computation

MDPI

Article

PyIncentiveBC: A Python Module for Simulation of Incentivization Mechanism Implemented in Blockchain-Based Systems

Abdellah Ouaguid [1], Mohamed Hanine [2,*], Zouhair Chiba [3], Noreddine Abghour [3] and Mohammed Ouzzif [4]

[1] 2IACS Laboratory, ENSET of Mohammedia, Hassan II University of Casablanca, Casablanca 20000, Morocco; ouaguid@gmail.com
[2] LTI Laboratory, National School of Applied Sciences, Chouaib Doukkali University, El Jadida 24000, Morocco
[3] LIS Labs, FSAC, Hassan II University of Casablanca, Casablanca 20000, Morocco
[4] C3S Laboratory, ESTC, Hassan II University of Casablanca, Casablanca 20000, Morocco
* Correspondence: hanine.m@ucd.ac.ma

Abstract: The diversity of approaches for retaining participants in a Blockchain-based system complicates benchmarking. The majority of proposals for rewarding and penalizing participants in these systems are limited to their own set of data and scenarios, making it hard to compare their effectiveness. To overcome these challenges, we developed PyIncentiveBC, a free, open-source, and modular simulator designed to evaluate the reliability of any approach, incorporating a dynamic and proportionate incentivization mechanism proposed in our previous work. PyIncentiveBC aims to provide the scientific communities with an extensible software solution facilitating the benchmarking of existing approaches with new ones proposed by them.

Keywords: reward-penalty mechanism; incentivization mechanism; blockchain; consensus algorithm; distributed system; loyalty programs; open-source software; python

Citation: Ouaguid, A.; Hanine, M.; Chiba, Z.; Abghour, N.; Ouzzif, M. PyIncentiveBC: A Python Module for Simulation of Incentivization Mechanism Implemented in Blockchain-Based Systems. *Computation* **2024**, *12*, 179. https://doi.org/10.3390/computation12090179

Academic Editor: Filippo Palombi

Received: 11 July 2024
Revised: 21 August 2024
Accepted: 30 August 2024
Published: 3 September 2024

1. Introduction and Motivation

Blockchain is a secure technology based on a distributed architecture, enabling the exchange, verification, and storage of information among independent entities known as "nodes" without resorting to a central or intermediary authority [1]. The nodes within the Blockchain network are responsible for ensuring the exchange, validation, and storage of information, typically in the form of blocks that group several legal or illicit transactions [2]. This technology has brought profound changes across various sectors and domains such as Trust management [3], Finance [4,5], IoT [6,7], Healthcare [8,9], E-learning [10,11], Supply chain [12], Security analysis [2,13], Access Control Systems [14], Video integrity verification [15], Digital Marketing [16], Artificial Intelligence [17], among others. Its effectiveness lies in establishing auditability, integrity, and transparency among participants. The continuous operation of participating nodes ensures the exchange data's availability, integrity, and reliability, subsequently stored by diverse entities. These entities collectively reach a consensus on the data's integrity through the application of a consensus algorithm, the types and characteristics of which have been widely explored in various studies [18], illustrating their contributions and shortcomings.

Participating nodes actively contribute their resources to ensure the stability and security of a blockchain-based system. These resources may be financial, such as stakes [19], or material, including storage [20] and computing power [21]. Additionally, nodes consistently aim to comply with the blockchain system's predefined policies, conditions, and rules. This compliance is essential for the node to remain in good standing and avoid being "ejected" from the network. However, the efforts exerted by participants in respecting these rules are not consistently acknowledged or fairly rewarded, potentially leading to a negative impact on participant retention rates and, consequently, destabilizing the entire network.

In the literature, several reward and penalization approaches applied to participants in various systems have been proposed and implemented, such as in the field of commerce or computerized systems based on Blockchain [22] and others [23]. In a traditional and conventional system, a typical loyalty program initially involves participants joining these programs, followed by the allocation of cumulative rewards proportional to the purchases made by the program's owner [24]. Member infidelity or their lack of active commitment to the merchant or partners may jeopardize their unused loyalty points, which can be considered a kind of penalty triggered by inactivity over a predefined period.

In most Blockchain-based systems implementing Proof of Work (PoW) as a consensus algorithm, such as Bitcoin [21] and its variants, only the elected Leader participant is rewarded with newly minted tokens. The same Leader also accumulates all transaction fees associated with the transactions related to the created block. Consequently, in each mining round, a singular participant benefits twice from the generated rewards. In contrast, the effort of the other participants does not undergo any recognition or monetization with the system in which they cooperate. In addition, in certain Blockchains, the newly created native rewards decrease over time [25], meaning that elected Leader miners will only collect transaction fees as a reward, which will deter most participants from continuing mining and cooperating actively for the proper functioning of their systems.

The various reward and penalization systems currently in existence exhibit both advantages and shortcomings, some of which may potentially discourage participants from actively engaging in the network, consequently weakening the system to which they belong. With this consideration in mind, we have introduced a novel dynamic, proportionate, and enduring "Reward and Penalty" mechanism [26]. This mechanism aims to ensure rewards for the majority of participating nodes, thereby enhancing their satisfaction and reducing the Return On Investment (ROI) time for their contributions. The implementation of our proposal is anticipated to render the Blockchain ecosystem profitable, robust, and stable.

Despite the widespread application and examination of reward and penalization approaches in diverse contexts, we have identified a gap in the availability of a free and open-source simulator designed to assess the reliability of any proposed approach. In response, this paper introduces PyIncentiveBC, a Python-3-based tool designed to challenge our previously proposed "Reward and Penalty" mechanism [26].

The primary motivation behind the development of PyIncentiveBC stems from the limitations in benchmarking and evaluating the reliability of incentivization mechanisms in blockchain systems. Existing methods for retaining participants often lack comprehensive evaluation tools, leading to inefficiencies in assessing their effectiveness [26]. We aim to address this gap by providing a modular and open-source simulator that enables rigorous and comparative analysis of reward and penalty systems, ultimately enhancing participant retention and network stability. Licensed under the MIT License, PyIncentiveBC comes with a three-layer/module architecture and is compatible with Microsoft Windows 10 (or later), macOS 10.15 Catalina (or later), and most modern Linux distributions (kernel version 5.4 or later).

The key contributions of this work are:

- Development of PyIncentiveBC, a Python-based, open-source simulator designed to assess the reliability of incentivization mechanisms in blockchain systems.
- Implementation of a dynamic, proportionate reward and penalty mechanism that addresses the inefficiencies in current blockchain incentivization methods.
- Provision of a modular framework that allows for easy integration and extension by the research community to benchmark and evaluate various approaches.

