

# A Systematic Review and Empirical Analysis of Blockchain Simulators

REMIGIJUS PAULAVIČIUS<sup>1</sup>, SAULIUS GRIGAITIS<sup>1</sup>, AND ERNESTAS FILATOVAS<sup>1</sup>

Institute of Data Science and Digital Technologies, Vilnius University, LT-08412 Vilnius, Lithuania

Corresponding author: Remigijus Paulavičius (remigijus.paulavicius@mif.vu.lt)

**ABSTRACT** In recent years, distributed ledger technologies, and especially blockchain, have gained tremendous interest from academia, government, and industry. Although various blockchain-based solutions were created, the lack of tools to evaluate these complex distributed systems may hinder the development of the field. Many advantages of blockchain systems can be demonstrated only at large scales, e.g., using thousands of nodes. An investigation of different implementations and design choices is complicated and hardly feasible on real systems. Meanwhile, blockchain simulators give the possibility to repeat the complex real-world processes at a low cost. This work provides the first and an up-to-date systematic review and empirical analysis of blockchain simulators. Simulators are easily extensible and can test the performance of distributed ledgers using different settings and parameters on a single computer. The features and limitations of selected simulators are summarized and experimentally validated. Finally, recommendations for potential future research directions in the field are provided.

**INDEX TERMS** Bitcoin, distributed ledger technology, blockchain, simulators, systematic review.

## I. INTRODUCTION

Since its first introduction in cryptocurrency, Bitcoin [1], blockchain has been recognized as a breakthrough technology and has attracted much attention from academia, government, and industry alike [2]–[4]. The distributed and decentralized nature of blockchain enables tamper-evident and tamper-resistant control of the transfer of assets. Today, blockchain technology has been applied to a broad field of various innovative solutions beyond cryptocurrencies (see, e.g., [3]–[7] and references given therein). However, compared to traditional centralized solutions, blockchain's decentralized nature limits its performance [8], and this becomes a significant constraint on applying blockchain in production more broadly [9]. Moreover, blind development of blockchain-based systems without initial performance evaluation may have an immense negative impact during the actual deployment stage [10]. Therefore, performance evaluation is a crucial topic for blockchain and all distributed ledger technologies (DLTs).

The architecture of DLTs is complex; even its simplified abstraction consists of five-layers (we discuss this in detail in Section II): network layer, consensus layer, data layer, execution layer, and application layer. The evaluation

of how various modifications of each layer influence the performance of other layers and the entire system is valuable for both industry and academia. Not coincidentally, there is increasing interest in the performance evaluation techniques of distributed ledger systems. The performance evaluation solutions based on analytical modeling (e.g., Markov chains, queuing models, stochastic Petri nets) leverage mathematical tools to formalize a blockchain system and provide analytical evidence of blockchain performance evaluation. Meanwhile, benchmarking, monitoring, and experimental analysis are specific performance evaluation solutions of DLT systems and stem from empirical analysis. However, all these practical evaluation strategies require the availability of existing systems. We refer interested readers to recent reviews [9], [11] for a more detailed analysis of performance evaluation techniques.

The complexity of large-scale, distributed ledger systems makes the performance evaluation process challenging. Excluding simple experiments, e.g., conducted on a single or a few nodes, experiments on blockchain systems are costly, thus rare. The preparation of a node for testing in a public blockchain network is needed to collect the information from the entire vast area. Moreover, examining solutions in testnets requires considerable effort to encourage users to participate. When a private experimental network is constructed, the information of the whole system can be obtained

The associate editor coordinating the review of this manuscript and approving it for publication was Liangxiu Han<sup>1</sup>.

easier. However, the preparation of many nodes is expensive, and experimental conditions and network configuration cannot be easily modified. In this context, the *simulation* of DLT systems becomes a very promising evaluation approach. Moreover, simulation often emerges as the only alternative to realistically evaluate blockchain performance under different scenarios since neither formal modeling nor deployment of thousands, or even tens of thousands of nodes (e.g., there are more than 11,000 reachable nodes in Bitcoin [12]) would be possible.

Blockchain simulation software – blockchain simulators – give the possibility to repeat real-world processes at a low cost. Blockchain simulators attempt to reproduce a real system's performance and progress over time by implementing and running a simulation model [13]. Simulation models are a particular type of mathematical model of a system. They may be classified as *static* or *dynamic*, *deterministic* or *stochastic*, *discrete* or *continuous* (see [14] for more information on this). Blockchain simulators implement different models for specific resources (e.g., network latency, bandwidth), and each of these models is described by a set of specific parameters [13]. By changing the simulated model variables and conditions, the simulated system can be thoroughly investigated without the need to implement an entire system [10]. This way, blockchain simulators allow users to quickly test system performance using different settings and parameters, study its behavior under various operational scenarios, and, in turn, to better define the proper configuration settings. Therefore, simulators are valuable tools in the development of blockchain protocols and systems. Unfortunately, the design and development process of blockchain simulators is complex. Thus, most blockchain simulators are developed to realistically reproduce only one or several aspects of the system (e.g., mining and miners' behavior, attacks on the network, or data dissimulation). The implementation of other processes (e.g., fixed network delays or approximation of hashing power with Gaussian distribution) is simplified, while some features (e.g., Merkle trees) may be skipped entirely.

Consequently, the selection of the most suitable simulator may be a challenging task. Although recent reviews of blockchain performance evaluation methods and solutions have covered analytical and empirical approaches (see, e.g., [9], [10]), there is still a lack of a review and an in-depth analysis of blockchain simulators. To address this, a systematic review and comparison of blockchain simulators in terms of their features, scope, and limitations is necessary and is the primary motivation for this work. Moreover, we analyze the key features of different simulators and experimentally validate the selected simulators. This study will be relevant for researchers and practitioners when developing secure and effective blockchain systems or analyzing specific features of interest.

The contributions of this paper are the following:

- It provides the first systematic review of existing blockchain simulators.

- It presents an experimental analysis of selected blockchain simulators, and assesses the quality of the simulation results.
- It identifies the current limitations of existing simulators and provides suggestions for future research.

The remainder of the paper is organized as follows. Section II presents an overview of distributed ledger technologies, categorization, and multi-layered abstraction. Section III provides a systematic review and comparison of existing blockchain simulators. Section IV comments on the preparation of input data and parameters required by simulators to reflect the current blockchain networks. Section V experimentally validates selected simulators. Finally, Section VI concludes our work.

## II. AN OVERVIEW OF DISTRIBUTED LEDGER TECHNOLOGIES

A blockchain is a distributed ledger in which trustless nodes across a peer-to-peer network maintain transactions and stores them in a series of back-linked sequential blocks. Blockchain technology is just one way to apply a distributed ledger but is a significant player among all distributed ledger technologies. In this section, we review the categorization of DLTs and blockchain designs from the perspective of abstraction layers.

### A. TAXONOMY OF DLTs

In [9], the authors propose a taxonomy of existing DLTs based on the ledger features. According to this taxonomy, all existing DLTs can be classified into three categories depending on the data architecture:

- Blockchain (e.g., Bitcoin [1], Ethereum [15]).
- Directed acyclic graph (DAG) (e.g., IOTA [16], Nano [17]).
- Others (e.g., Corda [18], Radix [19]).

In Blockchain, data (transactions) is stored in blocks linked via their hash values, forming a kind of a linked list that is immutable, while in DAGs, like IOTA, transactions are linked using reference relations, forming a directed graph.

Based on the accessibility (*private* and *public*) and permissions (*permissioned* and *permissionless*) of the ledger, DLTs can be classified into the following four categories:

- *Public Permissionless* (e.g., Bitcoin, Ethereum).
- *Public Permissioned* (e.g., EOS [20], Ripple [21]).
- *Private Permissionless* (e.g., Holochain [22], LTO Network [23]).
- *Private Permissioned* (e.g., Hyperledger [24], Corda [25]).

In public permissionless distributed ledgers, anyone can participate anonymously, without any restrictions. In public permissioned, participants must be identified, and all participants can read and validate transactions. In private permissionless, only authorized participants can read data, initiate transactions, and participate in the consensus mechanism. Finally, in private permissioned, access is restricted, and only

the network operator can initiate transactions and participate in the consensus mechanism, while authorized participants can only read data [26].

### B. MULTI-LAYERED ABSTRACTION OF DLTs

The authors in [27] define a multi-layered blockchain design comprised of four abstraction layers. The consensus layer defines how nodes agree on a ledger and resolve inconsistencies, the data model layer defines how data is stored, the execution engine layer defines how a distributed ledger becomes an execution environment for computer programs, and the application layer defines what type of applications can be built on top of DLT. Similarly, a slightly different four-layer hierarchical structure, consisting of network (defines how nodes are connected), protocol (analogous to the consensus layer), ledger (analogous to the data layer), and application layers, was defined in [4].

The five-layer abstraction of blockchain, including network, consensus, data, execution, and application layers, was defined in several recent works, see e.g., [2], [9], [11]. To describe the architecture and functionality of blockchain simulators, we adapt the five-layer abstraction of DLT, as shown in Fig. 1. We now discuss these five layers in turn (i.e., network layer, consensus layer, data layer, execution layer, and application layer).

#### 1) NETWORK LAYER

At the bottom of the multi-layered DLT stack is the foundation of a DLT system, a peer-to-peer (P2P) network (see Fig. 1), where participants/peers share resources in a decentralized way, i.e., without central entities. Although all participants in a P2P network are considered equal, there are two basic types of nodes: *full nodes* and *lightweight/light nodes*. Full nodes keep a complete copy of the ledger and take care of mining, transaction validation, and execution of consensus rules. Lightweight nodes (e.g., wallets) only store all block headers and act as clients to issue transactions. The network layer is critical and takes care of peer discovery, transactions and block propagation. For public blockchains (e.g., Bitcoin or Ethereum), where the network is vast, the speed of peer discovery, ledger synchronization, package loss rate, and network delay and propagation may have a tremendous impact on the performance of DLT [9].

#### 2) CONSENSUS LAYER

The role of the consensus layer is to get all nodes in the system to reach an agreement on the DLT system state. Most of the existing consensus algorithms fall into one of the following three categories [11]: i) proof-based, ii) vote/Byzantine Fault Tolerance (BFT)-based, and iii) DAG-based consensus.

Proof-based consensus algorithms (e.g., Proof-of-Work (PoW) [1], [28]) are prevalent among public permissionless DLTs as they can provide high security, scalability, and decentralization in a trustless environment. Unfortunately, the traditional PoW consensus is energy demanding and has a low transaction throughput and

confirmation speed. Therefore, various modifications (e.g., Proof-of-Stake (PoS) [29], Proof-of-Authority (PoA)) have been proposed to maintain the same level of safety but provide better performance.

BFT-based consensus algorithms (e.g., Practical Byzantine Fault Tolerance (PBFT) [30]) are commonly used among private permissioned DLTs. Contrary to PoW, BFT-based consensus supply deterministic results in synchronous and partially synchronous networks and achieve better performance. However, high communication costs cause low network scalability and centralization [31].

DAG-based distributed ledgers [32] organize transactions or blocks in a form of a directed graph instead of a traditional blockchain data structure (see Fig. 1). Based on this, many DAG-based distributed ledgers introduced their consensus mechanisms. For IOTA, a node in a DAG structure called *tangle* is a transaction, instead of a block, and it is validated directly or indirectly by new transactions. In this way, initial transactions are more confident as more new transactions are referring to them. Also, a DAG structure allows parallel transaction/block generation and inclusion. Therefore, DAG-based consensus ensure high network scalability and transaction throughput. Adding transactions to a DAG-based distributed ledger is easy, but DAG-based consensus is vulnerable to malicious participants who validate their transactions. The current workaround, i.e., introducing a trusted validator, is at the expense of losing decentralization, [11].

To sum up, there is no “one-size-fits-all” consensus. Therefore, the optimization of existing and the design of new consensus algorithms remain an actively ongoing research direction in DLT. The interested readers may refer to recent excellent surveys for more information on blockchain consensus algorithms [31], [33]–[35].

