of computational power for one fair miner results in decreasing of effective throughput of other fair components since they need to verify more blocks.

5 Analysis of the Invalid Block Injection Policy

In the literature, several solutions have been proposed to mitigate the Verifier's Dilemma. In this section, we discuss one mitigation solution, the intentional injection of invalid blocks, which was proposed to mitigate the problem by Alharby et al. [1].

In general, miners benefit from skipping the verification step because almost all blocks are valid anyway. Intentional production of invalid blocks aims to punish unfair miners since they could end up working on new blocks on top of the invalid ones, thus not gaining a fee for the new blocks. Invalid blocks are supposed to be generated by a special node, named *creator*.

When an invalid block is injected, it introduces a 'soft fork' of the chain, with the unfair miners working on an invalid branch and fair ones staying on a correct branch. We assume that, thanks to a blacklist of invalid block hashes, fair miners always recognise the correct branch. On the other hand, unfair miners realise that they are working on the wrong branch as soon as a fair miner announces a new block. Basically, since an unfair miner knows that they could be working on a wrong branch of the chain, it is rational for them to work on the last block announced (recall that PoW is a memoryless operation and, hence, this does not cause a waste of work).

5.1 Model Description

To study and optimise the invalid block injection approach, we extend the *Lumped_Network* with more components. We assume that there are no malicious miners in the network; thus, all blocks are valid with the exception of those intentionally injected. If this is not the case, the rate of invalid block creation from malicious miners should be added to that of the creator. The PEPA model of the system with invalid block injection is called the *Invalid Block Injection Model (IBIM)* and is reported in Table 6.

In presenting the model, we perform an abstraction by introducing a new component C that acts as a fake block creator. Note that, in real systems, this needs not to correspond to an actual single miner but rather depends on the design of the protocol (e.g., it can correspond to a subset

Table 6. PEPA Specification of the Lumped Model with Invalid Block Injection Functionality

M_U	$\stackrel{\text{def}}{=} (m_U, \gamma).M_U + (m_F, \top).M_U + (m_{E_F}, \top).M_U + (m_{E_U}, \top).M_U + (m_{fake}, \top).M_{U_{INV}}$
$M_{U_{INV}}$	$\stackrel{\text{def}}{=} (\tau, \gamma).M_{U_{INV}} + (m_F, \top).M_U + (m_{E_F}, \top).M_U + (m_{fake}, \top).M_{U_{INV}}$
E_U	$\stackrel{\text{def}}{=} (m_{E_U}, (N-1)\gamma).E_U + (m_F, \top).E_U + (m_{E_F}, \top).E_U + (m_U, \top).E_U + (m_{fake}, \top).E_{U_{INV}}$
$E_{U_{INV}}$	$\stackrel{\text{def}}{=} (\tau, (N-1)\gamma).E_{U_{INV}} + (m_F, \top).E_U + (m_{E_F}, \top).E_U + (m_{fake}, \top).E_{U_{INV}}$
M_F	$\stackrel{\text{def}}{=} (m_F, \gamma).M_F + (m_{E_F}, \top).V_F + (m_U, \top).V_U + (m_{E_U}, \top).V_U + (m_{fake}, \top).V_U$
V_F	$\stackrel{\text{def}}{=} (v_{E_F}, \beta).M_F + (m_{E_F}, \top).V_F + (m_U, \top).V_U + (m_{E_U}, \top).V_U + (m_{fake}, \top).V_U$
V_U	$\stackrel{\text{def}}{=} (v_U, \beta).M_F + (m_U, \top).V_U + (m_{E_U}, \top).V_U + (m_{fake}, \top).V_U$
E_F	$\stackrel{\text{def}}{=} (m_{E_F}, (K-1)\gamma).V_{E_2} + (m_F, \top).V_{E_1} + (m_U, \top).V_{E_U} + (m_{E_U}, \top).V_{E_U} + (m_{fake}, \top).V_{E_U}$
V_{E_1}	$\stackrel{\text{def}}{=} (v_F, \beta).E_F + (m_F, \top).V_{E_1} + (m_U, \top).V_{E_U} + (m_{E_U}, \top).V_{E_U} + (m_{fake}, \top).V_{E_U}$
V_{E_2}	$\stackrel{\text{def}}{=} (v_{E_F}, \beta).E_F + (m_{E_F}, \gamma).V_{E_2} + (m_U, \top).V_{E_U} + (m_{E_U}, \top).V_{E_U} + (m_{fake}, \top).V_{E_U}$
V_{E_U}	$\stackrel{\text{def}}{=} (v_U, \beta).E_F + (m_U, \top).V_{E_U} + (m_{E_U}, \top).V_{E_U} + (m_{fake}, \top).V_{E_U}$
C	$\stackrel{\text{def}}{=} (m_{fake}, \epsilon).C + (m_U, \top).C + (m_F, \top).C + (m_{E_U}, \top).C + (m_{E_F}, \top).C$
$Lumped_Network$	$\stackrel{\text{def}}{=} ((M_U \bowtie_L E_U) \bowtie_L (M_F \bowtie_{L \cup V} E_F)) \bowtie_L C$
where	$L = \{m_U, m_{E_U}, m_F, m_{E_F}, m_{fake}\}$ and $V = \{v_{E_F}, v_U\}$