In Section 2, we present related work to provide context and highlight the contributions of our approach. Section 3 presents a description of the reward and punishment system. Section 4 outlines the objectives of the developed tool and describes the implemented equations. Section 5 provides implementation details by describing the software architecture and its functionalities. Section 6 demonstrates the use of PyIncentiveBC with both synthetic and real-world data and parameters. An overview of the scope of our simu-

lator and its impacts is provided in Section 7. Finally, Section 8 serves as the conclusion of this paper.

2. Related Work

Various reward and penalty mechanisms have been proposed for blockchain-based systems to incentivize participation and maintain network security. Nakamoto [21] introduced Proof of Work (PoW), a consensus mechanism that rewards only the leader node that successfully solves the cryptographic puzzle. However, PoW has faced criticism for its energy inefficiency and its tendency to promote centralization. Subsequent innovations attempted to address these issues: King and Nadal [19] proposed Proof of Stake (PoS), linking reward probability to participants' stake, while Larimer [27] developed Delegated Proof of Stake (DPoS) to improve scalability. Researchers have continued to refine these approaches. GHOST protocol [28] was introduced to modify PoW reward calculations, while alternative mechanisms such as Proof of Burn [29], Proof of Reputation [30], and Proof of Activity [31] have emerged, each attempting to optimize different aspects of blockchain incentivization.

More recent approaches have addressed specific challenges in blockchain incentivization. Nguyen et al. [32] proposed a hierarchical incentive mechanism for federated learning in blockchain networks, while Matsunaga, Takeaki, et al. [33] introduced a mechanism for Proof of Stake (PoS) that enhances validator incentives by redistributing penalties from incorrect votes to accurate validators based on their voting history, improving vote accuracy and rewards for consistent validators.

Despite these advancements, there remains a lack of standardized evaluation methods for comparing these diverse incentivization mechanisms. While some researchers have attempted to develop evaluation frameworks [34–36], these efforts have been limited in scope, often focusing on specific aspects such as consensus efficiency or security metrics, without comprehensively addressing the full spectrum of incentivization strategies.

Our work addresses this gap by introducing PyIncentiveBC, a flexible, open-source simulator designed to accommodate and facilitate direct comparisons between various incentivization approaches. By integrating dynamic and proportionate reward and penalty mechanisms, our tool enables researchers to systematically benchmark and optimize incentivization strategies across diverse blockchain environments, contributing to the advancement of this critical aspect of blockchain technology.

3. Reward and Punishment System Description

Retaining participants by improving their satisfaction is the heart of our approach [26]. This is achieved through the implementation of a dynamic, proportionate, and automatic penalty and reward mechanism to eliminate any potential injustice that an active participant may experience. In certain instances, participants may find themselves forced to verify transactions and validate blocks, generated and propagated by other nodes, without subsequent recognition or reward.

Our approach offers a new revenue stream, ensuring that active participants receive due rewards. This involves setting up a gradual penalization system that applies to the nodes' sources of any abuse or non-compliance with the previously predefined rules of the Blockchain to which they belong. Nodes that comply with the protocol requirements of the Blockchain will have rewards (equal or proportional) guaranteed by our mechanism whether they are elected Leaders (via a consensus algorithm) or not.

The existing penalization systems in Blockchain-based systems do not facilitate the equitable distribution of penalties among honest nodes, such as sharing penalties collected by the system. In Blockchains based on the PoW consensus algorithm [21], any detected malicious nodes face the risk of being blacklisted and subsequently removed from the network. Despite this, these nodes can easily assume new identities and rejoin the network without encountering any restrictions. Conversely, in PoS systems, the implemented penalization system allows for a reduction in the stake of dishonest nodes; however, the recovered tokens are not proportionally redistributed to active participants in the network;

instead, these tokens are permanently removed from circulation (burned) [37] or allocated to a community or development fund [38]. In a general context, and for each mining round, nodes in a Blockchain-based system can be categorized based on their activity or role. Three main categories can be distinguished, as depicted in Figure 1: Nodes connected to the network but not participating in mining (NPN), honest participating nodes (PN), and malicious participating nodes (MN).

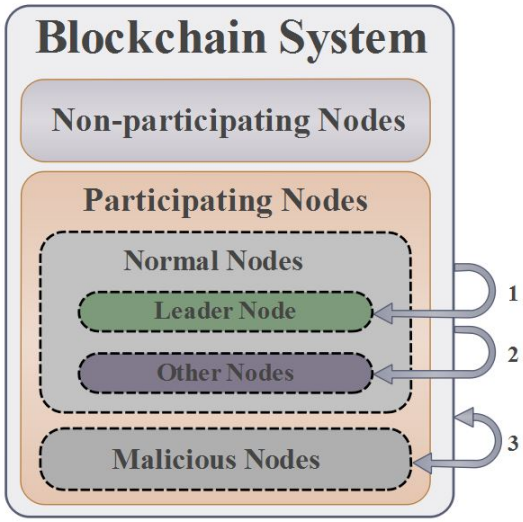


Figure 1. High-Level Architecture of a Blockchain-Based System.

The overall operation of our proposal is based on the fact that, for each block mining round, the system generates rewards that will be allocated proportionally to the Leader node (1) and to all nodes (2) that have participated in the same mining cycle and have validated the Leader's block. For a specific mining round, a portion of the rewards distributed to participants comes from the penalties imposed by the system on nodes deemed malicious (3). The enhanced cycle of rewards and penalties of our new mechanism is illustrated in Figure 2.

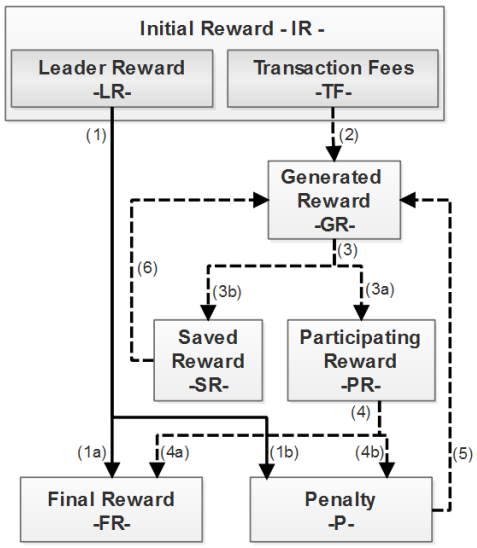


Figure 2. Cycle and Components of the Rewards and Penalties Mechanism.