#### 3) DATA LAYER

At the data (or storage) layer, the transaction model, the data model, the Merkle tree structure [36], the hash function algorithm (e.g., SHA-256 v.s. Ethash), and the encryption algorithm (e.g., ECC v.s. RSA) should be defined. The elementary unit of a distributed ledger is a transaction. The *unspent transaction output* (UTXO) and *account-based* are two of the most commonly used transaction models. Using UTXO, the sender (owner) transfers value by signing a transaction to transfer the UTXO to the receiver. The total “balance” of a user is the sum of all UTXOs which the user’s private key controls. The account-based data model is more straightforward and updates two accounts in one transaction. For traditional blockchains (e.g., Bitcoin), transactions are gathered into blocks, and each block is linked to the previous block by embedding the hash of the previous block’s header (see Fig. 2). Hashes of transaction pairs within a block are computed using the Merkle tree data structure and stored as Merkle Root, which guarantees integrity of the transactions. Previous hash and Merkle Root, combined with other metadata, namely version, timestamp, difficulty target, and

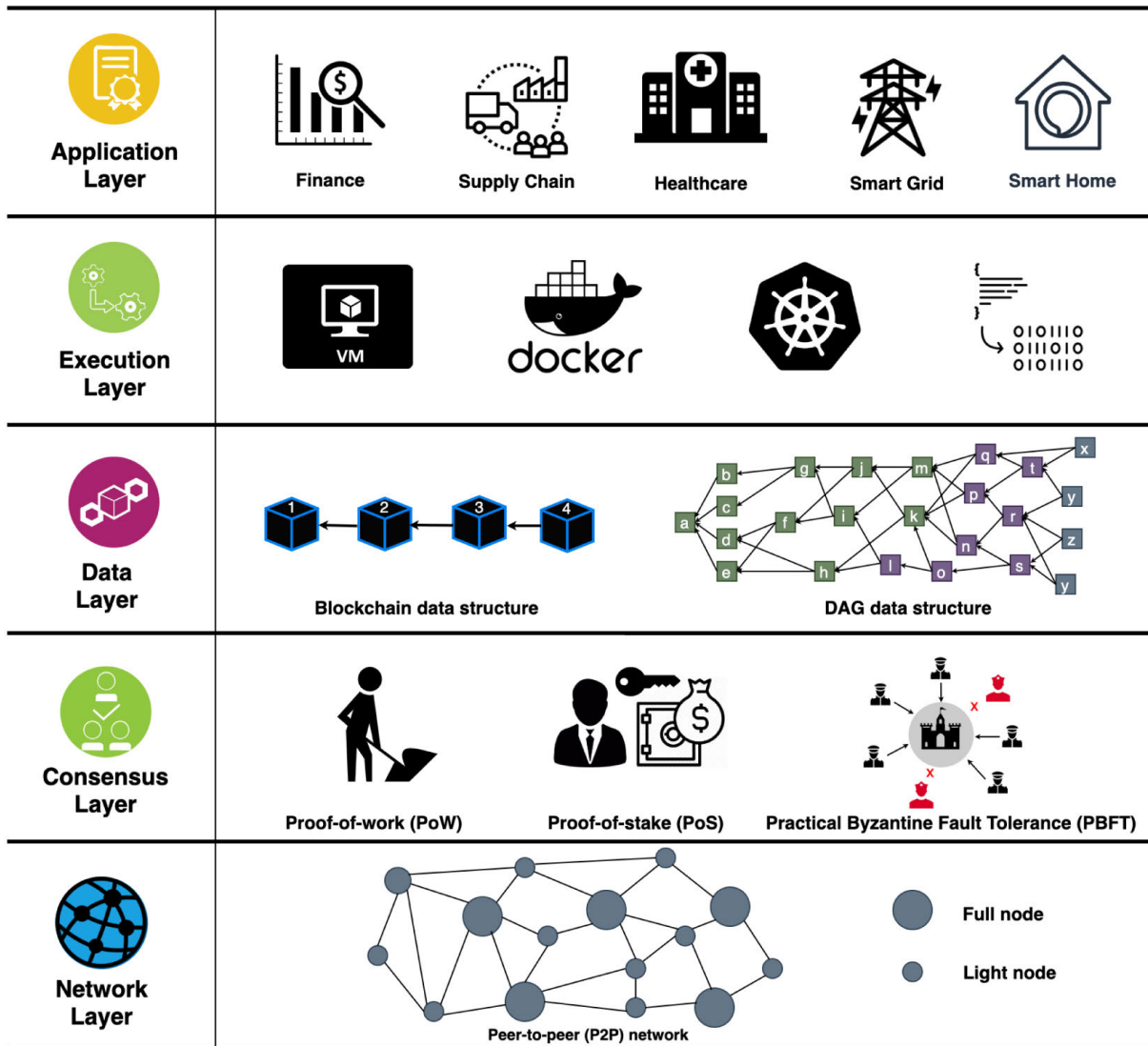


FIGURE 1. The abstraction of DLT as a multi-layered system.

nonce, are used for Bitcoin’s mining process. While such blockchains proved to have good security properties, they also have some well-known drawbacks, e.g., slow transaction processing and energy-demanding mining.

A possible solution to the scalability and efficiency problems is storing blocks/transactions in a DAG data structure, instead of a blockchain. DAG allows mining parallelism – given that enough nodes exist, a new node can be linked without forming a cycle. DAGs can be transaction-based, where transactions are directly linked with each other, and block-based, where blocks that contain transactions can have multiple predecessors and successors. As a result, a DAG-based system has a comparative advantage in performance and scalability over a blockchain. In terms of security, however, it has been shown that fewer resources are needed to launch a double-spending attack on DAG-based systems [11].

Besides the factors mentioned above, there are other design parameters on the data layer, e.g., the type of database used to store states or snapshots. We refer the readers to [9] for more information on this.

4) EXECUTION LAYER

At the execution level, we have runtime environments such as virtual machines (VMs), containers, and compilers installed on DLT network nodes. The level is used to execute *smart contracts* (introduced by Szabo [37]) or low-level machine code (bytecode). Indeed, all blockchains have built-in smart contracts that implement their transaction logic. The execution of smart contracts is determined by the predefined transition rules that will change the contract’s state and assets. In Bitcoin, ‘smart contracts’ are limited functionality *scripts*, while Ethereum has its machine language (bytecode) and a virtual machine (EVM) developed to run them. Moreover,



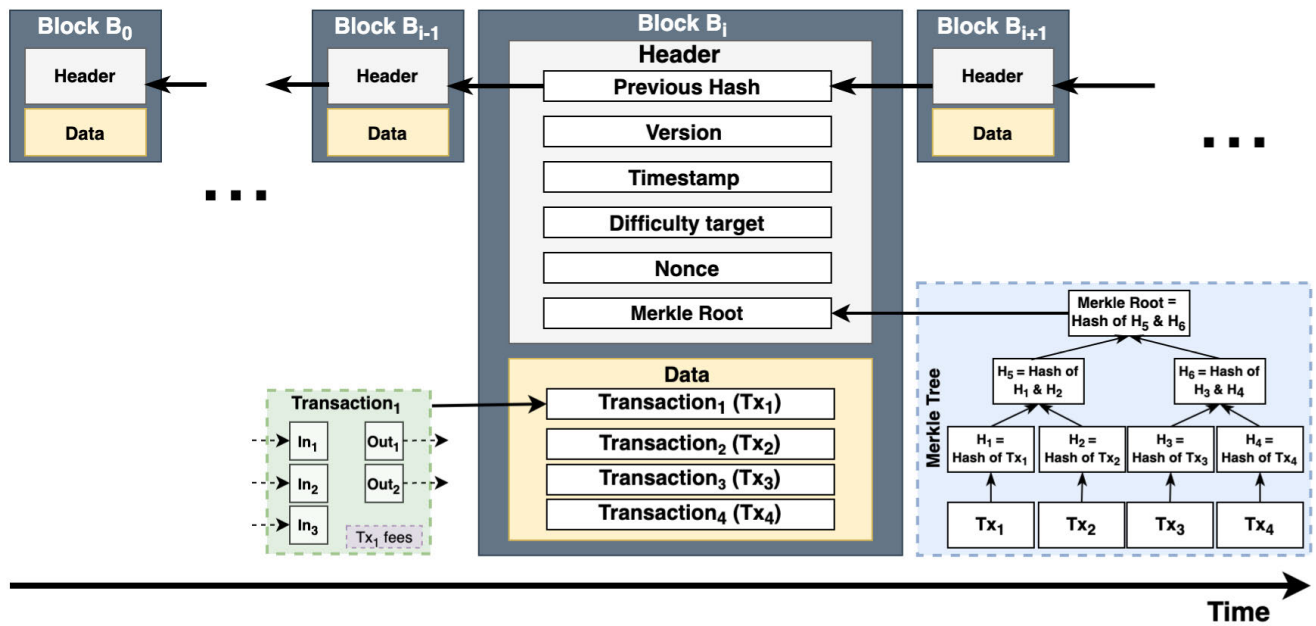


FIGURE 2. The simplified Bitcoin blockchain data structure.

in Ethereum, smart contracts are programmed in a high-level Turing-complete language, Solidity. Hyperledger Fabric takes a step forward and allows smart contracts (chain-code) to be programmed using general high-level programming languages (Go, node.js, and Java). However, not all DLTs support smart contracts, e.g., DAG-based IOTA does not support them yet.

In summary, smart contracts make DLT platforms useful not only for cryptocurrencies but also for various real-world applications. Therefore, the runtime environment used to execute them needs to be efficient and ensure the determinism.

5) APPLICATION LAYER

At the top level, we find various built-in interfaces (APIs) to implement decentralized applications on top of the underlying DLTs. The most popular DLT application remains cryptocurrency. Other popular examples include crypto-money wallets, smart contracts, and various decentralized applications (DApps), such as decentralized autonomous organizations (DAOs) in Ethereum. Most smart contracts are related to cryptocurrency, but in general, they are designed to digitally facilitate, verify, or enforce the execution of any contract in a DLT environment. This layer is in charge of presenting the final results; therefore, the performance evaluation of the application layer reflects the overall performance of DLT [9].

III. SYSTEMATIC REVIEW OF BLOCKCHAIN SIMULATORS

This section systematically reviews and compares the existing blockchain simulators, considering their technical features and providing references for selecting the most suitable blockchain simulator to evaluate specific features of a blockchain system.

A. METHODOLOGY

To identify the key publications on blockchain simulators, we performed a literature search in scientific databases following the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) methodology [38] with minor modifications. The analysis covered the leading publishers of computer science journals and conferences: IEEE Xplore, Springer Link, ACM Digital Library, ScienceDirect, Wiley, SAGE Journals Online, as well as publications in WoS and Scopus databases. To find relevant publications for our research, we used the following search string: ((Blockchain OR Bitcoin OR Ethereum OR Hyperledger OR DAG) AND (Simulator)). The search was performed in titles, abstracts, and keywords. We considered only peer-reviewed, high-quality papers published in conferences, workshops, symposiums, books, and journals related to the research topic. All the searches were conducted in November 2020.

In total, a set of 157 potentially relevant publications, excluding grey literature and pre-prints, was found. Potential duplicates were removed, leaving 111 sources. The titles, keywords, and abstracts were then analyzed to identify publications describing a blockchain system simulation tool/framework or approach. As a result, a provisional collection of 48 publications was composed. Next, the new references found within this collection were analyzed, resulting in 64 publications. Once the literature collection process was completed, we read the selected sources and identified the existing blockchain simulators and their extensions. A complete flowchart of the literature search and selection process is presented in Fig. 3.