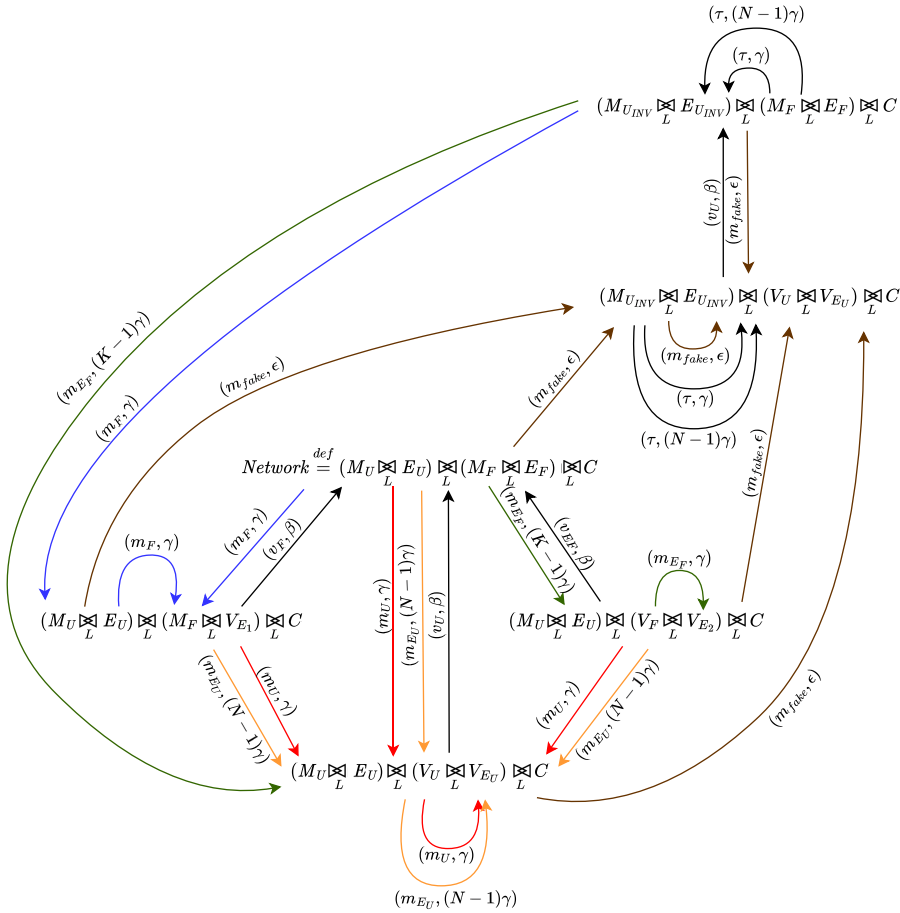
Table 7. Description of the Model in Table 6

Symbol	Description
C	Invalid block creator that creates invalid blocks with action type m_{fake} and rate ϵ .
$M_{U_{INV}}$	Single unfair miner with a fake block in its blockchain copy. It mines blocks with action type τ and rate γ . When it receives a block from M_F or E_F , it obtains the valid blockchain copy and returns to the initial state M_U .
$E_{U_{INV}}$	Unfair environment with an invalid blockchain copy. It almost always remains in the same state and returns to E_U only when the ledger is synchronised with M_F and E_F .

of miners selected randomly). The new component C models the behaviour of the invalid block creator. It produces invalid blocks with the action type m_{fake} and rate ϵ . The creator recognises and rejects all invalid blocks and always works on the valid blockchain copy.

To model the behaviour of non-verifying miners M_U and E_U , we introduce the components $M_{U_{INV}}$ and $E_{U_{INV}}$ representing the same components working on an invalid blockchain copy. Once miner M_U , which is working on the valid chain, receives a fake block from the creator, its state changes to $M_{U_{INV}}$. The same explanation can be applied to the states of the unfair environment E_U . A description of the new components is presented in Table 7; the underlying derivation graph of the model is depicted in Figure 8.