Following our model illustrated in Figure 2, for each mining cycle, participants are remunerated through an Initial Reward (IR) provided by the system. This compensation consists of two parts:

- A portion newly created at the time of mining, called Leader Reward (LR). This is directly disbursed to the elected Leader node through a consensus algorithm.
- The other part, known as Transaction Fees (TF), is already encapsulated in transactions previously grouped in the created block and disseminated by the Leader node for future validation.

Transaction fees TF contribute to the Generated Reward (GR) (2), which is also fueled by:

- Penalties (P) recovered from malicious nodes during the preceding mining round (5).
- A portion of the GR not disbursed to participants (referred to as Saved Reward, or SR) but allocated for future nodes participating in upcoming mining cycles (6). The ultimate aim of this accumulation is to incentivize nodes to participate in future mining rounds and enable them to benefit, for as long as possible, from transaction fees and penalties collected during previous mining cycles.

The vast majority of the Generated Reward (GR) will be used as a reward for actively participating nodes in the mining process (3a), and the remainder will be re-injected into the GR of the next cycle (3b) to ensure that, regardless of the value of transaction fees or penalties collected, additional rewards will always be present in each mining cycle. Before the distribution of rewards to the concerned participants, the Participating Reward (PR) and Leader Reward (LR) may undergo treatments to subtract penalties (1b and 4b) when a violation of the Blockchain protocol requirements is identified. In the opposite case, participants will receive their entire reward without any deductions, thanks to their compliance with the protocol of the said Blockchain. Right after, the system will allocate the remaining rewards, named Final Reward (FR), to participants either integrally (if participants comply) or proportionally to the debited penalties. The calculation of various components (Node Score, Generated Reward -GR-, Saved Reward -SR-, Participating Reward -PR-, Penalty -P-, and Final Reward -FR-) of the proposed approach is performed through step-by-step formulas in our previous works [26], simplified in Appendix B, and summarized in the following section.

To provide a more robust benchmarking framework, we have integrated the incentivization mechanism used in Bitcoin's Proof of Work (PoW) consensus algorithm into PyIncentiveBC. In Section 6.2, we will present a comparative analysis using PyIncentiveBC to evaluate the Bitcoin blockchain's original incentivization system alongside our proposed approach.

4. Tool Setup and Underlying Equations

The objective of the developed tool is to enable the calculation and visualization of the contribution of the implemented approaches in terms of rewards and penalties generated during each round, for each participating node in a given simulation.

Demonstrating the reliability of these approaches by comparing them to each other through simplified tables accompanied by user-friendly graphs will provide a concise overview of the strengths and weaknesses of each implemented approach, as well as enable the incorporation of possible algorithmic improvements to ensure the stability of the Blockchain on which it will be deployed.

To achieve this goal, our simulator defaults to hosting two approaches. The main equations used in this tool are (summarized below) already explained in our prior research and are further elucidated in Appendixes A and B:

- Approach 1: Normal Approach
  In the 'normal' approach, only the leader node receives the final reward (FR), which includes the leader's remuneration (LR) and the total transaction fees (TF) for the current round (Formula (1)). Other participants, despite their contributions and incurred costs, do not receive any reward.

$$FR_{node} = \begin{cases} LR_{node} + TF & \text{, if node = Leader} \\ 0 & \text{, if node = Participant} \end{cases} \quad (1)$$

- Approach 2: Improved approach
  The calculation of the final reward (FR) adopted by the second approach (Formula (2)) applies to every node that participated in the mining and validation round. This approach allows the leader node to receive the rewards appropriate to it (LR), but after deducting any penalties (P) associated with the percentage of its non-conformity. In contrast to the normal approach, the improved approach also enables the other participating nodes to receive their rewards (PR), funds for which are collected from previous rounds. These rewards are also subject to deductions based on the degree of conformity of each node. The details of the various calculations are well-described in Appendix B.

$$FR_{node} = \begin{cases} LR_{node} - P_{node} & \text{, if node = Leader} \\ PR_{node} - P_{node} & \text{, if node = Participant} \end{cases} \quad (2)$$

5. Implementation and Usage

5.1. Software Architecture

The proposed program, which is called "PyIncentiveBC", is made up of three bricks: "implementation", "comparison", and "graph generation".

The "implementation" brick contains the algorithmic implementation of one or more reward and penalty approaches. It includes the different calculations necessary for the different approaches and provides the user with an output containing the final result of each approach. These results will be reorganized and then compared to each other in the "comparison" brick to help users visualize the contributions of previously implemented approaches. For our example, we have adapted the output of each approach to extract the rewards generated for each node according to the normal approach and the proposed approach discussed in our previous work [26]. The latter describes the steps to calculate the weight of the nodes which is based on a score assigned to them during the different rounds of the simulation. As for the "graph generation" brick, as its name suggests, it makes it possible to identify the best approach thanks to ergonomic graphs which will be generated subsequently according to the indicators that the end user wants to visualize. The separation of the functionalities makes it possible to provide the scientific community with an extensible solution that can be easily integrated into Jupyter notebooks as well as into other programs based on Python or others (Figure 3). It should be noted that the implementation of the various features of PyIncentiveBC described above is based on several external Python libraries namely: Pandas [39], Numpy [40], and Plotly [41].

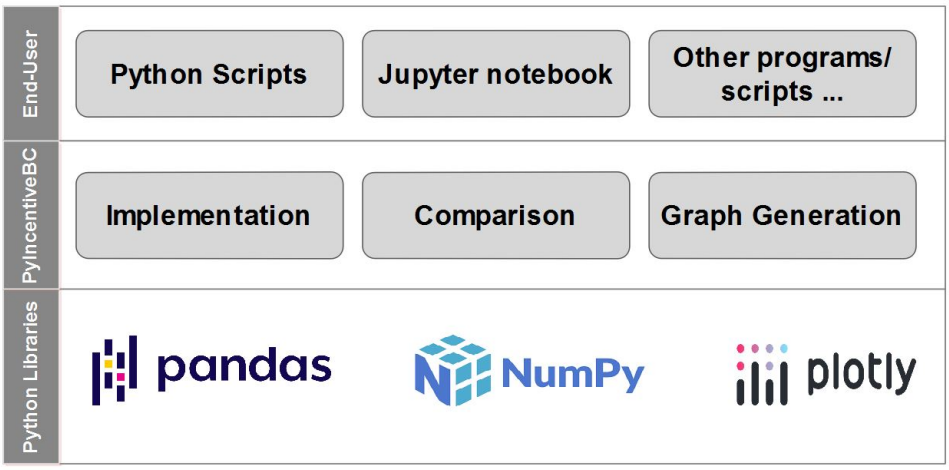


Figure 3. Software architecture of PyIncentiveBC.