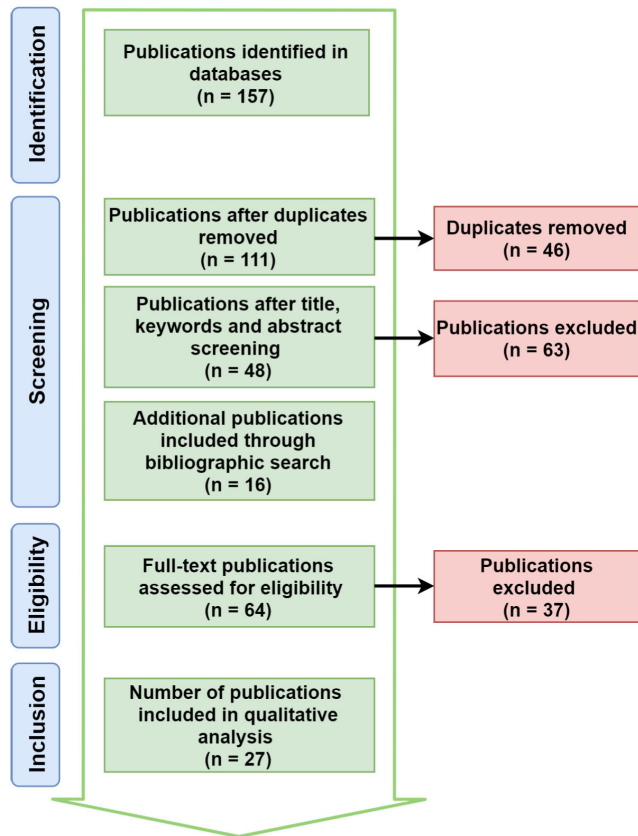


FIGURE 3. The flowchart of the literature search strategy.

## B. SUMMARY OF FINDINGS OF LITERATURE REVIEW

In Table 1, we summarize 27 publications that present the identified blockchain simulators and their extensions. This table includes the following information: simulator name, authors, publication title, year of publication, and the total number of citations in Google Scholar. The first publications introducing blockchain simulators appeared in 2013, while most works (17 out of 27) were published in 2019 and 2020. The vast majority of these works (22 out of 27) were published in conference proceedings. The most cited publications are devoted to Bitcoin protocol simulator, Bitcoin-Simulator, and “Bitcoin privacy simulator.” These works are also among the oldest, which could have impacted the total number of citations. Unfortunately, only the last of these publications has a publicly available source code (see Table 2).

To sum up, the field of development and application of blockchain simulators has received more attention from the research community only recently, and requires significantly more consideration.

## C. OVERVIEW OF BLOCKCHAIN SIMULATORS

In Table 2, we provide further information related to the identified blockchain simulators: the type(s) of platform simulated, blockchain layers analyzed, model type, language and framework (if any) used in the implementation, link to source code (if provided), and the date of the last commit.

PoW-based blockchains still dominate the sector, and account for almost 80% of the total market capitalization<sup>1</sup> (a drop from more than 90% in 2016 [44]). Therefore, it is not surprising that most simulators (21 out of 26) are intended to simulate various PoW-based blockchain platforms where the Bitcoin network simulation is the primary focus.

Usually, simulators’ architecture follows the multi-layered blockchain paradigm and consists of several layers (see Subsection II-B). Simulation models are used to implement the behavior of different layers and their interaction; however, some of them are very simplified. Thus, in the column “Layer(s)” we note only those layers that have more sophisticated models and could be investigated under various parameters and scenarios. Notably, only three simulators – Shadow Bitcoin, Blocksim:Faria, and SIMBA (which is the extension of Blocksim:Faria) – can simulate three layers: consensus, data, and network. Our more in-depth investigation revealed that simulators covering fewer layers usually have more sophisticated models (for more details, see Table 3). Only one simulator, the I-Green simulator, includes the application layer and focuses on the simulation of a system dedicated to distributed renewable energy.

Blockchain simulators typically follow a discrete-event system simulation (18 out of 22 simulators that specified this information) in which states change at a discrete set of points in time. Only three simulators are agent-based. The most utilized programming language is Python, used to build 11 simulators, while five simulators were developed using C++ and Java. Source codes of 17 simulators are publicly available, and eight were updated within the last year.

## D. COMPARATIVE ANALYSIS

In this subsection, we perform an in-depth analysis and feature comparison of five selected simulators.

In the selection of blockchain simulators for our analysis, the main criteria were: (a) availability of the source code; (b) ability to explore a wide range of features of a simulated blockchain network; and (c) possibility to update input data and parameters to reflect current situation in blockchain networks. Following these criteria, five PoW-based blockchain simulators were selected: Bitcoin-Simulator, BlockSim:Faria, SimBlock, BlockSim:Alharby, and VIBES.

Then, after carefully studying the related literature, source codes, and running various examples, we determined a set of possible input and output parameters (see column “Parameter” in Table 3). The main criteria were the simulator’s ability to simulate blockchain networks and provide sufficient output results realistically. In Table 3, we aggregate our observations and note the existing features using ✓, while the missing functions are marked using ✗.

Considering the network layer, Bitcoin-Simulator has the most developed simulation model, providing various

<sup>1</sup><https://cryptoslate.com/cryptos/proof-of-work/>

**TABLE 1. Identified publications presenting blockchain simulators and their extensions (in ascending order by the year of publication).**

Simulator name	Authors & sources	Title	Year	# Citations
"Bitcoin privacy simulator" <sup>a</sup>	Androulaki et al. [39]	Evaluating User Privacy in Bitcoin	2013	563
BTCsim	Brune [40]	Bitcoin Network Simulator	2013	2
Bitcoin protocol simulator	Eyal & Sirer [41]	Majority Is Not Enough: Bitcoin Mining Is Vulnerable	2014	1599
Simbit	Bowe [42]	Javascript P2P Network Simulator	2014	7
Shadow-Bitcoin	Miller & Jansen [43]	Shadow-bitcoin: Scalable simulation via direct execution of multi-threaded applications	2015	39
Bitcoin-Simulator	Gervais et al. [44]	On the Security and Performance of Proof of Work Blockchains	2016	775
Bitcoin mining simulator	Carlsten et al. [45]	On the instability of Bitcoin without the block reward	2016	216
VIBES	Stoykov et al. [46]	VIBES: fast blockchain simulations for large-scale peer-to-peer networks	2017	28
eVIBES	Deshpande et al. [47]	eVIBES: Configurable and Interactive Ethereum Blockchain Simulation Framework	2018	1
CLOTH	Conoscenti et al. [48]	The cloth simulator for HTLC payment networks with introductory lightning network performance results	2018	9
BlockSim:Hao	Hao et al. [49]	BlockP2P: Enabling Fast Blockchain Broadcast with Scalable Peer-to-Peer Network Topology	2019	3
BlockSim:Faria	Faria & Correia [13]	BlockSim: Blockchain simulator	2019	9
BlockSim:Alharby	Alharby & Moorsel [50]	BlockSim: A Simulation Framework for Blockchain Systems	2019	18
SimBlock	Aoki et al. [51]	SimBlock: A Blockchain Network Simulator	2019	32
LUNES-Blockchain	Rosa et al. [52]	Agent-based Simulation of Blockchains	2019	4
Ext1-Bitcoin-Simulator <sup>b</sup>	Sai et al. [53]	Assessing the security implication of Bitcoin exchange rates	2019	2
"Algorand simulator"	Conti et al. [54]	Blockchain Trilemma Solver Algorand has Dilemma over Undecidable Messages	2019	5
Bitcoin network simulator	Azimy & Ghorbani [55]	Competitive Selfish Mining	2019	0
DAGsim	Zander et al. [56]	DAGsim: Simulation of DAG-Based Distributed Ledger Protocols	2019	7
"Mining strategy simulator"	Bruschi et al. [57]	Mine with it or sell it: The superhashing power dilemma	2019	1
"Proof of Prestige Simulator"	Król et al. [58]	Proof-of-Prestige: A Useful Work Reward System for Unverifiable Tasks	2019	5
Ext2-Bitcoin-Simulator	Foytik et al. [59]	A blockchain simulator for evaluating consensus algorithms in diverse networking environments	2020	1
Local Bitcoin Network Simulator	Alsahan et al. [60]	Local Bitcoin Network Simulator for Performance Evaluation using Lightweight Virtualization	2020	3
SIMBA	Fattahi et al. [61]	An Efficient Simulator for Blockchain Applications	2020	0
Ext-BlockSim:Alharby	Alharby et al. [62]	Data-Driven Model-Based Analysis of the Ethereum Verifier's Dilemma	2020	0
Ext-SimBlock	Chin et al. [63]	Simulating Difficulty Adjustment in Blockchain with SimBlock	2020	0
I-Green simulator	Zhao et al. [64]	Individual Green Certificates on Blockchain: A Simulation Approach	2020	1

<sup>a</sup> The simulators not named by their authors were given names in quotation marks.

<sup>b</sup> Names of the extended versions of simulators were given a prefix "Ext".

input parameters and outputs of various calculated metrics. SimBlock and Blocksim:Faria are also rich in features and can simulate, e.g., the network layer by taking into account the geographical distribution of the nodes and the geographical bandwidth and latency distributions. Nevertheless, an important limitation of SimBlock

is that it does not separate full nodes and miners. VIBES and Blocksim:Alharby lack the ability to simulate network behavior realistically, e.g., both use fixed block and transaction propagation delays. At the moment, BlockSim:Faria simulator is suitable only for investigations with a small number of nodes. Authors of

**TABLE 2. Overview of existing blockchain simulators and their extensions.**

Simulator name	Platform(s)	Layer(s)	Model type	Language/ Framework	Source code	Last commit
"Bitcoin privacy simulator"	Bitcoin	data	discrete-event	C++	not available	N/A <sup>a</sup>
BTCsim	Bitcoin	consensus, network	discrete-event	Python	<a href="https://github.com/rbrune/btesim">https://github.com/rbrune/btesim</a>	17/12/2017
Bitcoin protocol simulator	Bitcoin	consensus	discrete-event	not specified	not available	N/A
Simbit	Bitcoin	consensus, network	discrete-event	Javascript	<a href="https://github.com/ebfull/simbit">https://github.com/ebfull/simbit</a>	27/05/2016
Shadow-Bitcoin	Bitcoin	consensus, data, network	discrete-event	Python/Shadow	<a href="https://github.com/shadow/shadow-plugin-bitcoin">https://github.com/shadow/shadow-plugin-bitcoin</a>	11/04/2018
Bitcoin-Simulator	PoW blockchains	consensus, network	discrete-event	C++/NS3	<a href="https://github.com/arthurgervais/Bitcoin-Simulator">https://github.com/arthurgervais/Bitcoin-Simulator</a>	13/10/2016
Bitcoin mining simulator	Bitcoin	consensus	not specified	C++	<a href="https://github.com/citp/mining-simulator">https://github.com/citp/mining-simulator</a>	07/07/2017
VIBES	PoW blockchains	consensus, data	discrete-event	Scala	<a href="https://github.com/i13-msrg/vibes">https://github.com/i13-msrg/vibes</a>	25/04/2020
eVIBES	Pow Ethereum	consensus, data	discrete-event	Scala	<a href="https://github.com/i13-msrg/evibes">https://github.com/i13-msrg/evibes</a>	12/12/2018
CLoTH	Bitcoin	network	discrete-event	Go	not available	N/A
BlockSim:Hao	PoW blockchains	network	discrete-event	Java/PeerSim	not available	N/A
BlockSim:Faria	Bitcoin, Ethereum	consensus, data, network	discrete-event	Python/SimPy	<a href="https://github.com/carlosfaria94/blocksim">https://github.com/carlosfaria94/blocksim</a>	19/05/2020
BlockSim:Alharby / Ext:BlockSim:Alharby	PoW blockchains	consensus, data	discrete-event	Python	<a href="https://github.com/maher243/BlockSim">https://github.com/maher243/BlockSim</a>	05/12/2020
SimBlock	PoW blockchains	consensus, network	discrete-event	Java	<a href="https://github.com/dsg-titech/simblock">https://github.com/dsg-titech/simblock</a>	22/08/2020
LUNES-Blockchain	Bitcoin	consensus, network	agent-based	ARTIS+GAIA	available upon e-mail request	04/01/2018
Ext1-Bitcoin-Simulator	PoW blockchains	consensus, network	discrete-event	C++/NS3	<a href="https://github.com/ashishrsai/BTC">https://github.com/ashishrsai/BTC</a>	26/04/2019
"Algorand simulator"	Algorand	consensus	not specified	Java	not available	N/A
Bitcoin network simulator	Bitcoin	consensus	discrete-event	Python	not available	N/A
DAGsim	DAG	consensus, network	agent-based	Python	<a href="https://github.com/IC3RE/DAGsim">https://github.com/IC3RE/DAGsim</a>	25/03/2020
"Mining strategy simulator"	Bitcoin	consensus	agent-based	Python/Mesa	<a href="https://github.com/lorenzogentile404/incentive-network-simulator">https://github.com/lorenzogentile404/incentive-network-simulator</a>	05/10/2018
"Proof of Prestige Simulator"	Proof-of-Prestige	consensus	not described	Python	<a href="https://gitlab.com/mharnen/pop">https://gitlab.com/mharnen/pop</a>	15/03/2019
Ext-Bitcoin-Simulator	Raft	consensus, network	discrete-event	C++/NS3	not available	N/A
Local Bitcoin Network Simulator	Bitcoin	network	virtualization-based	Python	<a href="https://github.com/noureddinel/core-bitcoin-net-simulator">https://github.com/noureddinel/core-bitcoin-net-simulator</a>	11/02/2020
SIMBA	Bitcoin	consensus, data, network	discrete-event	Python/SimPy	<a href="https://github.com/nyit-vancouver/SIMBA">https://github.com/nyit-vancouver/SIMBA</a>	20/05/2020
Ext-SimBlock	PoW blockchains	consensus, network	discrete-event	Java	<a href="https://github.com/Z-Hau/SimBlock-with-Difficulty-Adjustment">https://github.com/Z-Hau/SimBlock-with-Difficulty-Adjustment</a>	06/07/2020
I-Green simulator	PoW, PoS, Proof-of-Generation	application, consensus	not specified	Python	not available	N/A