The performance metric of interest is the effective throughput of miners, i.e., the expected amount of blocks that a miner can consolidate per unit of time. This index is used to estimate



the profit obtained by miners. In fact, the expected reward for a block consolidation is independent of the mining process and equal for all miners.

5.2 Model Assessment

The CTMC underlying the model is shown in Figure 9, and the description of its states is shown in Table 8.

The steady-state probabilities of the Markov chain are as follows:

$$\begin{aligned}\pi_0 &= \frac{K\beta^2\gamma}{(\beta + \epsilon)(K\gamma + \epsilon)(\beta + (K + N)\gamma + \epsilon)} & \pi_1 &= \frac{\beta\gamma(K\gamma(\beta + \epsilon) + \epsilon(\beta + N\gamma + \epsilon))}{(\beta + \epsilon)(K\gamma + \epsilon)(\beta + N\gamma + \epsilon)(\beta + (K + N)\gamma + \epsilon)} \\ \pi_2 &= \frac{(K - 1)K\beta^2\gamma^2}{(\beta + \epsilon)(K\gamma + \epsilon)(\beta + N\gamma + \epsilon)(\beta + (K + N)\gamma + \epsilon)} & \pi_3 &= \frac{\beta\gamma(KN\gamma + (K - 1)\epsilon)}{(\beta + \epsilon)(K\gamma + \epsilon)(\beta + N\gamma + \epsilon)} \\ \pi_4 &= \frac{\beta\epsilon}{(\beta + \epsilon)(K\gamma + \epsilon)} & \pi_5 &= \frac{\epsilon}{\beta + \epsilon}\end{aligned}\quad (1)$$

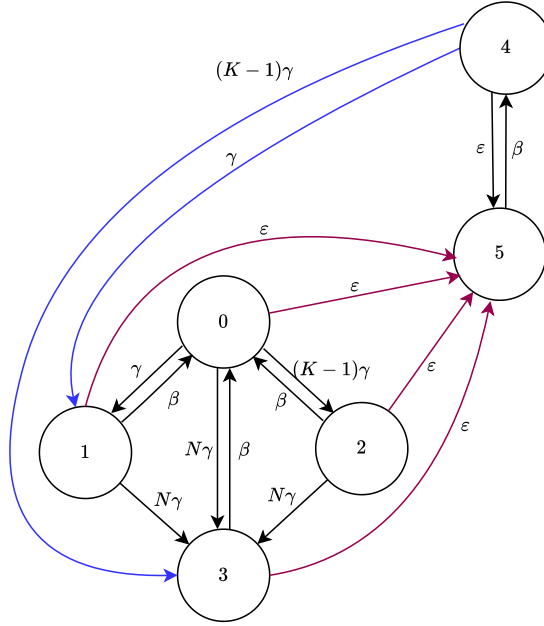


Fig. 9. Markov chain of the model in Table 6.

Table 8. Description of the Markov Chain in Figure 9

State	Short description
0	The creator and all miners are mining.
1	The fair environment is in verification step (V_{E_1}) and the rest of the network (including the creator) is mining.
2	All fair components with exception of one from the environment are verifying (V_F and V_{E_2}) and the rest of the network is mining.
3	All fair components are verifying (V_U and V_{E_U}) and all unfair are mining.
4	The creator and all fair miners are mining, while unfair miners work on the invalid blockchain copy.
5	All fair components are verifying (V_U and V_{E_U}) and the creator is mining, while unfair miners work on the invalid blockchain copy.

The plots in Figure 10 describe the relation between the effective throughput and the rate of invalid block injection ϵ in a network of 100 miners. Figures 10(a)–10(f) study the network with the fraction of fair miners $p = 0.9$, $p = 0.5$, and $p = 1$, respectively. The effective throughput for the unfair miner in Figure 10(b) is about 0.00083 blocks/s if $\epsilon = 0$. This corresponds to the effective throughput for the single unfair miner shown in Figure 6(b). The same holds for the fair miner. Figures 10(a), 10(c), and 10(e) show the impact of invalid block creation on both the environments E_F and E_U in terms of effective throughput, i.e., the throughput of miners that brings them a reward [20].