5.2. Software Functionalities

To facilitate the use of the various features of PyIncentiveBC, the main program is in the form of a class named "PyIncentiveBCSystem" which can be imported for any other end-user program (another Python script or Jupyter notebook) by adding its call to the beginning of the program (Figure 4).

```
import pandas as pd
from PyIncentiveBCSystem import PyIncentiveBCSystem
```

Figure 4. Example of an import section inside a Jupyter notebook.

The "PyIncentiveBCSystem" class provides, through its attributes and methods, a complete implementation of different PyIncentiveBC bricks. Figure 5 presents an overview of the internal structure of the PyIncentiveBCSystem class.

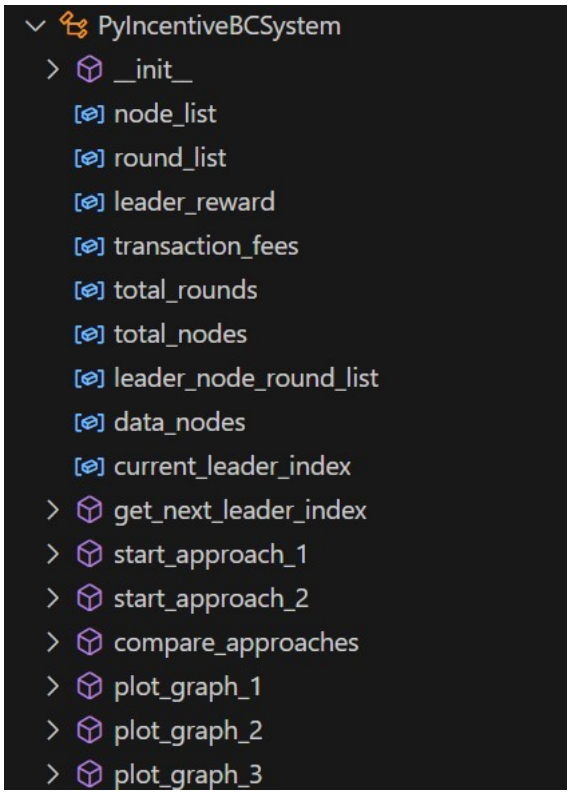


Figure 5. An overview of the PyIncentiveBCSystem internal structure.

The methods presented can be grouped into four categories:

- The first category concerns the initialization step (__init__()), during which the elementary attributes, namely the list of participating nodes, the list of simulation rounds, the rewards assigned to the nodes per round (e.g., leader reward, transaction fee)—are initialized.
- The second category includes the methods defining the consensus algorithms implemented to designate a Leader for each simulation round, particularly when the list of leader nodes is not predefined. In the first illustrative scenarios with synthetic exemplary data, we used a single consensus algorithm for the two implemented approaches. This consensus algorithm consists of electing nodes in turn.
- The functions containing the definitions of the implemented approaches belong to the third category of methods. Indeed, the methods "start_approach_1()", "start_approach_2()" contain the different processes necessary to implement the approach in question and

ultimately provide a two-dimensional data structure type output named DataFrame whose data can be easily manipulated via its various attributes and methods [42].

- The fourth method category ("compare_approaches()" and "plot_graph_X()") makes it possible to carry out the elementary processing operations ensuring the comparison of the results obtained by the implemented approaches, preparing and structuring the data necessary to allow the end user to visualize the results in different forms: data structure or graph.

For each approach implemented, our program offers default parameters that can be modified at any time by the user to allow them to customize the execution of their simulation and test the different scenarios according to their needs.

To increase the utility of PyIncentiveBC, we have extended its support for various blockchain consensus algorithms. While Section 6.2 demonstrates the tool's application with a real dataset from a Proof of Work (PoW) system, PyIncentiveBC is now capable of simulating and benchmarking incentivization mechanisms across a diverse range of consensus algorithms. This includes, but is not limited to, Proof of Stake (PoS), Delegated Proof of Stake (DPoS), Practical Byzantine Fault Tolerance (PBFT), and Proof of Authority (PoA).

This enhancement is achieved through the redesign of the PyIncentiveBCSystem class, which now accepts generalized input parameters. These parameters include the "list of rounds", "list of participating nodes", "list of leader nodes per round", "rewards earned by leader nodes", and "transaction fees per round". These parameters can be extracted from blockchain implementations using any consensus algorithm, allowing researchers to simulate and compare incentivization models across different blockchain frameworks. Users can now input data from their specific blockchain implementation, regardless of the underlying consensus mechanism, enabling more comprehensive and versatile analyses of incentivization strategies.

5.3. Enhanced Accessibility through CodeOcean and Documentation

Recognizing the potential complexity of PyIncentiveBC for users who may not be well-versed in Python or blockchain technology, we have taken several steps to ensure broader accessibility:

- CodeOcean Container: PyIncentiveBC has been published as an executable container on CodeOcean [43], a platform designed for reproducible and traceable computational science. This container requires no installation or configuration, as the environment and package dependencies are pre-installed. Users can run the code directly on the platform, either with default inputs or customized parameters, and easily visualize and download the outputs generated. The container faithfully replicates the original development environment, allowing seamless integration into existing workflows, desktop or web GUIs, and Jupyter notebooks.
- Comprehensive Documentation and guide: We have developed extensive documentation and a step-by-step guide to assist users further. These resources offer detailed instructions on the installation, setup, and execution of the tool, making it accessible even to those with limited technical expertise. The documentation and guide, detailed in the README.md file, are available on our GitHub and CodeOcean repositories, ensuring that users can easily access the support they need to utilize PyIncentiveBC effectively.

6. Test Cases: Illustrative Scenarios and Real-World Case Studies

6.1. Illustrative Scenarios with Synthetic Exemplary Data

In this section, we will illustrate the use of PyIncentiveBC via a demonstration that will use fixed input data (e.g., list of nodes, list of rounds, list of transaction fees, leader reward, node score, etc.) to highlight the final result of the approaches implemented and identify the contribution of our approach previously presented in our previous work [26]. It should be noted that our implementation can rely on randomly generated data to make it easy to execute and visualize the results of each approach. Our scenario is implemented

in Jupyter Notebooks to execute, step by step, on a single canvas the different bricks of PyIncentiveBC in an interactive way and visualize the result of each step.