<sup>a</sup> N/A - Not Applicable



TABLE 3. Key features of selected simulators.

Layer	Parameter description	Simulator name				
		Bitcoin Simulator	BlockSim: Faria	SimBlock	BlockSim: Alharby	VIBES
<b>Input data</b>						
Network	Block size distribution / Fixed block size	✓/✓	X/✓	X/✓	X/✓	X/✓
	Transaction size distribution / Fixed transaction size	X/X	X/✓	X/X	✓/X	X/✓
	Total number of reachable network nodes	✓	✓	✓	✓	✓
	Geographic node distribution	✓	✓	✓	X	X
	# of neighbour connections per node distribution / Fixed	✓/X	X/✓	✓/X	X/X	X/✓
	Geographic miners (mining pools) distribution	✓	✓	X	X	X
	# of neighbour connections per miner distribution	✓	X	X	X	X
	Geographical bandwidth distribution in the network	✓	✓	✓	X	X
	Geographical latency distribution in the network	✓	✓	✓	X	X
	Information propagation mechanism	✓	✓	✓	X	X
Consensus	Consensus algorithm PoW / Other	✓/X	✓/X	✓/✓	✓/X	✓/X
	Block interval distribution / Fixed block interval	✓/✓	X/✓	X/✓	X/✓	X/✓
	Total number of generated blocks	✓	✓	✓	✓	✓
	Mining power distribution of miners (mining pools)	✓	✓	✓	✓	✓
	Dynamic difficulty adjustment	X	X	X	X	X
	Transaction fee distribution / Fixed transaction fee	X/X	X/X	X/X	✓/✓	X/✓
Data	Transaction generation ability	X	✓	X	✓	✓
	UTXO model / Account-based model	X/X	X/X	X/X	X/X	X/X
<b>Total number of available inputs</b>		15	13	12	10	10
<b>Output data (calculated statistics)</b>						
Network	Average block size	✓	✓	X	✓	✓
	Average number of connections per node	✓	X	X	X	X
	Average number of connections per miner	✓	X	X	X	X
	Average (median) miners block propagation time (delay)	✓	✓	✓	X	X
	Average (median) transactions propagation time (delay)	X	X	X	X	✓
	Throughput (tps)	X	X	X	✓	✓
Consensus	Average (median) block interval	✓	X	✓	X	✓
	Total number of generated blocks	✓	✓	✓	✓	X
	Mined blocks distribution	X	✓	X	✓	X
	Stale (orphan) block rate (percentage)	✓	X	✓	✓	✓
	Security provision or parameters	✓	X	X	X	✓
General	Simulation time compared to real time	✓	X	✓	X	X
	<b>Total number of available outputs</b>	9	4	5	5	6

Bitcoin-Simulator, VIBES, and SimBlock declared the ability to simulate large networks. However, in the case of VIBES, simulations of real-size public networks consisting of more than 10,000 nodes (as in Bitcoin [12] and Ethereum [65]) are complicated as this simulator utilizes a centralized coordinator, which is a severe performance bottleneck. The drawback of all simulators is that they typically use a fixed number of nodes, which is not in line with public blockchain networks.

Analyzing the features through the consensus layer perspective, most of the simulators focus solely on PoW algorithms and only SimBlock has the simulation ability of abstracted PoS. Some simulators, e.g., BlockSim:Alharby and SimBlock, assume that all nodes are honest; thus, they cannot be used to explore the strategies of the nodes' malicious behaviors (e.g., selfish mining, double-spending). In contrast, Bitcoin-Simulator and VIBES can be used to test PoW consensus security and investigate the potential of attacks on various PoW

blockchain networks. Unfortunately, no simulator implements dynamic difficulty adjustment during the mining.

As for the data layer, BlockSim:Faria, VIBES, and Blocksim:Alharby support transaction simulation. However, this process is limited, as simulators do not implement any transaction accounting model (UTXO or account-based). The main limitations of Bitcoin-Simulator and SimBlock are that they simulate a blockchain network only at the block level, i.e., the propagation of transactions is not considered.

Finally, the total number of available input and output parameters (features) was calculated for each simulator. Bitcoin-Simulator has the most features, while BlockSim:Faria and SimBlock take the second and third place accordingly.

To sum up, there is no universal simulator that could be applied to a wide range of scenarios. The capabilities of existing simulators are limited as they are designed to simulate a few critical aspects of an actual

blockchain, keeping the rest simplified. Therefore, for an in-depth experimental validation in Section V, we selected three of the most promising (based on the comparative analysis provided in this section) PoW-type simulators – Bitcoin-Simulator [44], SimBlock [51], [66], and BlockSim:Alharby [50]) – and tested them versus the current Bitcoin network.

#### IV. EXPERIMENTAL SETUP

The validation of simulators requires the input data to reflect the actual blockchains. In this section, we comment on preparing the input data corresponding to the present situation of the Bitcoin network.

##### A. INPUT DATA FOR NETWORK LAYER

Simulators are useful only if the simulated network resembles the real-world network [67]. Below, we provide details on the input data used for the network layer simulation by the selected simulators.

##### 1) BLOCK SIZE DISTRIBUTION

To measure block size distribution in Bitcoin, we used Python library `python-bitcoinlib` [68]. The period of three months (September–November 2020) was considered, and different block size intervals (see Fig. 4) were extracted from a full Bitcoin node (Bitcoin Core<sup>2</sup>). Our analysis shows that the average block size in 2020 is 1.19 Megabytes (MB), while the standard deviation is 0.26 MB. More than 82% of the generated blocks fit in intervals between 1.1 and 1.5 MB. Note that Bitcoin’s block size distribution in recent years has changed substantially. Thus, the situation in 2020 differs significantly from the situation in, e.g., 2016. According to [44], the average block size in 2016 was 535 KB. However, the original block size limit of 1,000,000 bytes (1 MB) changed with the introduction of SegWit (Segregated Witness) [69] and its activation in 2017 [70]. A new *weight* parameter was defined, and now blocks are allowed to have up to 4 million block “weight units”. In practice, “weight units” are bytes, therefore the maximum block size is the same as the block weight limit.

Finally, let us note that for simulators that support only a fixed block size, the average value of 1.19 MB was used in our experimental validation and comparison (see Table 7 and Section V-B).

##### 2) GEOGRAPHIC NETWORK LATENCY AND BANDWIDTH DISTRIBUTIONS

Network latency and bandwidth play a vital role in the overall performance of blockchains. Latency is especially critical for mining nodes since they need to be working on top of the most recent block. Moreover, when a new block is found, it must be shared with all other nodes. If the communication of new blocks takes too long, the stale block rate increases, which reduces the security of an entire blockchain

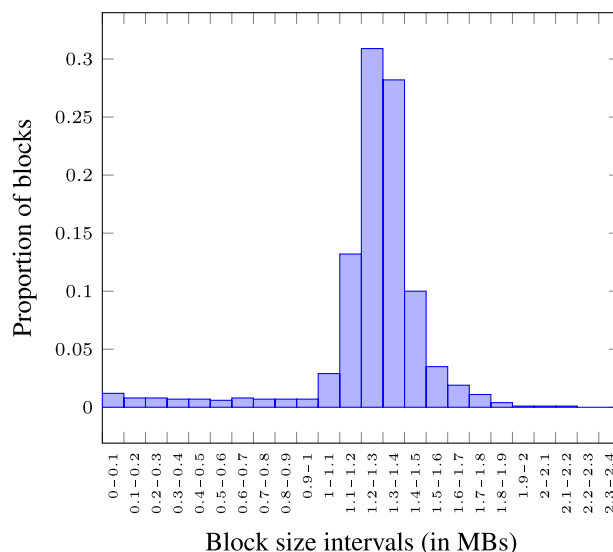


FIGURE 4. The proportion of generated blocks in Bitcoin within each 0.1 Megabyte (MB) size interval.

system. Therefore, the input data corresponding to the network latency and bandwidth should correspond to the real situation.

Two out of three simulators (Bitcoin-Simulator and SimBlock) establish connections between nodes by creating point-to-point channels characterized by network latency and bandwidth (see Tables 3 and 7). Realistic network latencies and bandwidth are taken into account by adopting the global IP latency statistics from Verizon [71] and bandwidth distribution from `testmy.net`.<sup>3</sup> In Tables 4 and 5, we provide the data on network latency (in ms) and bandwidth (download and upload speeds in Mbps) in 2016 (taken from the source-code of Bitcoin-Simulator [72]) and 2019 (taken from the source-code of SimBlock [73]) according to geographical location. As the difference in network latency and bandwidth in 2019 and 2020 is negligible, the data from 2019 will be used as representative of the current situation.

##### 3) GEOGRAPHICAL DISTRIBUTION OF NODES AND MINERS

In Bitcoin, there are two types of nodes: (i) regular nodes and (ii) miners. In Figs 5 and 6, we present the geographical distribution of regular nodes and miners. The data on regular nodes in 2016 is taken from [44], while the situation in 2020 was retrieved from [12]. In 2020, the real geographical location of more than a quarter of nodes was unknown as these nodes operated under the Tor network. To use realistic network parameters (e.g., network latency and bandwidth), Bitcoin-Simulator automatically splits the percentage of ‘Other’ between the regions based on existing proportions (see Table 6). To have comparable results, we have used this modified geographic distribution as the input for other simulators. Among the simulators considered for experimental validation, this was relevant to SimBlock.

<sup>2</sup><https://bitcoin.org/en/bitcoin-core/>

<sup>3</sup><https://testmy.net/country>

**TABLE 4.** Comparison of network latencies (in ms) according to geographical location. Data for 2016 is taken from the source-code of Bitcoin-Simulator [72], while data for 2019 is taken from the source-code of SimBlock [73].