Figures 10(b), 10(d), and 10(f) are crucial for our reasoning and show the impact of invalid block injection rate ϵ on a single fair and unfair miner. The importance of these plots rely on the fact that

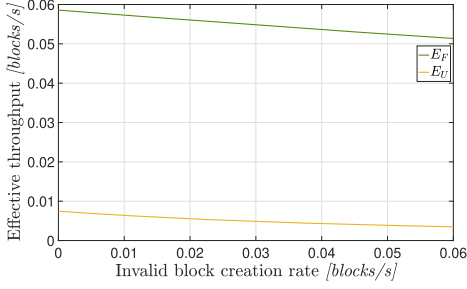
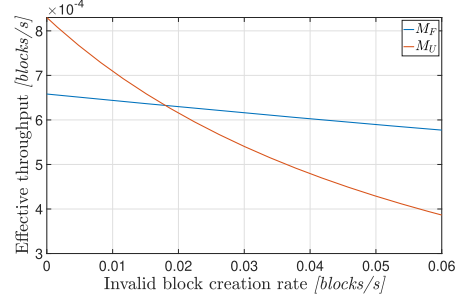
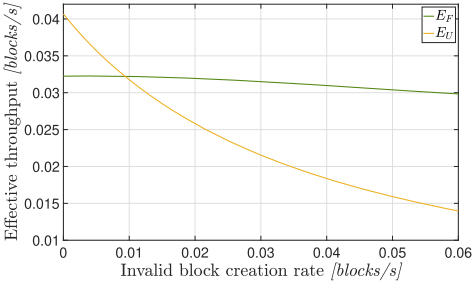
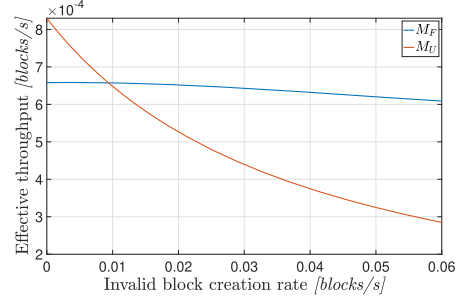
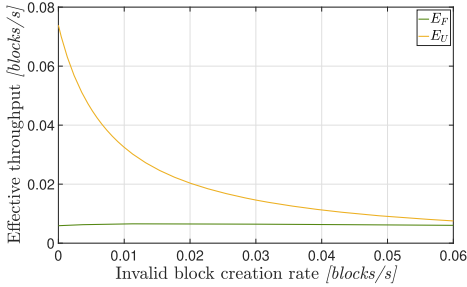
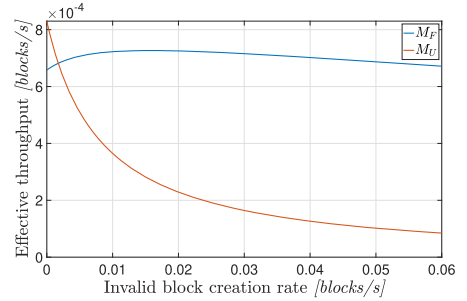

 (a) Effective throughput of 90 fair miners in E_F and 10 unfair miners in E_U

 (b) Effective throughput of a single fair miner M_F and a single unfair miner M_U in the network of 90 fair miners and 10 unfair miners

 (c) Effective throughput of 50 fair miners in E_F and 50 unfair miners in E_U

 (d) Effective throughput of a single fair miner M_F and a single unfair miner M_U in the network of 50 fair miners and 50 unfair miners

 (e) Effective throughput of 10 fair miners in E_F and 90 unfair miners in E_U

 (f) Effective throughput of a single fair miner M_F and a single unfair miner M_U in the network of 10 fair miners and 90 unfair miners

 Fig. 10. Effective throughput as a function of the rate of invalid block creation, where γ and β are taken from Table 5.

rational miners decide whether a fair or unfair behaviour is convenient based on the maximisation of the expected profits. Hence, we desire to choose ϵ in such a way that the effective throughput of a fair miner is higher than that of an unfair one. We know from the base model that for $\epsilon = 0$, rational miners choose not to verify. However, as ϵ grows, we observe a point at which it is more convenient to be fair. What stops us to choose a very large ϵ ? This is explained in the plots of the left-hand side column of Figure 10. Indeed, when ϵ grows, the effective throughput of both fair and unfair miners decrease for different reasons. Indeed, fair miners must also verify the invalid blocks and

Table 9. Coefficients of $P(\epsilon)$

Coefficient	Expression	Sign
c_3	β	+
c_2	$\beta(2\beta + \gamma + 2N\gamma)$	+
c_1	$\beta(\beta^2 + \beta\gamma + 2N\beta\gamma + K\gamma^2 - K^2\gamma^2 + N\gamma^2 - KN\gamma^2 + N^2\gamma^2)$	+/-
c_0	$\beta(K\beta\gamma^2 - K^2\beta\gamma^2 - KN\beta\gamma^2 - K^2N\gamma^3 - KN^2\gamma^3)$	-

unfair miners may be mining their new block after an invalid one, thus, without getting a reward. This leads us to conclude that *the optimal value for ϵ is that at which the effective throughput of a single fair miner is equal to that of a single unfair one*, as shown by the right-hand column of Figure 10.