Figure 6 illustrates one way to integrate PyIncentiveBC into a Jupyter Notebook. Lines 4 to 13 of the first code section show the different basic data to initialize before calling the different methods of the "PyIncentiveBCSystem" class. For this scenario, the master data is initialized as follows:

1- Exploring Blockchain Incentivization with PyIncentiveBC: Synthetic Data Scenarios

A Comprehensive Analysis of Reward and Penalty Mechanisms Using Simulated Blockchain Data. This section uses synthetic data to explore and simulate various reward and penalty mechanisms in blockchain systems. Through controlled scenarios, we gain insights into the potential effects of different incentivization strategies.

```
[9]  1 import pandas as pd
     2 from PyIncentiveBCSystem import PyIncentiveBCSystem
     3
     4 leader_reward = 10.0
     5 transaction_fees = 2.0
     6 saved_part = 0.1
     7 distributed_part = 0.9
     8 df_approach_1 = pd.DataFrame()
     9 df_approach_2 = pd.DataFrame()
    10 all_df_result = pd.DataFrame()
    11 node_list = ["1","2","3","4","5","6","7","8","9","10"]
    12 round_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
    13 transaction_fees = [2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0]
    14
    15 reward_system = PyIncentiveBCSystem(node_list, round_list, leader_reward, transaction_fees)

[10] 1 df_approach_1 = reward_system.start_approach_1()
     2 score_nodes = [0.44, 0.68, 0.61, 0.87, 0.28, 0.12, 0.09, 0.66, 0.72, 0.87]
     3 df_approach_2 , score_nodes = reward_system.start_approach_2(saved_part, distributed_part, score_nodes)

[4]  1 all_df_result_1,all_df_result_2 = reward_system.compare_approaches(df_approach_1, df_approach_2, score_nodes)
     2 fig1 = reward_system.plot_graph_1(all_df_result_1, True)  # graph 1
     3 fig1.show()
     4 fig2 = reward_system.plot_graph_2(all_df_result_2, True)   # graph 2
     5 fig2.show()
     6 fig3 = reward_system.plot_graph_3(all_df_result_2)   # graph 3
     7 fig3.show()
```

Figure 6. A PyIncentiveBC demo using synthetic data implemented in Jupyter Notebook (deployed in Google Colaboratory Service).

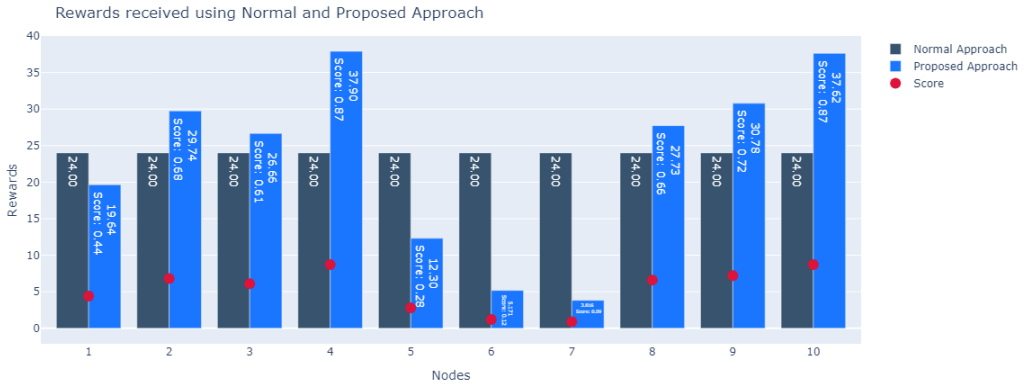
- leader_reward = 10.0;
- transaction_fees = 2.0;
- saved_part = 0.1;
- distributed_part = 0.9.

After instantiating the PyIncentiveBCSystem class (line 15) with initialization data, we launch the execution of the first approach (line 1 of the second code section) and then display the final result.

Lines 2 and 3 make it possible to do the same thing for approach 2: we launch the execution and then display the final result.

The call to the "compare_approaches()" function is made in the third section code to restructure the results obtained by the approaches implemented and illustrate them in ergonomic graphs explained below.

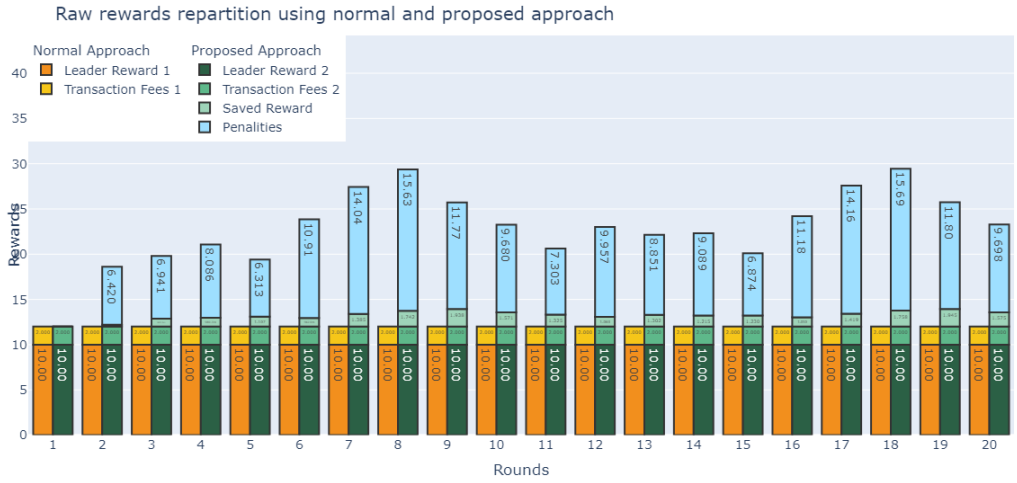
For our demonstration, three graphs are generated making it easier to compare the contributions between the two approaches implemented: Figure 7 clearly illustrates that nodes with a high score compared to others saw their profits increase (node 4 for example which received 37.90 tokens instead of 24 tokens). However, nodes with a low score experienced a drastic decrease in profit: node 7 received just 3.82 tokens instead of the 24 tokens that it might have collected if our proposed approach was not activated.



Node	Score	Normal Approach	Proposed Approach
1	0.440	24.000	19.639
2	0.680	24.000	29.739
3	0.610	24.000	26.658
4	0.870	24.000	37.898
5	0.280	24.000	12.305
6	0.120	24.000	5.171
7	0.090	24.000	3.816
8	0.660	24.000	27.730
9	0.720	24.000	30.781
10	0.870	24.000	37.623

Figure 7. Rewards received using Normal and Proposed Approach.

Figure 8 shows a comparison of raw rewards (without deduction of penalties) generated by the normal approach and the proposed approach. We can see that the new sources of income proposed by the new approach occupy a significant percentage of the raw rewards of each round.