	North America		Europe		South America		Asia Pacific		Japan		Australia	
	2016	2019	2016	2019	2016	2019	2016	2019	2016	2019	2016	2019
North America	36	32	119	124	255	184	310	198	154	151	208	189
Europe	119	124	12	11	221	227	242	237	266	252	350	294
South America	255	184	221	227	137	88	347	325	256	301	269	322
Asia Pacific	310	198	242	237	347	325	99	85	172	58	278	198
Japan	154	151	266	252	256	301	172	58	9	12	163	126
Australia	208	189	350	294	269	322	278	198	163	126	22	16

**TABLE 5.** Comparison of network bandwidth (download and upload speeds in Mbps) according to geographical location. Data for 2016 is taken from the source-code of Bitcoin-Simulator [72], while data for 2019 is taken from the source-code of SimBlock [73].

	Download speed		Upload speed	
	2016	2019	2016	2019
North America	41.7	52	6.7	19.2
Europe	21.3	40	6.7	20.7
South America	9.9	18	2.2	5.8
Asia Pacific	14.6	22.8	6.5	15.7
Japan	6.9	22.8	1.7	10.2
Australia	16	29.9	6.1	11.3

**TABLE 6.** Geographical node distributions in the actual Bitcoin network and values obtained using Bitcoin-Simulator with the input data and parameters corresponding to 2020.

	Bitcoin network	Bitcoin-Simulator
North America	19.51%	27.12%
Europe	51.59%	55.39%
South America	0.65%	0.94%
Asia Pacific	9.2%	12.35%
Japan	1.98%	2.53%
Australia	1.21%	1.68%
Other	26.12%	0.0%

The mining pool’s geographic distribution in 2016 is taken from [44], while the new data for 2020 was retrieved from [74] and represents the aggregate geolocation data based on hashers’ IP addresses connected to mining pools.<sup>4</sup>

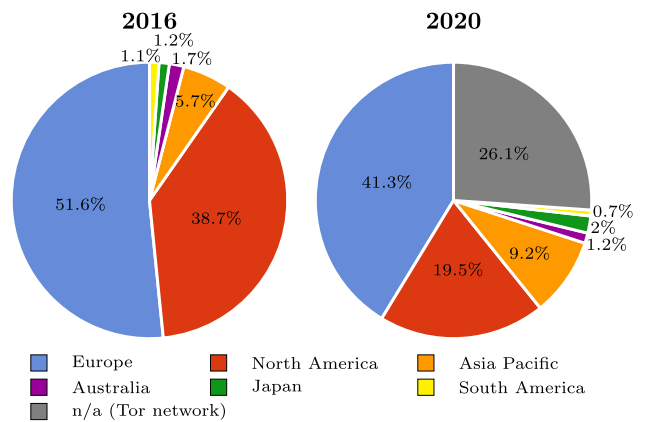
Finally, in two of the selected simulators (SimBlock and Bitcoin-Simulator), the number of neighbor connections per node and per miner follows the distribution described in [75].

**B. INPUT DATA FOR CONSENSUS LAYER**

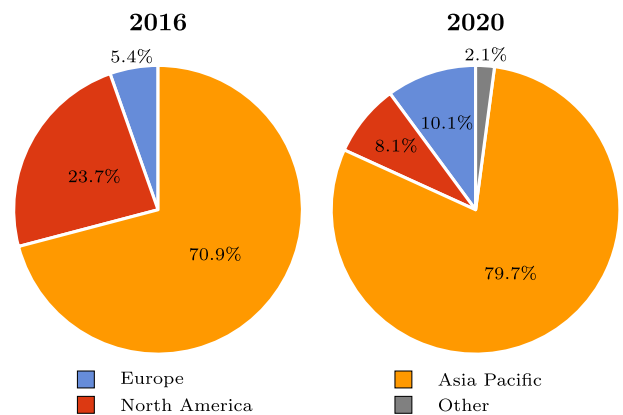
1) BLOCK INTERVAL DISTRIBUTION

The *block interval* (or the *block generation time*) defines the latency at which content is written to the blockchain. The shorter the block interval is, the faster transactions

<sup>4</sup>[https://cbeci.org/mining\\_map/methodology](https://cbeci.org/mining_map/methodology)



**FIGURE 5.** Comparison of the geographical distribution of Bitcoin nodes. Distribution of nodes in 2016 is taken from [44], while distribution in 2020 is calculated based on data from [12].



**FIGURE 6.** Comparison of the geographical distribution of Bitcoin miners (mining pools) hashrate. Distribution in 2016 is taken from [44], while distribution in 2020 is calculated based on data from [74].

are confirmed, but this results in a higher probability of stale blocks [44]. The block generation time in PoW-type blockchains directly relates to the difficulty change of the underlying PoW protocol. For example, Bitcoin’s difficulty target (256-bit hash) is adjusted every 2,016 blocks (around every two weeks) based on the time it took to mine the previous 2,016 blocks. This way, the protocol seeks that the average block generation time stays around ten minutes.

We measured Bitcoin’s block interval distribution using the same data collection strategy as for measuring the block size distribution. The time intervals between the generation of consecutive blocks were extracted. Our analysis shows that the average block generation time is 595 seconds (approximately 10 minutes), while the standard deviation is 600 seconds (10 minutes). Therefore, only 64% of the blocks were generated in less than 10 minutes, while the remaining 36% required between 10 minutes and 2 hours.



In [76], the authors show that the block generation distribution in Bitcoin could be modeled with a shifted geometric distribution. Let the time be divided into time intervals  $(t_0, t_1], \dots, (t_{n-1}, t_n], \dots$  of equal size  $\Delta$ , where  $\Delta = t_i - t_{i-1}, \forall i = 1, 2, \dots$ , and  $t_0 = 0$  denotes the time of the last block generation. Additionally, let the random variable  $X_k$  denote the event of success, i. e., a new block being generated within the time interval  $k = (t_{k-1}, t_k]$ :

$$X_k = \begin{cases} 1, & \text{if a new block is mined within } (t_{k-1}, t_k] \\ 0, & \text{otherwise.} \end{cases}$$

If the probability of successfully generating a new block within each time interval  $k$  is  $\text{Prob}(X_k = 1) = p$ , and  $0 < p \leq 1$ , then the probability that the new block will be generated within the time interval  $k$  (assuming the failure of this during the previous  $k - 1$  time intervals) is

$$\text{Prob}(X_k = 1) \cdot \prod_{i=1}^{k-1} \text{Prob}(X_i = 0) = p \cdot (1 - p)^{k-1}. \quad (1)$$

Authors in [76] show that assuming two-minute time intervals (i.e.,  $\Delta = 2$ ), the block generation distribution follows a shifted geometric distribution (1) with the success probability  $p = 0.19$ .

A generation of new blocks in Bitcoin-Simulator [44] is based on this result. Therefore, we tested if this is a valid assumption in 2020. For our experiment, we considered  $\Delta$  to be the same 2 minutes as in [76]. In Fig. 7, we depict the distribution of generated blocks within each two-minute time interval (see  in Fig. 7) and the shifted geometric distribution with  $p = 0.19$  (see  in Fig. 7). This graph confirms that the Bitcoin block generation distribution in 2020 still matches the shifted geometric distribution with  $p = 0.19$ .

## 2) MINING POWER DISTRIBUTION OF MINERS (MINING POOLS)

For the selected simulators (see Table 7), we adopted the distribution of mining power (hashrate) of Bitcoin miners (mining pools) based on the data from [77] (see Fig. 8).

## C. SUMMARY OF INPUT PARAMETERS

For our experimental validation of simulators, we used the prepared input data and set additional input parameters to reflect the current characteristics of actual blockchains. In Table 7, we summarize the values used for each selected simulator; these values represent the Bitcoin situation in 2020. We also provide additional information

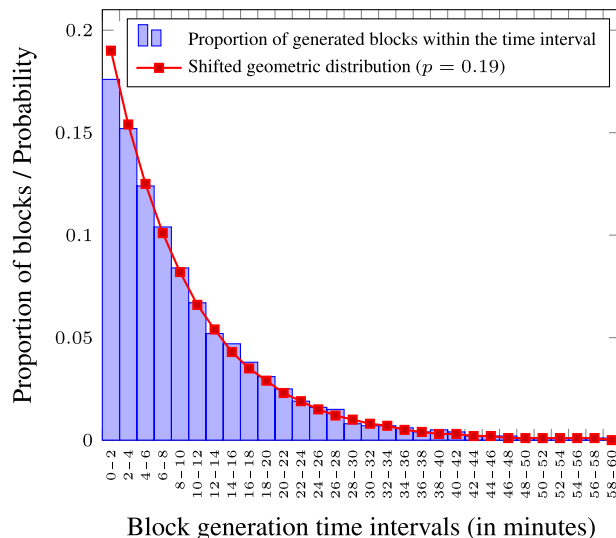


FIGURE 7. The proportion of generated blocks in Bitcoin within each 2-minute time interval, and a shifted geometric distribution with success probability  $p = 0.19$ .

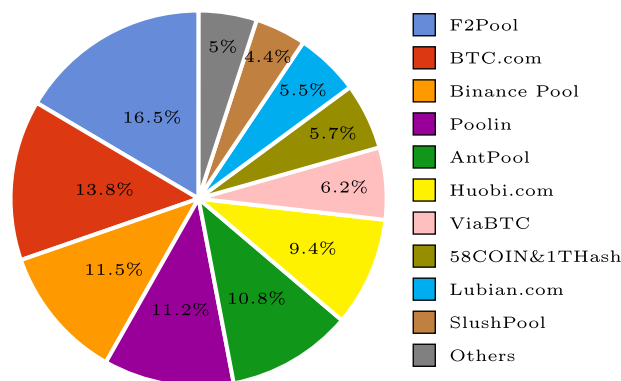


FIGURE 8. Hashrate distribution among the largest mining pools [77].

about modified variables in the source-code, when no interface has been made to update them, e.g., using the command line parameters. Additionally, more detailed technical descriptions of existing parameters, default values and other important information related to input parameters of selected simulators’ are provided in Appendix, Tables 9 to 11.

## V. VALIDATION OF SIMULATORS IN BITCOIN

The development of real blockchain networks is very active. Unfortunately, this is not the case with blockchain simulators as most of the developed simulators are not actively maintained or updated (see column “Last commit” in Table 2). This section’s primary goal is to experimentally validate whether existing (selected) simulators can realistically simulate current blockchains by limiting this investigation to the Bitcoin network.