Another important observation is that the optimal value of ϵ is higher when there are many fair miners. This seems to be quite counterintuitive, but the explanation is connected to the way that forks due to the injection of invalid blocks are resolved. In fact, the presence of many verifying miners helps the unfair ones because they reveal, by announcing their blocks, that the previous one was invalid. Thus, the unfair miner knows earlier that it is wasting its work. As a consequence, to make the unfair behaviour inconvenient the creator has to choose a value of ϵ higher than what would happen if the network had more unfair miners. This observation is proved for small values of γ in Section 6.

6 Optimal Invalid Block Injection Rate

In this section, we analyse the conditions under which it becomes advantageous for a fair miner to shift to unfair behaviour. Although the model does not explicitly account for changes in a miner's type, it is clear that if an unfair miner's profit exceeds that of a fair miner, a rational actor would be inclined to alter their behaviour. Conversely, if both miner types yield the same profit, we can assume the network is in equilibrium, as no miner has an incentive to change their strategy. Thanks to the previous analysis, we can derive the expression of the effective throughput for a single fair (T_F) and unfair miner (T_U)—recall that miners' effective throughput is proportional to their profit:

$$T_F = (\pi_0 + \pi_1 + \pi_4)\gamma, \quad T_U = (\pi_0 + \pi_1 + \pi_2 + \pi_3)\gamma.$$

The optimal injection rate is the rate at which $T_F = T_U$, that is $\pi_4 = \pi_2 + \pi_3$. Therefore, we need to find the real zeros of the rational function $\pi_4 - \pi_2 - \pi_3 = 0$, where these quantities depend on ϵ according to Equation (1).

These zeros correspond to the roots of the polynomial

$$P(\epsilon) = \sum_{i=0}^3 c_i \epsilon^i,$$

where the coefficients are shown in Table 9. Recall that when $M = K + N$ and $p = \frac{K}{M}$ is the fraction of fair miners, then we denote by $P_p(\epsilon)$ the same polynomial where coefficients are expressed as functions of M and p , as described in Table 10.

Theorem 6.1 ensures that there exists a unique solution to our optimisation problem. Hence, it is the first step that we need to provide a closed-form expression for the optimal injection rate.

THEOREM 6.1. *$P(\epsilon)$ has exactly one positive real root.*

Table 10. Coefficients of $P_p(\epsilon)$

Coefficient	Expression	Sign
c_3	β	+
c_2	$\beta(2\beta + \gamma + 2M(1-p)\gamma)$	+
c_1	$\beta(\beta^2 + \beta\gamma + M\gamma^2 + 2M(1-p)\beta\gamma - M^2p\gamma^2 + M^2(1-p)^2\gamma^2)$	+/-
c_0	$\beta(Mp\beta\gamma^2 - M^2p\beta\gamma^2 - M^3(1-p)p\gamma^3)$	-

PROOF. By Descartes' rule of signs [19], the number of sign changes between consecutive (nonzero) coefficients is 1 and then the number of positive roots is exactly 1. This can be easily observed by inspecting the coefficients of $P(\epsilon)$ in Table 9 with some care for the coefficient of the linear term that is negative and for the coefficient of degree 1 that can be either positive or negative.

However, in both cases, the number of sign changes is always 1. $\square$

Since $P(\epsilon)$ has degree 3, one may derive the positive root with the closed-form solution for its roots. However, the complexity of the coefficients may make this step cumbersome. Nevertheless, we can also derive with arbitrary accuracy the optimal injection rate thanks to the bisection method. Indeed, it is easy to observe that $P_p(0) < 0$ while $P_p(\infty) > 0$. Therefore, once an interval $[0, b]$ is found such that the $P_p(b) > 0$, the standard bisection algorithm can be applied.

Unfortunately, in general, it is impossible to estimate the value of p from the observation of the real system, since it is impossible to assess how many miners correctly perform the verification task. However, we solve this problem using the following result that formally supports the experimental observations of Section 5.