Round	Leader Reward 1	Transaction Fees 1	Leader Reward 2	Transaction Fees 2	Saved Reward	Penalities
1	10.000	2.000	10.000	2.000	0.000	0.000
2	10.000	2.000	10.000	2.000	0.200	6.420
3	10.000	2.000	10.000	2.000	0.862	6.941
4	10.000	2.000	10.000	2.000	0.980	8.086
5	10.000	2.000	10.000	2.000	1.107	6.313
6	10.000	2.000	10.000	2.000	0.942	10.911
7	10.000	2.000	10.000	2.000	1.385	14.037
8	10.000	2.000	10.000	2.000	1.742	15.633
9	10.000	2.000	10.000	2.000	1.938	11.770
10	10.000	2.000	10.000	2.000	1.571	9.680
11	10.000	2.000	10.000	2.000	1.325	7.303
12	10.000	2.000	10.000	2.000	1.063	9.957
13	10.000	2.000	10.000	2.000	1.302	8.851
14	10.000	2.000	10.000	2.000	1.215	9.089
15	10.000	2.000	10.000	2.000	1.230	6.874
16	10.000	2.000	10.000	2.000	1.010	11.181
17	10.000	2.000	10.000	2.000	1.419	14.164
18	10.000	2.000	10.000	2.000	1.758	15.694
19	10.000	2.000	10.000	2.000	1.945	11.803
20	10.000	2.000	10.000	2.000	1.575	9.698

Figure 8. Raw Rewards repartition using normal and proposed approach.

With the proposed approach, the sources of income have diversified compared to the normal approach. This increase will provide the necessary tokens to reward all participants proportionally according to their score as already mentioned above. The graph in Figure 9 shows the distribution of raw rewards according to their origin during all simulation rounds.

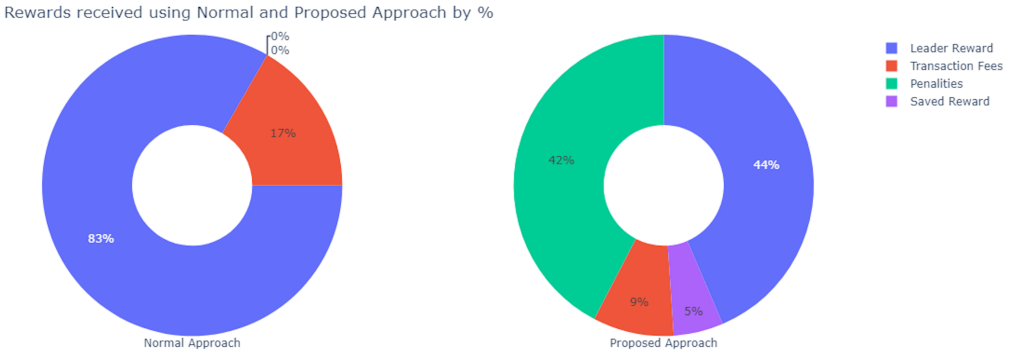


Figure 9. Raw Rewards repartition using normal and proposed approach represented in percentage.

6.2. Real-World Case Study

To demonstrate the applicability of PyIncentiveBC in real-world scenarios, we conducted a case study using data from the Bitcoin blockchain, which employs Proof of Work (PoW) as its consensus algorithm. We analyzed the rewards distribution for Bitcoin miners over a day (11 August 2024), comparing the current reward mechanism to our proposed approach.

6.2.1. Methodology

- Data Collection: We obtained Bitcoin block data from Blockchair [44], a blockchain analytics engine, on 11 August 2024. This dataset, spanning blocks 856,222 to 856,382, includes information about the miner (block creator), the reward amount generated, and the total transaction fees for each block (Table 1).

Table 1. An Overview of Block Rewards and Fees Distribution in Bitcoin Blockchain.

Block ID	Transaction Fees	Generated Reward	Reward Total per Round	Leader Node
856,222	0.06015836	3.125	3.18515836	AntPool
856,223	0.06024748	3.125	3.18524748	ViaBTC
856,224	0.02683187	3.125	3.15183187	AntPool
856,225	0.03156567	3.125	3.15656567	AntPool
856,226	0.04934574	3.125	3.17434574	Foundry USA Pool
856,227	0.05281368	3.125	3.17781368	Foundry USA Pool
856,228	0.0268494	3.125	3.1518494	AntPool
856,229	0.03970941	3.125	3.16470941	Foundry USA Pool
856,230	0.0292995	3.125	3.1542995	AntPool
856,231	0.03004645	3.125	3.15504645	AntPool
856,232	0.02757023	3.125	3.15257023	AntPool
856,233	0.02818699	3.125	3.15318699	Foundry USA Pool
856,234	0.03242674	3.125	3.15742674	Foundry USA Pool
856,235	0.03383881	3.125	3.15883881	Foundry USA Pool
856,236	0.03435604	3.125	3.15935604	ViaBTC
...	...	...	...	...
856,381	0.02353156	3.125	3.14853156	Unknown
856,382	0.05294336	3.125	3.17794336	F2Pool

- Miner Evaluation: We evaluated each miner based on two criteria, retrieved on 10 August 2024, related to their previous activities:

  - Average Block Health (AvgBH): A measure of how many transactions appear intentionally excluded from a block. 100% health indicates no intentionally excluded transactions [45].
  - Empty Blocks Mined (EBM): Retrieved from the "mempool.space" platform over the past 24 months, this represents the number of blocks the miner created that contain no transactions [46]. Although Bitcoin's protocol rewards empty blocks, we propose adjusting the reward structure to penalize this practice, encouraging miners to include transactions and improve network efficiency.
- Score Calculation: We calculated a score for each miner using Formula (A7), considering two rules as previously mentioned: Node's Block Health (NBH), based on the AvgBH value, and Preventing Empty Blocks (PEB), based on EBM.
  The Conformity Rate (CR) quantifies the degree of adherence to a specified rule, with values closer to 1 indicating a higher level of compliance by the participant. The Conformity Rate for the NBH rule is calculated as (Formula (3)):

$$ConformityRate_{ruleNBH} = AvgBH \times 0.01 \quad (3)$$

  while the Formula (4) determines the Conformity Rate for the PEB rule:

$$ConformityRate_{rulePEB} = 1 - (EBM \times 0.01) \quad (4)$$

  The final scores for known miners were calculated as described in Table 2.

Table 2. Performance Metrics of Bitcoin Miners: Node's Block Health and Preventing Empty Blocks.