**TABLE 7. Description of input parameters and values set for selected blockchain simulators. Values specified here correspond to the situation in 2020.**

Parameter	Bitcoin-Simulator	SimBlock	BlockSim
Block size distribution / Fixed block size	Bitcoin's block size distribution (see Section IV-A1); Set by updating variables <i>intervals</i> and <i>weights</i> in <code>bitcoin-miner.cc</code> source-file	1,250,000; Set using <code>BLOCK_SIZE</code> parameter (unit: bytes)	1.22; Set using <code>B<sub>size</sub></code> parameter (unit: Megabytes)
Total number of reachable network nodes	11,000; Set using <code>nodes</code> parameter	11,000; Set using <code>NUM_OF_NODES</code> parameter	11,000; Set using <code>N<sub>n</sub></code> parameter
Geographical node distribution	Geographical Bitcoin's node distribution (see Section IV-A3)); Set by updating variable <i>nodesDistributionWeights</i> in <code>bitcoin-topology-helper.cc</code>	Geographical Bitcoin's node distribution (see Table 6); Set using <code>REGION_DISTRIBUTION</code> parameter	Unsupported
# of neighbour connections per node distribution / Fixed	Distribution according to Miller et al. [75]	Distribution according to Miller et al. [75]	Unsupported
Geographical distribution of miners (mining pools)	Geographical miner distribution (see Section IV-A3); Set by updating variable <i>bitcoinMinersRegions</i> in <code>bitcoin-miner.cc</code>	Unsupported	Unsupported
# of neighbour connections per miner distribution	Distribution according to Miller et al. [75]	Unsupported	Unsupported
Geographical bandwidth distribution in the network	Geographical (six regional) network bandwidth distribution from [73]; Set by updating variables <i>m<sub>regionDownloadSpeeds</sub></i> and <i>m<sub>regionUploadSpeeds</sub></i> in <code>bitcoin-topologyhelper.cc</code>	Geographical (six regional) network bandwidth distribution from [73]; Set using <code>UPLOAD_BANDWIDTH</code> and <code>DOWNLOAD_BANDWIDTH</code> parameters	Unsupported
Geographical latency distribution in the network	Geographical (six regional) network latency distribution from [73]; Set by updating variable <i>regionLatencies</i> in <code>bitcoin-topologyhelper.cc</code>	Geographical (six regional) network latency distribution from [73]; Set using <code>LATENCY</code> parameter	Unsupported
Information propagation mechanism	Legacy protocol with unsolicited block push and relay network support; Set using <code>unsolicitedRelayNetwork</code> parameter	Updated protocol with the support of compact block relay [66]; Set using <code>CBR_USAGE_RATE</code>	Unsupported
Block interval distribution / Fixed block interval	Shifted geometric distribution with $p = 0.19$ (see Section IV-B1)	600,000; Set using <code>INTERVAL</code> parameter (unit: milliseconds)	600; Set using <code>B<sub>interval</sub></code> parameter (unit: seconds)
Total number of generated blocks	10,000; Set using <code>noBlocks</code> parameter	10,000; Set using <code>END_BLOCK_HEIGHT</code> parameter	~10,000; Set using <code>B<sub>interval</sub></code> and <code>Sim<sub>time</sub></code> parameters
Mining power distribution of miners (mining pools)	Mining pools mining power distribution (see Section IV-A3); Set by updating variable <i>bitcoinMinersHash</i> in <code>bitcoin-miner.cc</code>	Gaussian hash power distribution [66]	Mining pools mining power distribution; Set using <code>HashPower</code> parameter

## A. COMPARISON CRITERIA

In this section, we comment on the comparison criteria used for the evaluation of blockchain simulators.

### 1) BLOCK PROPAGATION TIME (DELAY)

The *average block propagation time* is the average time needed for a new block to reach most network nodes. Block propagation time is one of the critical factors that impact a blockchain system's scalability and affect the stale block rate. The Bitcoin network monitoring website [78] confirms that block propagation delay has decreased significantly in recent

years. For example, since 2015, the median block propagation delay (the amount of time until 50% of peers have processed a block) has decreased from more than six seconds to less than one. Moreover, the block propagation delay for the 90th percentile has decreased from more than 15 seconds in 2015 to around three seconds in 2020. There are several principal reasons behind this development [67]: i) the emergence of relay networks [79], such as FIBRE [80] and Falcon [81], capable of transmitting blocks at very high transmission rates; ii) extensions of Bitcoin protocol, e.g., the introduction of the compact block relay [82], which enables faster propagation of



**TABLE 8.** Comparison of the average block size  $s_B$  (in kilobytes, KB), the 50th and 90th percentile of the block propagation delay  $t_{BPD}$  (in seconds), and the stale block rate  $r_s$  (in percents) in the Bitcoin network and using simulators with different input data corresponding to Bitcoin's network situation in a specific Year.

Simulator	Year	$s_B$	$t_{BPD}^{50th}$	$t_{BPD}^{90th}$	$r_s$
Bitcoin (measured)	2016	534.8	8.7 [44]	17.3 [78]	0.41 [44]
	2019	1,010.1	0.4	2.4	N/R
	2020	1,219.8	0.5 [78]	3.3 [78]	0.06 [84]
Bitcoin-Simulator	2016 [44]	N/R <sup>f</sup>	9.42	N/R	0.14
	2016 <sup>a</sup>	526.1	9.12	17.68	0.13
	2020	1,224.2	26.94	53.25	0.08
SimBlock	2015 [51]	534 <sup>d</sup>	8.94	N/R	0.58
	2019 [66]	1,000 <sup>d</sup>	1.3	2.4	N/R
	2019 <sup>b</sup>	1,000 <sup>d</sup>	1.4	2.1	0.2
	2020	1,220 <sup>d</sup>	1.3	2.1	0.19
BlockSim:	2016 [85]	534 <sup>d</sup>	14.7 <sup>e</sup>	-	1.69
Alharby	2016 <sup>c</sup>	534 <sup>d</sup>	14.7 <sup>e</sup>	-	1.61
	2020	1,220 <sup>d</sup>	0.5 <sup>e</sup>	-	0.07

<sup>a</sup> Repeated simulation using the same conditions as in [44].

<sup>b</sup> Repeated simulation using the same conditions as in [66].

<sup>c</sup> Repeated simulation using the same conditions as in [85].

<sup>d</sup> Fixed block size was set as the input parameter.

<sup>e</sup> Fixed block propagation delay was set as the input parameter.

<sup>f</sup> N/R - Not Reported.

blocks by sending only transaction IDs instead of all transactions contained in a block; iii) performance improvements in client implementations, e.g., the use of hardware optimization for SHA256 hashing; and iv) significant improvement of network parameters such as bandwidth and latency.

## 2) STALE BLOCK RATE

*Stale blocks* refer to mined blocks that are not included in the longest chain due to, e.g., concurrency, conflicts, or network propagation delays. Note that in Ethereum, uncle blocks correspond to stale blocks with parents that are ancestors (max. 6 blocks back) of the included block [83]. Unfortunately, a well-known website<sup>5</sup> provides inaccurate information regarding the current *stale block rate* ( $r_s$ ). Therefore, the current stale block rate was measured based on the information provided in [84], where a total of 10 stale blocks were reported throughout more than three months (i. e., out of around 16,000 blocks; see Table 8).

<sup>5</sup><https://www.blockchain.com/charts/n-orphaned-blocks>

**TABLE 9.** Summary of available input parameters for Bitcoin-Simulator.

Parameter	Description	Default value
blockSize	The fixed block size (Bytes). If the default value is used then the blockSize follows the real bitcoin block size distribution as estimated by collecting stats from <a href="https://blockchain.info">https://blockchain.info</a>	-1
noBlocks	The number of generated blocks	100
nodes	The total number of nodes in the network. The number of nodes should always be greater or equal to the number of miners. The number of miners, their hash rates and their locations can be changed by modifying the variables <code>bitcoinMinersHash/litecoinMinersHash/dogecoinMinersHash</code> and <code>bitcoinMinersRegions/litecoinMinersRegions/dogecoinMinersRegions</code>	16
minConnections	The <code>minConnectionsPerNode</code> of the grid	-1
maxConnections	The <code>maxConnectionsPerNode</code> of the grid. If <code>minConnections &lt;= 0</code> or <code>maxConnections &lt;= 0</code> , the nodes follow the connections distribution as described in [75]	-1
blockIntervalMinutes	The average block generation interval in minutes	10
invTimeoutMins	The inv block timeout. If the default value is used, the timeouts are twice as long as <code>blockIntervalMinutes</code>	-1
unsolicited	Change the miners block broadcast type to UNSOLICITED. Each newly mined block is broadcast immediately to all the peers of the miner who mined it	false
relayNetwork	Change the miners block broadcast type to RELAY_NETWORK. The miners use a relay network for communicating among themselves and send compressed blocks	false
unsolicitedRelayNetwork	Change the miners block broadcast type to UNSOLICITED_RELAY_NETWORK. The miners use a relay network among themselves and send compressed blocks and each newly mined block is broadcast immediately to all the peers of the miner who mined it	false
sendheaders	Change the protocol to sendheaders	false
litecoin	Imitate the litecoin network behaviour. It sets the <code>-nodes=1000</code> , <code>-blockIntervalMinutes=2.5</code> , follows the same connection distribution as bitcoin, but the litecoin distribution for the block generation interval and block size	false
dogecoin	Imitate the dogecoin network behaviour. It sets the <code>-nodes=650</code> , <code>-blockIntervalMinutes=1</code> , follows the same connection distribution as bitcoin, but the dogecoin distribution for the block generation interval and block size	false
blockTorrent	Enable the BlockTorrent protocol	false
chunkSize	The chunksize of the blockTorrent in Bytes. Used only in conjunction with <code>-blockTorrent</code>	-1
spv	Enable the spv mechanism in blockTorrent. Used only in conjunction with <code>-blockTorrent</code> . The nodes are able to advertise chunks of blocks which are not yet validated	false

**TABLE 10.** Summary of available input parameters for SimBlock.

Parameter	Description
REGION_LIST	List of regions, where nodes can exist. Available regions are North America, South America, Europe, Asia and Pacific, Japan, Australia
LATENCY	List of latency assigned to each region (unit: millisecond)
UPLOAD_BANDWIDTH	List of upload bandwidth assigned to each region (unit: bit per second)
DOWNLOAD_BANDWIDTH	List of download bandwidth assigned to each region (unit: bit per second)
REGION_DISTRIBUTION	The distribution of node's region. Each value means the rate of the number of nodes in the corresponding region to the number of all nodes
DEGREE_DISTRIBUTION	The cumulative distribution of number of outbound links according to [75]
NUM_OF_NODES	The number of nodes participating in the blockchain network
TABLE	The kind of routing tables
INTERVAL	The expected value of block generation interval. The difficulty of mining is automatically adjusted by this value and the sum of mining power (unit: millisecond)
AVERAGE_MINING_POWER	The average mining power of each node. Mining power corresponds to Hash Rate in Bitcoin, and is the number of mining (hash calculation) executed per millisecond
STDEV_OF_MINING_POWER	The mining power of each node is determined randomly according to the normal distribution whose average is AVERAGE_MINING_POWER and standard deviation is STDEV_OF_MINING_POWER
END_BLOCK_HEIGHT	The block height when a simulation ends
BLOCK_SIZE	Block size (unit: byte)
CBR_USAGE_RATE	The usage rate of compact block relay (CBR) protocol
CBR_FAILURE_RATE_FOR_CONTROL_NODE	CBR failure rate for a node that always connect network
CBR_FAILURE_RATE_FOR_CHURN_NODE	CBR failure rate for a node that causes churn
CBR_FAILURE_BLOCK_SIZE_DISTRIBUTION_FOR_CONTROL_NODE	The distribution of data size that a control node receives when fails CBR
CBR_FAILURE_BLOCK_SIZE_DISTRIBUTION_FOR_CHURN_NODE	The distribution of data size that a churn node receives when fails CBR

### 3) AVERAGE BLOCK SIZE

Low transaction throughput is one of the key challenges for Bitcoin and other PoW-based blockchains. Transaction throughput attained by the system is practically limited by a parameter known as the *maximum block size*. This parameter indirectly defines the maximum number of on-chain transactions carried within a single block. Over the history of Bitcoin, various solutions to increase or obliterate the maximum block size have been proposed. Authors in [44] showed (using *Bitcoin-Simulator*) that simply shortening the block generation interval or increasing the maximum block size does not solve the low transaction throughput problem. Both such strategies improve the blockchain throughput. However, larger blocks incur slower propagation, while shorter block generation intervals cause inconsistencies of the system, increasing the stale block rate and weakening the blockchain system's security [44], [66].

The following section compares the current block propagation delay and the stale block rate in the existing Bitcoin network with the simulated values obtained using the selected simulators. In this way, we seek to investigate whether the

existing simulators follow the progress of and manage to realistically simulate the present Bitcoin network.

### B. SIMULATION RESULTS AND DISCUSSION

To experimentally validate the selected simulators, we compared the Bitcoin network in 2016, 2019, and 2020 with their respective simulated counterparts. Specifically, we compared the *average block size*  $s_B$  (measured in kilobytes, KB), the 50th and 90th percentiles of the *block propagation delay*  $t_{BPD}$  (measured in seconds), and the *stale block rate*  $r_s$  (measured in percents) in the actual Bitcoin network with the same parameter values obtained using the simulators. Each simulator was run 10 times, generating 10,000 blocks each time, and all experiments were carried out on a computer with a 64-bit AMD Ryzen 5 2600X Six-Core @3.6 GHz Processor, 16 GB RAM, and running Ubuntu Linux OS.