THEOREM 6.2. *The optimal invalid block injection rate is monotonically increasing with the fraction of fair miners p . Hence, it reaches its maximum for $p = 1$ provided that $\beta > \frac{M}{M-1}\lambda$, where λ is proportional to the total hash power of the entire network of miners, i.e., $\lambda = M\gamma$.*

PROOF. The proof relies on the observation that, for $\epsilon \geq 0$, it holds that

$$p < q \implies P_p(\epsilon) > P_q(\epsilon).$$

This can be proved by computing the polynomial $Q_{p,q}(\epsilon) = P_p(\epsilon) - P_q(\epsilon)$ and by observing that the coefficients of $Q_{p,q}(\epsilon)$ are all non-negative, hence, $Q_{p,q}(\epsilon) > 0$ for all $\epsilon \geq 0$ provided that $\beta > \frac{M}{M-1}\lambda$ where $\lambda = M\gamma$. Since $P_q(0) < 0$ and, by Theorem 6.1, $P_p(\epsilon)$ must have a unique positive root, this implies that the root of $P_p(\epsilon)$ is strictly higher than that of $P_q(\epsilon)$. $\square$

In conclusion, it is safe for the network to compute the optimal injection rate assuming that $p = 1$ and maintain the same rate. In this way, p will not change even in the presence of rational miners since verifying the blocks is more convenient than cheating.

Since the degree of $P(\epsilon)$ is 3, we have an explicit formula for the optimal invalid block injection rate that can be obtained by setting $p = 1$:

$$\epsilon_{opt} = \frac{1}{2} \left(-\beta - \gamma + \sqrt{\beta^2 + 2\beta\gamma + \gamma^2(2M-1)^2} \right).$$

In other words, ϵ_{opt} is the minimum rate at which we have to inject invalid blocks in the blockchain when all miners are honest in such a way that it is convenient for no miner to change its policy and behave unfairly.

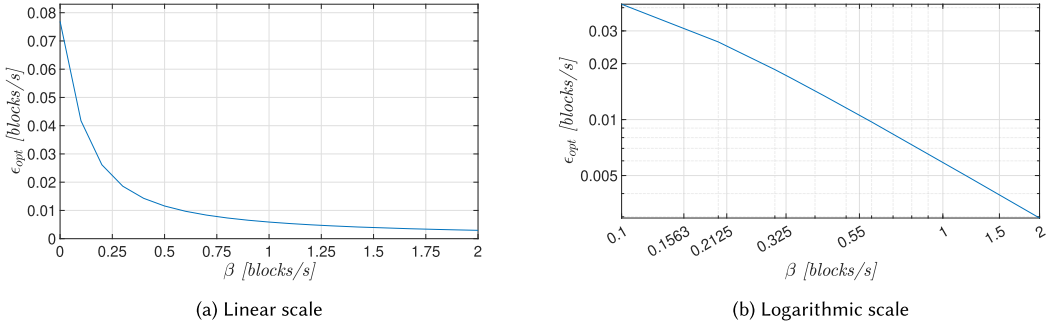


Fig. 11. Optimal injection rate as function of β for the scenario of Example 6.3.

Example 6.3. Let us consider the Ethereum Classic network. In February 2023, the network hash rate is 124.93 TH/s and the average block time is 13 seconds. Suppose that each miner uses one ZOTAC Gaming GeForce RTX GPU with hash rate 125.7 MH/s, then the total number of miners is $M = 994096$. Thus, $\gamma = \frac{1}{13 * 994096} = 7.738 * 10^{-8}$ blocks/s. Suppose that the verification rate is $\beta = 0.314 \text{ s}^{-1}$ as in [1]. In this scenario, the optimal injection rate $\epsilon_{opt} = 1.78318 * 10^{-2}$ blocks/s, i.e., around 1 block per minute.

We consider now the relation between the block injection rate and the verification rate. In general, the average block verification rate depends on the blockchain and may even vary with time. For example, if the workload is low, blocks will be only partially full and, hence, the average verification time will be lower. It is not hard for the blockchain to estimate the verification time and adapt the invalid block injection rate accordingly. Theorem 6.4 states that if the block verification takes more time, then we need to inject more invalid blocks. This is indeed intuitive since longer verification times imply more advantages in skipping the verification.

THEOREM 6.4. *The optimal injection rate ϵ_{opt} is monotonically decreasing with respect to β .*

PROOF. it is sufficient to observe that the derivative of ϵ_{opt} with respect to β is always negative for positive rates and $M > 1$. $\square$

Example 6.5. Let us consider again the scenario depicted in Example 6.3. Figure 11 shows the plots of the optimal injection rate as a function of the verification rate β in linear (11(a)) and logarithmic (11(b)) scales.

Example 6.6. Considering the example with the model parameters from Table 5, we obtain Figure 12, which shows the plots of the optimal injection rate as a function of the verification rate β . We can see that when $\beta = 0.314$ blocks/s, which corresponds to the scenario of Figures 10(b), 10(d), and 10(f), the optimal injection rate for $p = 1$ is approximately 1 block every 50 seconds.