Miners	Node's Block Health			Preventing Empty Blocks			Score
	AvgBH	CR	Coef.	EBM	CR	Coef.	
AntPool	98.5%	0.985	1	35.7%	0.643	1	0.814
Binance	97.15%	0.9715	1	13.3%	0.867	1	0.91925
F2Pool	98.16%	0.9816	1	16.7%	0.833	1	0.9073
Foundry USA Pool	98.82%	0.9882	1	0%	1	1	0.9941
Luxor	98.72%	0.9872	1	3.4%	0.966	1	0.9766
MaraPool	97.62%	0.9762	1	0%	1	1	0.9881
Poolin	98.28%	0.9828	1	1.7%	0.983	1	0.9829
SBICrypto	98.92%	0.9892	1	1.7%	0.983	1	0.9861
SlushPool/Braiins Pool	98.73%	0.9873	1	3.4%	0.966	1	0.97665
SpiderPool	94.38%	0.9438	1	5.1%	0.949	1	0.9464
ViaBTC	98.67%	0.9867	1	13.9%	0.861	1	0.92385
Unknown	-	-	-	-	-	-	0.814

CR: Conformity Rate.

  The Coefficient for each rule is set to 1, representing full weight in the final score. Given the inherent difficulty in retrieving the historical data of anonymous nodes (unknown miners), it is not feasible to construct a history-based score for them. Consequently, we propose assigning these nodes the lowest score derived from our calculations. This approach ensures that participants with known participation histories are not unfairly disadvantaged.
- Simulation: We used PyIncentiveBC to simulate two scenarios: (a) The current Bitcoin reward mechanism (Approach 1); (b) Our proposed dynamic, proportionate incentivization mechanism (Approach 2) (Figure 10).

2- Real-World Blockchain Incentives: A Case Study Using Bitcoin Data

Applying PyIncentiveBC to Analyze Incentivization Mechanisms in the Bitcoin Blockchain. This section leverages real-world Bitcoin blockchain data to evaluate the effectiveness of reward and penalty mechanisms. By applying the PyIncentiveBC framework, we can assess the impact of these mechanisms in a real blockchain environment.

```
[5]  leader_reward = 3.125
     saved_part = 0.1
     distributed_part = 0.9
     df_approach_1 = pd.DataFrame()
     df_approach_2 = pd.DataFrame()
     all_df_result = pd.DataFrame()
     node_list = ["AntPool","Binance","F2Pool","FoundryUSAPool","Luxor","MaraPool","Poolin","SBICrypto","SlushPool","SpiderPool","ViaBTC","Unknown"]
     elected_leader_node_list = ["AntPool","ViaBTC","AntPool","AntPool","FoundryUSAPool","FoundryUSAPool","AntPool","FoundryUSAPool","AntPool","AntPo
     round_list = [856222,856223,856224,856225,856226,856227,856228,856229,856230,856231,856232,856233,856234,856235,856236,856237,856238,856239,856
     transaction_fees = [0.06015836,0.06024748,0.02683187,0.03156567,0.04934574,0.05281368,0.0268494,0.03970941,0.0292995,0.03004645,0.02757023,0.028

     reward_system = PyIncentiveBCSystem(node_list, round_list, leader_reward, transaction_fees, elected_leader_node_list)

[6]  df_approach_1 = reward_system.start_approach_1()
     score_nodes = [0.814,0.91925,0.9073,0.9941,0.9766,0.9881,0.9829,0.9861,0.97665,0.9464,0.92385,0.814]
     df_approach_2 , score_nodes = reward_system.start_approach_2(saved_part, distributed_part, score_nodes)

[8]  all_df_result_1,all_df_result_2 = reward_system.compare_approaches(df_approach_1, df_approach_2, score_nodes)
     fig1 = reward_system.plot_graph_1(all_df_result_1, True)  # graph 1
     fig1.show()
     fig2 = reward_system.plot_graph_2(all_df_result_2, True)   # graph 2
     fig2.show()
     fig3 = reward_system.plot_graph_3(all_df_result_2)   # graph 3
     fig3.show()
```

Figure 10. A PyIncentiveBC demo using real data from the Bitcoin blockchain (deployed in Google Colaboratory Service).

6.2.2. Results

This real-world example illustrates how PyIncentiveBC can be applied to existing blockchain systems and provides insights into potential improvements in incentivization mechanisms.

For our demonstration, we also generated three graphs to facilitate the comparison of the contributions of our approach on a blockchain-based system (Bitcoin) operating with Proof of Work as the consensus algorithm.

The graph in Figure A1 in Appendix C and its corresponding descriptive table (Table 3) clearly illustrate that participants with higher scores experienced significant increases in their profits. For instance, participants such as Foundry USA Pool, MaraPool, and SBICrypto received additional tokens, with gains reaching up to 4.665 tokens. We also observe that participants like Poolin and Luxor, who had not earned any rewards during the entire period of our analysis (24 h on 11 August 2024), were able to secure substantial gains (4.994 and 4.962 bitcoins, respectively) thanks to their high compliance scores and the continuous revenue streams generated by the implementation of our new approach. In contrast, nodes with a low score (0.814) experienced a drastic reduction in their profits. For example, the participant AntPool received only 102.389 tokens, instead of the 123.431 tokens it could have accumulated if our proposed approach had not been activated.

Table 3. Score, Normal Approach, Proposed Approach, and Revenue Variation for various nodes.

Node	Score	Normal Approach	Proposed Approach	Revenue Variation
AntPool	0.814	123.431	102.389	−17.05%
Binance	0.919	3.169	7.514	+137.11%
F2Pool	0.907	44.249	43.928	−0.73%
FoundryUSA	0.994	151.935	152.898	+0.63%
Luxor	0.977	0	4.962	+4.962 bitcoins
MaraPool	0.988	12.655	17.245	+36.27%
Poolin	0.983	0	4.994	+4.994 bitcoins
SBICrypto	0.986	9.533	14.198	+48.94%
SlushPool	0.977	6.366	10.931	+72.05%
SpiderPool	0.946	19.031	22.312	+17.24%
ViaBTC	0.924	79.177	76.016	−3.99%
Unknown	0.814	60.164	51.895	−13.74%

Table A1, which provides the data for Figure A2, compares the Bitcoin Raw Rewards (before penalty deductions) generated by the normal approach and the proposed approach. The data demonstrates that the new revenue streams introduced by the proposed approach constitute a significant percentage of the raw rewards in each round.

The proposed approach makes revenue sources more diversified than the normal approach. This diversification ensures the availability of an adequate amount of Bitcoin to proportionally reward all participants according to their respective scores, as previously discussed. Figure A3 illustrates the repartition of Bitcoin Raw Rewards, using normal and proposed approaches, according to their origin across all simulation rounds.