The results of the simulations using the three tested simulators and the values measured in the real Bitcoin network are summarized in Table 8. First, we note that using the same conditions, we were able to reproduce results very close to those reported in the literature, i.e., for

**TABLE 11.** Summary of available input parameters for BlockSim:Alharby.

Parameter	Description
$B_{interval}$	Average time to generate a block (unit: seconds)
$B_{size}$	Block size (unit: MegaBytes)
$B_{limit}$	Block gas limit (in Ethereum) (unit: gas)
$B_{delay}$	Average propagation delay of a block (unit: seconds)
$B_{reward}$	Block generation reward
hasTrans	Enable/disable transactions (unit: boolean)
Ttechnique	Technique for modelling transactions (Full/Light)
$T_n$	The rate of the number of transactions to be created per second
$T_{delay}$	Average propagation delay of a transaction (only if Full technique is used)
$T_{fee}$	Transaction fee (calculated based on probability distribution)
$T_{size}$	Transaction size (calculated based on probability distribution) (unit: Megabytes)
hasUncles	Enable/Disabled uncle inclusion mechanism (in Ethereum) (unit: boolean)
$B_{uncles}$	Maximum number of uncles allowed per block
$U_{generations}$	The number of generations (depth) in which the uncle can be included
$U_{reward}$	Uncle generation reward
$U_{ireward}$	Uncle inclusion reward
$N_n$	The total number of nodes in the network
HashPower	The hashpower is defined for every node individually as a percentage of all hashpower that assumed to be 100%
$Sim_{time}$	The length of simulated period (units: seconds)
Runs	The number of simulation runs

<sup>8</sup><https://github.com/maher243/BlockSim/blob/master/InputsConfig.py>

Bitcoin-Simulator as reported in [44] (see the fifth and sixth rows in Table 8), for SimBlock as reported in [66] (see the ninth and tenth rows in Table 8), and for BlockSim as reported in [85] (see the twelfth and thirteenth rows in Table 8).

Next, we observe that all simulators were able to simulate the Bitcoin network in 2016 very accurately. Unfortunately, one of the most promising and functional simulators (see Table 3), Bitcoin-Simulator is currently outdated and cannot accurately simulate the current Bitcoin network (see simulation results and measured values for 2020). The recent extensions to the Bitcoin protocol, such as the compact block relay [82], have dramatically decreased the stale block rate and the block propagation delay. This can be observed in simulation results obtained using SimBlock, which supports the compact block relay [66]. We can state that SimBlock managed to reproduce the Bitcoin network of 2019 and 2020 at an acceptable level – the simulations resulted in higher  $r_{BPD}^{50th}$  and  $r_s$  values than the measured ones, but lower  $r_{BPD}^{90th}$  values. However, without simulating transactions propagation, it is not entirely clear how accurate and realistic the simulation process is.

In contrast, BlockSim can simulate transactions, but its simulation of the network layer is greatly simplified.

This simulator cannot simulate the block propagation delay and requires fixed values as the input parameters. Note that the simulated block rate for 2016 is higher than the real one. However, for 2020, a very close value was obtained even with simplified network layer implementation. Therefore, again, it is not entirely clear how realistic the simulation process is.

## VI. CONCLUSION AND POSSIBLE FUTURE RESEARCH DIRECTIONS

This paper conducted a systematic review and a detailed comparative analysis of the state-of-the-art blockchain simulators as well as provided the information needed for selecting the best available simulator for specific simulation purposes. We identified that distributed ledger simulation is still in the early stages and the existing simulators are limited to specific abstraction levels. The most advanced existing simulators are intended to simulate various PoW-based blockchains. Unfortunately, most of them are not actively developed and remain outdated. This is one of the main reasons why new blockchain projects have not actively used simulators to advance initial deployment.

Our experimental validation of the selected PoW simulators revealed that there is no “one-size-fits-all” simulator that

could accurately simulate all layers of a PoW blockchain. Further improvement of existing and the design of new blockchain simulators seem a vital research direction in the DLT field. Moreover, outdated simulators could no longer simulate the current actively developing blockchain networks.

One possible future research direction could be a comparative analysis of the simulators designed to study specific aspects of a blockchain system (i.e., security) or an investigation of blockchain simulators that support layers not considered in this study. Another research avenue could be the development of simulators or their extensions that support modern consensus algorithms. Our analysis revealed a need for simulators supporting other types of consensus algorithms, such as PoS or various BFT variations. Finally, continuous development and improvement of existing simulators are needed to accurately simulate the most recent blockchain networks.

## APPENDIX. CONFIGURATION OPTIONS FOR SELECTED BLOCKCHAIN SIMULATORS

### A. BITCOIN-SIMULATOR

In Table 9 we list the configuration parameters<sup>6</sup> along with their default values available in the latest version of Bitcoin-Simulator [72]. Configuration options should be specified via command line arguments.

### B. SIMBLOCK

In Table 10 we list the configuration parameters<sup>7</sup> defined in the latest version of SimBlock [73]. Configuration options can be specified in classes *NetworkConfiguration.java* and *SimulationConfiguration.java*, where default values are already specified for simulating Bitcoin, Litecoin, and Dogecoin blockchain networks.

### C. BLOCKSIM: ALHARBY CONFIGURATION OPTIONS

In Table 11 we list the configuration parameters<sup>8</sup> defined in the latest version of BlockSim:Alharby [50]. Configuration options can be specified in file *NInputsConfig.py* and where default values are already specified for simulating Bitcoin and Ethereum blockchain networks.

## REFERENCES

- [1] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: <https://bitcoin.org/bitcoin.pdf>
- [2] M. Belotti, N. Bozic, G. Pujolle, and S. Secci, "A vademecum on blockchain technologies: When, which, and how," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3796–3838, 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8760539/>
- [3] D. Di Francesco Maesa and P. Mori, "Blockchain 3.0 applications survey," *J. Parallel Distrib. Comput.*, vol. 138, pp. 99–114, Apr. 2020.
- [4] R. Paulavičius, S. Grigaitis, A. Igumenov, and E. Filatovas, "A decade of blockchain: Review of the current status, challenges, and future directions," *Informatica*, vol. 30, no. 4, pp. 729–748, Jan. 2019.
- [5] U. Bodkhe, S. Tanwar, K. Parekh, P. Khanpara, S. Tyagi, N. Kumar, and M. Alazab, "Blockchain for industry 4.0: A comprehensive review," *IEEE Access*, vol. 8, pp. 79764–79800, 2020.
- [6] L. Lao, Z. Li, S. Hou, B. Xiao, S. Guo, and Y. Yang, "A survey of IoT applications in blockchain systems: Architecture, consensus, and traffic modeling," *ACM Comput. Surv.*, vol. 53, no. 1, pp. 1–32, May 2020.
- [7] M. Pournader, Y. Shi, S. Seuring, and S. C. L. Koh, "Blockchain applications in supply chains, transport and logistics: A systematic review of the literature," *Int. J. Prod. Res.*, vol. 58, no. 7, pp. 2063–2081, Apr. 2020.
- [8] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, pp. 16440–16455, 2020.
- [9] C. Fan, S. Ghaemi, H. Khazaei, and P. Musilek, "Performance evaluation of blockchain systems: A systematic survey," *IEEE Access*, vol. 8, pp. 126927–126950, 2020.
- [10] S. Smetanin, A. Ometov, M. Komarov, P. Masek, and Y. Koucheryayv, "Blockchain evaluation approaches: State-of-the-art and future perspective," *Sensors*, vol. 20, no. 12, pp. 1–20, 2020.
- [11] Q. Zhu, S. W. Loke, R. Trujillo-Rasua, F. Jiang, and Y. Xiang, "Applications of distributed ledger technologies to the Internet of Things," *ACM Comput. Surv.*, vol. 52, no. 6, pp. 1–34, Jan. 2020.
- [12] *Bitnodes: Global Bitcoin Nodes Distribution*. Accessed: Oct. 10, 2020. [Online]. Available: <https://bitnodes.io/>
- [13] C. Faria and M. Correia, "BlockSim: Blockchain simulator," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Jul. 2019, pp. 439–446.
- [14] J. Banks, J. S. Carson, II, B. L. Nelson, and D. M. Nicol, *Discrete-Event System Simulation*, 5th ed. London, U.K.: Pearson, 2014.
- [15] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.
- [16] S. Popov. (2018). *The Tangle*. Accessed: Oct. 6 2020. [Online]. Available: <https://bit.ly/3d8jHAV>
- [17] C. LeMahieu. (2018). *Nano: A Feeless Distributed Cryptocurrency Network*. Accessed: Oct. 6, 2020. [Online]. Available: <https://nano.org/en/whitepaper>
- [18] R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn, "Corda: An introduction," *R3 CEV*, vol. 1, p. 15, Aug. 2016.
- [19] *Radix DeFi White Paper*, Thomson Reuters Found., London, U.K., 2020, pp. 1–31. Accessed: Oct. 6, 2020.
- [20] EOS. (2017). *EOS.IO Technical White Paper*. Accessed: Oct. 6, 2020. [Online]. Available: <https://github.com/EOSIO/Documentation>
- [21] D. Schwartz, N. Youngs, and A. Britto. (2014). *The Ripple Protocol Consensus Algorithm*. Accessed: Oct. 6, 2020. [Online]. Available: <https://cryptoguide.ch/cryptocurrency/ripple/whitepaper.pdf>
- [22] E. Harris-Braun, N. Luck, and A. Brock. (2018). *Holochain: Scalable Agent-Centric Distributed Computing*. Accessed: Oct. 6, 2020. [Online]. Available: <https://github.com/holochain/holochain-proto/blob/whitepaper/holochain.pdf>
- [23] (2018). *Blockchain for Decentralized Workflows*. Accessed: Oct. 6, 2020. [Online]. Available: <https://tonetwork.com/documents/LTO%20Network%20-%20Technical%20Paper.pdf>
- [24] (2015). *Hyperledger Project*. Accessed: Jun. 4, 2019. [Online]. Available: <https://www.hyperledger.org>
- [25] *Corda Platform*. Accessed: Oct. 10, 2020. [Online]. Available: <https://www.r3.com/corda-platform/>
- [26] F. R. Batubara, J. Ubacht, and M. Janssen, "Unraveling transparency and accountability in blockchain," in *Proc. 20th Annu. Int. Conf. Digit. Government Res.*, Jun. 2019, pp. 204–213.
- [27] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "BLOCKBENCH: A framework for analyzing private blockchains," in *Proc. ACM Int. Conf. Manage. Data*, May 2017, pp. 1085–1100.
- [28] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 1992, pp. 139–147.
- [29] S. King and S. Nadal. (2012). *PPCoin: Peer-to-Peer Crypto-Currency With Proof-of-Stake*. [Online]. Available: <https://decred.org/research/king2012.pdf>
- [30] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. 3rd Symp. Oper. Syst. Design Implement.*, 1999, pp. 173–186.
- [31] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1432–1465, 2nd Quart., 2020.
- [32] Q. Wang, J. Yu, S. Chen, and Y. Xiang, "SoK: Diving into DAG-based blockchain systems," 2020, *arXiv:2012.06128*. [Online]. Available: <http://arxiv.org/abs/2012.06128>

<sup>6</sup>[http://arthurgervais.github.io/Bitcoin-Simulator/use\\_it.html](http://arthurgervais.github.io/Bitcoin-Simulator/use_it.html)

<sup>7</sup><https://github.com/dsg-titech/simblock/blob/master/docs/en/usage.md>