6.1 Discussion of the Model Assumptions and Validity of the Results

This article has introduced a Markovian model to analyze the Verifier's Dilemma and investigates a mitigation strategy in which selected miners inject deliberately incorrect blocks. With this mitigation policy, it is important to assess the rate at which invalid blocks should be injected in the network. We answer this question under some assumptions that allow for an analytical tractability of the problem. The strongest of these assumptions is that the hash rate of each miner that is called to verify the blocks is the same. A violation of this assumption may lead to an inaccurate

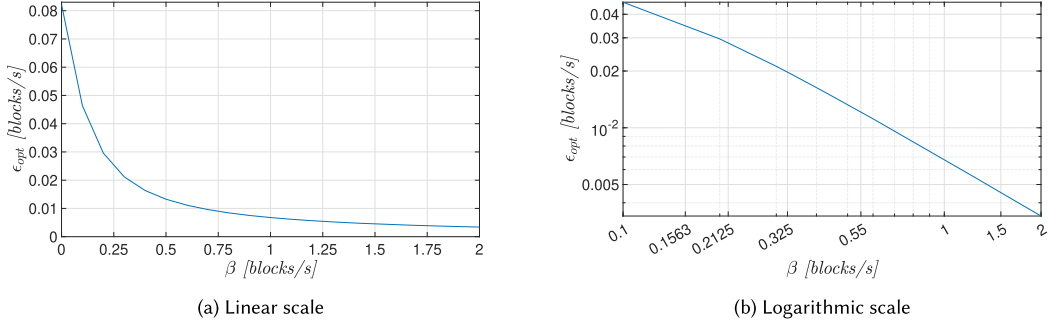


Fig. 12. Optimal injection rate as function of β with the parameters from Table 5.

estimation of the unfair miners' profit and, therefore, to an inaccurate estimation of the optimal invalid block injection rate. However, to the best of our knowledge, our result is the first that quantifies this rate and can still be useful to guide the engineering phase of the system, since it is very hard to understand whether miners actually verify the blocks. We obtain another result —that the optimal injection rate must be computed assuming a situation in which all miners are honest. The rate must be sufficiently high to discourage any miner to assume unfair behavior. We conjecture that this insight remains valid even if there are strong differences in miners' hash rates, coherently with the interpretation that we gave to the result. Finally, our model assumes that miners do not maintain a queue of blocks to verify but rather a new announced block forces miners to start its verification. As we discussed before, this assumption can be relaxed and our model can be modified to work with finite queues of blocks to verify. In these cases, the model is still numerically tractable but loses its analytical properties. However, we observed that the two cases do not differ significantly for the type of analysis we propose here.

7 Conclusion

In this article, we have investigated the Verifier's Dilemma by means of quantitative Markovian models expressed in PEPA. The models allowed us to study the problem by showing that the profits of an unfair (i.e., non-verifying) miner can be significantly higher than those of fair ones and that these depend on the proportion of unfair miners. Indeed, the optimal scenario for an unfair miner is that it is the only one that skips the verification of announced blocks. Then, we have proposed a model for the countermeasure that requires the injection of invalid blocks to encourage unfair miners to abandon their unethical behaviour by making it economically disadvantageous. This latter model has two purposes: (i) it allows us to determine the optimal injection rate, i.e., the minimum rate at which the average reward of an unfair miner is lower than that of a fair one, and (ii) it can be used to estimate the impact of this countermeasure on the overall network throughput and miners' rewards. More specifically, regarding (i), we have shown that the optimal rate can be computed numerically or analytically as the unique positive root of a certain polynomial.

When all miners are currently honest, the optimal invalid block injection rate is maximal and we gave an analytical expression to compute it. Moreover, we showed that this rate depends on the average verification time: the heavier the verification phase, the more invalid blocks we need to inject in the blockchain.

Our future work will take two directions. First, we will study and adapt the model to encompass PoS blockchains. Then, we plan to propose a protocol to reward miners that inject invalid blocks with the aim of controlling the Verifier's Dilemma problem. Miners must be selected randomly according to a certain policy and will be rewarded within the next valid block. This poses the

problem of studying the trade-off between the costs connected to the introduction of this protocol and the improvement in the blockchain security that can be achieved.

References

- [1] Maher Alharby, Roben Castagna Lunardi, Amjad Aldweesh, and Aad van Moorsel. 2020. Data-driven model-based analysis of the Ethereum Verifier's Dilemma. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'20)*. IEEE, 209–220.
- [2] Maher Alharby and Aad van Moorsel. 2020. BlockSim: An extensible simulation tool for blockchain systems. *Front. Blockchain* 3, Art. 28 (2020), 1–16.
- [3] Giacomo Alzetta, Andrea Marin, Carla Piazza, and Sabina Rossi. 2018. Lumping-based equivalences in Markovian automata: Algorithms and applications to product-form analyses. *Information and Computation* 260 (2018), 99–125.
- [4] Parwat Singh Anjana, Sweta Kumari, Sathya Peri, Sachin Rathor, and Archit Somani. 2019. An efficient framework for optimistic concurrent execution of smart contracts. In *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP'19)*. IEEE, 83–92.
- [5] Simonetta Balsamo, Andrea Marin, Isi Mitrani, and Nicola Rebagliati. 2021. Prediction of the consolidation delay in blockchain-based applications. In *ICPE '21: ACM/SPEC International Conference on Performance Engineering, Virtual Event, France, April 19–21, 2021*. 81–92.
- [6] Vitalik Buterin. 2014. Ethereum: A next-generation smart contract and decentralized application platform. *Ethereum White Paper*. (2014).
- [7] Jing Chen and Silvio Micali. 2019. Algorand: A secure and efficient distributed ledger. *Theor. Comput. Sci.* 777 (2019), 155–183.
- [8] Sourav Das, Nitin Awathare, Ling Ren, Vinay J. Ribeiro, and Umesh Bellur. 2021. Tuxedo: Maximizing smart contract computation in PoW blockchains. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5, 3 (2021), 1–30.
- [9] Sourav Das, Vinay Joseph Ribeiro, and Abhijeet Anand. 2018. YODA: Enabling computationally intensive contracts on blockchains with Byzantine and Selfish nodes. *arXiv preprint arXiv:1811.03265* (2018).
- [10] Thomas Dickerson, Paul Gazzillo, Maurice Herlihy, and Eric Koskinen. 2020. Adding concurrency to smart contracts. *Distributed Computing* 33, 3 (2020), 209–225.
- [11] Marko Dimjašević and Nicholas Clarke. 2019. Specification of the Blockchain Layer. Retrieved from <https://github.com/input-output-hk/cardano-ledger/releases/latest/download/byron-blockchain.pdf>
- [12] Beltrán Borja Fiz Pontiveros, Christof Ferreira Torres, et al. 2019. Sluggish mining: Profiting from the Verifier's Dilemma. In *International Conference on Financial Cryptography and Data Security*. Springer, 67–81.
- [13] Jane Hillston. 1996. *A Compositional Approach to Performance Modelling*. Cambridge University Press.
- [14] Jane Hillston, Andrea Marin, Sabina Rossi, and Carla Piazza. 2013. Contextual lumpability. In *Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools*. 194–203.
- [15] Harry Kalodner, Steven Goldfeder, Xiaoqi Chen, S. Matthew Weinberg, and Edward W. Felten. 2018. Arbitrum: Scalable, private smart contracts. In *27th USENIX Security Symposium (USENIX Security'18)*. 1353–1370.
- [16] John G. Kemeny and J. Laurie Snell. 1976. *Finite Markov Chains*. Springer.
- [17] Loi Luu, Jason Teutsch, Raghav Kulkarni, and Prateek Saxena. 2015. Demystifying incentives in the consensus computer. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 706–719.
- [18] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review* (2008).
- [19] Frank W. J. Olver, Daniel W. Lozier, Ronald F. Boisvert, and Charles W. Clark. 2010. *The NIST Handbook of Mathematical Functions*. Cambridge University Press.
- [20] Daria Smuseva, Ivan Malakhov, Andrea Marin, Aad van Moorsel, and Sabina Rossi. 2022. Verifier's Dilemma in Ethereum blockchain: A quantitative analysis. In *Quantitative Evaluation of Systems: 19th International Conference, QEST 2022, Warsaw, Poland, September 12–16, 2022, Proceedings*. Springer, 317–336.
- [21] Jason Teutsch and Christian Reitwießner. 2019. A scalable verification solution for blockchains. *arXiv preprint arXiv:1908.04756* (2019).
- [22] Mirco Tribastone, Adam Duguid, and Stephen Gilmore. 2009. The PEPA eclipse plugin. *SIGMETRICS Perform. Evaluation Rev.* 36, 4 (2009), 28–33.
- [23] Lian Yu, Wei-Tek Tsai, Guannan Li, Yafe Yao, Chenjian Hu, and Enyan Deng. 2017. Smart-contract execution with concurrent block building. In *2017 IEEE Symposium on Service-Oriented System Engineering (SOSE'17)*. IEEE, 160–167.

Received 6 March 2023; revised 25 February 2025; accepted 7 March 2025