<sup>8</sup><https://github.com/maher243/BlockSim/blob/master/InputsConfig.py>



- [33] L. M. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *Proc. 41st Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May 2018, pp. 1545–1550.
- [34] S. M. H. Bamakan, A. Motavali, and A. B. Bondarti, "A survey of blockchain consensus algorithms performance evaluation criteria," *Expert Syst. Appl.*, vol. 154, Sep. 2020, Art. no. 113385.
- [35] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz, "Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities," *IEEE Access*, vol. 7, pp. 85727–85745, 2019.
- [36] R. C. Merkle, "Protocols for public key cryptosystems," in *Proc. IEEE Symp. Secur. Privacy*, Apr. 1980, p. 122.
- [37] N. Szabo. (1994). *Smart Contracts*. Accessed: Dec. 20, 2020. [Online]. Available: <https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html>
- [38] D. Moher, A. Liberati, J. Tetzlaff, and D. G. Altman, "Preferred reporting items for systematic reviews and meta-analyses: The prisma statement," *PLoS Med.*, vol. 6, no. 7, 2009, Art. no. e1000097.
- [39] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *Financial Cryptography and Data Security (Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7859. Berlin, Germany, 2013, pp. 34–51. [Online]. Available: [http://link.springer.com/10.1007/978-3-642-39884-1\\_4](http://link.springer.com/10.1007/978-3-642-39884-1_4)
- [40] R. Brune. (2013). *Bitcoin Network Simulator*. Accessed: Dec. 6, 2020. [Online]. Available: <https://github.com/rbrune/btcsim/>
- [41] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Financial Cryptography and Data Security*, N. Christin and R. Safavi-Naini, Eds. Berlin, Germany, 2014, pp. 436–454.
- [42] S. Bove. (2013). *Javascript P2P Network Simulator*. Accessed: Dec. 6, 2020. [Online]. Available: <https://github.com/rbrune/btcsim/>
- [43] A. Miller and R. Jansen, "Shadow-bitcoin: Scalable simulation via direct execution of multi-threaded applications," in *Proc. 8th Workshop Cyber Secur. Exp. Test (CSET)*, 2015, pp. 1–8.
- [44] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 3–16.
- [45] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan, "On the instability of bitcoin without the block reward," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 154–167.
- [46] L. Stoykov, K. Zhang, and H.-A. Jacobsen, "VIBES: Fast blockchain simulations for large-scale peer-to-peer networks," in *Proc. 18th ACM/IFIP/USENIX Middleware Conf., Posters Demos*, Dec. 2017, pp. 19–20.
- [47] A. Deshpande, P. Nasirifard, and H.-A. Jacobsen, "EVIBES: Configurable and interactive ethereum blockchain simulation framework," in *Proc. 19th Int. Middleware Conf. (Middleware)*, Dec. 2018, pp. 11–12. [Online]. Available: <http://dl.acm.org/citation.cfm?doi=3284014.3284020>
- [48] M. Conoscenti, A. Vetrò, J. De Martin, and F. Spini, "The CLoTH simulator for HTLC payment networks with introductory lightning network performance results," *Information*, vol. 9, no. 9, p. 223, Sep. 2018.
- [49] W. Hao, J. Zeng, X. Dai, J. Xiao, Q. Hua, H. Chen, K.-C. Li, and H. Jin, "BlockP2P: Enabling fast blockchain broadcast with scalable peer-to-peer network topology," in *Proc. Int. Conf. Green, Pervasive, Cloud Comput.* Cham, Switzerland: Springer, 2019, pp. 223–237.
- [50] M. Alharby and A. van Moorsel, "Blocksim: A simulation framework for blockchain systems," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 46, no. 3, pp. 135–138, 2019.
- [51] Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo, "SimBlock: A blockchain network simulator," in *Proc. IEEE INFOCOM, Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 325–329.
- [52] E. Rosa, G. D'Angelo, and S. Ferretti, "Agent-based simulation of blockchains," in *Methods and Applications for Modeling and Simulation of Complex Systems*, vol. 1094, G. Tan, A. Lehmann, Y. M. Teo, and W. Cai, Eds. Singapore: Springer, 2019, pp. 115–126.
- [53] A. R. Sai, J. Buckley, and A. L. Gear, "Assessing the security implication of bitcoin exchange rates," *Comput. Secur.*, vol. 86, pp. 206–222, Sep. 2019.
- [54] M. Conti, A. Gangwal, and M. Toderò, "Blockchain trilemma solver algorithm has dilemma over undecidable messages," in *Proc. 14th Int. Conf. Availability, Rel. Secur.*, Aug. 2019, pp. 1–8.
- [55] H. Azimy and A. Ghorbani, "Competitive selfish mining," in *Proc. 17th Int. Conf. Privacy, Secur. Trust (PST)*, Aug. 2019, pp. 1–8.
- [56] M. Zander, T. Waite, and D. Harz, "DAGsim: Simulation of DAG-based distributed ledger protocols," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 46, no. 3, pp. 118–121, Jan. 2019.
- [57] F. Bruschi, V. Rana, L. Gentile, and D. Sciuto, "Mine with it or sell it: The superhashing power dilemma," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 46, no. 3, pp. 127–130, Jan. 2019.
- [58] M. Krol, A. Sonnino, M. Al-Bassam, A. Tasiopoulos, and I. Psaras, "Proof-of-prestige: A useful work reward system for unverifiable tasks," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2019, pp. 293–301.
- [59] P. Foytik, S. Shetty, S. P. Gochhayat, E. Herath, D. Tosh, and L. Njilla, "A blockchain simulator for evaluating consensus algorithms in diverse networking environments," in *Proc. Spring Simulation Conf. (SpringSim)*, May 2020, pp. 1–12.
- [60] L. Alsahan, N. Lasla, and M. Abdallah, "Local bitcoin network simulator for performance evaluation using lightweight virtualization," in *Proc. IEEE Int. Conf. Inform., IoT, Enabling Technol. (ICIOT)*, Feb. 2020, pp. 293–301.
- [61] S. M. Fattahi, A. Makanju, and A. M. Fard, "SIMBA: An efficient simulator for blockchain applications," in *Proc. 50th Annu. IEEE-IFIP Int. Conf. Dependable Syst. Netw.-Suppl. Volume (DSN-S)*, Jun. 2020, pp. 51–52.
- [62] M. Alharby, R. C. Lunardi, A. Aldweesh, and A. van Moorsel, "Data-driven model-based analysis of the ethereum verifier's dilemma," in *Proc. 50th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2020, pp. 209–220, doi: 10.1109/dsn48063.2020.00038.
- [63] Z. H. Chin, T. T. V. Yap, and I. K. T. Tan, "Simulating difficulty adjustment in blockchain with SimBlock," in *Proc. 2nd ACM Int. Symp. Blockchain Secure Crit. Infrastruct.*, Oct. 2020, pp. 192–197.
- [64] F. Zhao, X. Guo, and W. K. V. Chan, "Individual green certificates on blockchain: A simulation approach," *Sustainability*, vol. 12, no. 9, p. 3942, May 2020.
- [65] *The Ethereum Network Node Explorer*. Accessed: Dec. 10, 2020. [Online]. Available: <https://www.ethernodes.org/>
- [66] R. Nagayama, R. Banno, and K. Shudo, "Identifying impacts of protocol and Internet development on the bitcoin network," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2020, pp. 1–6.
- [67] T. Neudecker. (2019). *Characterization of the Bitcoin Peer-to-Peer Network (2015-2018)*. [Online]. Available: [http://dsn.tm.kit.edu/bitcoin/publications/bitcoin\\_network\\_characterization.pdf](http://dsn.tm.kit.edu/bitcoin/publications/bitcoin_network_characterization.pdf)
- [68] P. Todd. *Python-Bitcoinlib, Version 0.11.0*. Accessed: Nov. 30, 2020. [Online]. Available: <https://pypi.org/project/python-bitcoinlib/>
- [69] E. Lombrozo, J. Lau, and P. Wuille. (Aug. 24, 2017). *Bip: 141—Segregated Witness (Consensus Layer)*. [Online]. Available: <https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki>
- [70] J. Hilliard. *Bip: 91—Reduced Threshold Segwit MASF, Segregated Witness Upgrade Activated at Block 477,120*. Accessed: Dec. 18, 2020. [Online]. Available: <https://github.com/bitcoin/bips/blob/master/bip-0091.mediawiki>
- [71] *Verizon: IP Latency Statistics*. Accessed: Oct. 10, 2020. [Online]. Available: <https://enterprise.verizon.com/terms/latency/>
- [72] A. Gervais and V. Glykantzis. (2016). *Bitcoin and Blockchain Simulator*. [Online]. Available: <https://github.com/arthurgervais/Bitcoin-Simulator>
- [73] Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo. (2019). *An Open Source Blockchain Network Simulator*. [Online]. Available: <https://github.com/dsg-titech/simblock>
- [74] *Cambridge Bitcoin Mining Map*. Accessed: Oct. 20, 2020. [Online]. Available: [https://cbeci.org/mining\\_map](https://cbeci.org/mining_map)
- [75] A. Miller, J. Litton, A. Pachulski, N. Gupta, N. Spring, B. Bhattacharjee, and D. Levin. (2015). *Discovering Bitcoin's Public Topology and Influential Nodes*. Accessed: Dec. 6, 2020. [Online]. Available: <https://www.cs.umd.edu/projects/coinscope/coinscope.pdf>
- [76] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2012, pp. 906–917.
- [77] *Bitcoin Pool Distribution*. Accessed: Nov. 10, 2020. [Online]. Available: <https://btc.com/stats/pool>
- [78] *Bitcoin Network Monitor—DSN Research Group*. Accessed: Dec. 12, 2020. [Online]. Available: <https://dsn.tm.kit.edu/bitcoin/>
- [79] K. Otsuki, Y. Aoki, R. Banno, and K. Shudo, "Effects of a simple relay network on the bitcoin network," in *Proc. Asian Internet Eng. Conf. (AINTEC)*, 2019, pp. 41–46. [Online]. Available: <http://dl.acm.org/citation.cfm?doi=3340422.3343640>
- [80] *FIBRE Fast Internet Bitcoin Relay Engine*. Accessed: Dec. 12, 2020. [Online]. Available: <https://www.bitcoinfibre.org/>



- [81] *Falcon—A Fast Bitcoin Backbone*. Accessed: Dec. 12, 2020. [Online]. Available: <https://www.falcon-net.org/>
- [82] M. Corallo. (2016). *BIP: 152—Compact Block Relay*. Accessed: Nov. 30, 2020. [Online]. Available: <https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki>
- [83] *Ethereum Wiki—Mining*. Accessed: Dec. 12, 2020. [Online]. Available: <https://eth.wiki/en/fundamentals/mining>
- [84] *Fork Monitor—BTC Stale Block Candidates*. Accessed: Dec. 12, 2020. [Online]. Available: <https://forkmonitor.info/notifications>
- [85] M. Alharby and A. van Moorsel, “BlockSim: An extensible simulation tool for blockchain systems,” *Frontiers Blockchain*, vol. 3, pp. 1–16, Jun. 2020.



**REMIGIJUS PAULAVIČIUS** received the Ph.D. degree in computer science from Vytautas Magnus University, Kaunas, Lithuania, in 2010. He was a Postdoctoral Researcher with Vilnius University, Vilnius, Lithuania, and a Research Associate with Imperial College London, London, U.K. He is currently a Professor and the Head of the Blockchain Technologies Group, Institute of Data Science and Digital Technologies, Vilnius University. His research interests include distributed ledger technologies, parallel and distributed computing, machine learning, and the development and application of various operation research techniques.



**SAULIUS GRIGAITIS** is currently pursuing the Ph.D. degree in informatics with Vilnius University, Lithuania. He also holds a Partnership Associate Professor position with Vilnius University. In addition to academic activities, he has extensive industry experience in building large-scale systems and blockchain technologies. His research interests include modern consensus algorithms and privacy-preserving cryptography, such as zero-knowledge proofs and homomorphic encryption.



**ERNESTAS FILATOVAS** received the Ph.D. degree in informatics engineering from Vilnius University, Lithuania, in 2012. He was a Postdoctoral Researcher with Vilnius University, Vilnius, Lithuania, where he is currently a Senior Researcher and a Co-founder of the Blockchain Technologies Group, Institute of Data Science and Digital Technologies. His research interests include distributed ledger technologies, high-performance computing, machine learning, multiobjective and global optimization, and the development and application of various operation research techniques.

• • •