The results indicate that our proposed approach applies to existing blockchain systems and offers valuable insights into potential enhancements of incentivization mechanisms. With its ability to handle diverse real-world scenarios, PyIncentiveBC provides a robust platform for researchers to benchmark and improve blockchain incentivization strategies. Ultimately, it could contribute to more efficient and robust blockchain networks by better aligning miner incentives with the overall network objectives.

7. Impact

We will now provide an overview of the scope of our simulator and how it influences and has a significant impact on both the research environment and education.

7.1. Research & Scientific

The idea for a new reward and punishment approach was born out of a proposal made for Blockchain-based systems in 2019 [26]. At the time, we investigated the possibility of offering new, permanent, and tangible revenue streams to reward active and honest participants. We have successfully evaluated our approach and demonstrated its effectiveness, via our simulator, to a PhD thesis committee at Hassan II University [47]. However, since the tool used in benchmarking with an already existing system has not been publicly disclosed, we thought of making it open source while adapting it so that the scientific communities can use it directly or integrate it into their existing programs.

The systematic review [48] demonstrates that the number of proposed approaches regarding consensus algorithms and their loyalty systems has increased in recent years. Despite the claimed contributions, the implementation of the proposed approaches is generally not publicly available. Moreover, almost all of the approaches do not highlight their contributions through simple, understandable, and adaptable simulation programs, which makes it difficult to compare the performance of new approaches with those already existing. PyIncentiveBC is considered a practical tool to address this issue by offering researchers two easy-to-understand examples that can be used as a model or template to facilitate the implementation of their own approaches and visualize their contributions.

The design and extensibility of PyIncentiveBC allow scientific communities to significantly reduce the effort and time required for the development and testing of such a tool. Our simulator has the potential to have a positive impact by encouraging researchers in this field to enrich their scientific productions with quality implementations thus providing reproducible and comparable results with two basic approaches offered by default with PyIncentiveBC. Moreover, our simulator generates ergonomic graphs which can be enriched later by adding other graphs exploiting the already structured results. Adopting PyIncentiveBC by researchers will enable them to reduce the high costs associated with resource allocation and the complexity of implementing proofs of concept (PoCs) for their novel reward and punishment approaches. Indeed, our simulator can be adapted to different use cases via an easy configuration of its parameters such as the number of nodes, the number of mining rounds as well as the different desired values according to the needs, etc. This encourages the scientific research community to innovate more, to explore other avenues of research, and to test their approaches on a large scale, without limits or constraints related to the hardware or software resources to be implemented.

Additionally, by combining PyIncentiveBC with other AI or optimization algorithm-

based systems, researchers can automate the generation of exemplary and relevant data for training, testing, and validating their proposals or models. This enables them to make their systems Blockchain-based, using a reward and punishment mechanism, profitable while ensuring a sufficient level of commitment of the participants of said system. Our software is therefore essential for research teams and organizations striving to stabilize and secure their Blockchain-based systems.

7.2. Education

Among the major objectives of our simulator, is to ensure an efficient and easy implementation of the approaches by researchers, even for those who are not familiar with programming. This will facilitate the sharing of information about the claimed benefits with fellow researchers, regardless of their computer experience. Additionally, it will also benefit their students who will also be able to take advantage of our simulator as a training tool. This will not only help them understand how reward and punishment systems work but also enable them to implement any proposal related to an already existing or newly designed approach within the context of research or work related to the computer development of Blockchain-based systems. In addition, using the simulator as a training and learning tool is an increasingly widespread teaching method [49,50]. It promotes transmitting knowledge and skills to learners in a specific context, with its data sets, resources, and experimental scenarios that can be personalized according to their studies. This ensures correct and reproducible execution of the simulations set up. The detailed technical documentation of each approach implemented in our simulator will easily convert it into training and learning support, including details of the algorithms and comments on the calculations applied at each step of each approach implemented.

In conclusion, PyIncentiveBC, already published on GitHub [51] and CodeOcean container [43], a platform for reproducible and traceable computational science that supports collaborative research in the cloud [52]. This container replicates the original software environment used to implement the PyIncentiveBC program. This latter can be easily integrated into existing programs or linked to desktop or web GUIs. It can also be used through Jupyter notebooks, as we did in our simulation. Thanks to the code's total parameterization, it offers the possibility of easily configuring and executing a wide range of numerical simulations. Moreover, the diverse methods that have been proposed are developed to provide clear descriptions and easy usability. These methods allow developers and researchers to personalize and seamlessly integrate them into their programs or research projects concerning the rewards and penalties applied in consensus algorithms used in Blockchain-based systems.

8. Conclusions and Outlook

PyIncentiveBC, a Python implementation of rewards and penalties approaches used in Blockchain-based systems, was presented in this article. The theoretical basis of a proposed approach and its contributions were discussed compared to a normal approach.

In an illustrative example, the methods ensuring the different elementary calculations of two approaches put in place, the comparison of their contributions, and the display of the final results (via graphs) were presented. PyIncentiveBC features are adaptable and well documented in the code itself helping developers understand the program for better implementation and evolution with their own settings and approaches.

The extensibility of PyIncentiveBC and its ease of integration into other programs gives it the possibility of being used directly by researchers to visualize the contribution of our approach via personalization of the basic data of the two approaches already implemented, or by developers who can integrate PyIncentiveBC into their programs and implement their new approaches based on existing or new consensus algorithms and test their contributions.

While PyIncentiveBC provides a valuable tool for evaluating incentivization mechanisms in blockchain-based systems, several research gaps and future directions remain.

Future research could explore the integration of advanced machine learning algorithms with PyIncentiveBC to predict and optimize participant behavior in blockchain systems. Additionally, there is a significant gap in the availability of tools that offer operational simulation and analysis of blockchain incentivization mechanisms—a gap that PyIncentiveBC aims to bridge. Further enhancements could involve integrating additional consensus algorithms to create a more comprehensive testing environment. Additionally, developing methods to incorporate real-time, real-world blockchain data would significantly enhance the simulator's practical relevance. Moreover, creating a user-friendly graphical interface for PyIncentiveBC would make it more accessible to a broader range of users, including those with limited programming experience. By addressing these gaps, PyIncentiveBC paves the way for the development of more robust, adaptable, and user-friendly incentivization strategies in blockchain-based systems.

Author Contributions: A.O.: Conceptualization, Methodology, Software, Writing—Original Draft. M.H.: Supervision, Investigation, Formal analysis, Writing—Review & Editing, Z.C.: Investigation, Formal analysis, N.A.: Validation, Supervision, Writing—Review & Editing. M.O.: Validation, Supervision, Writing—Review & Editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The software and data presented in this study are openly available at https://github.com/ouaguid/pyIncentiveBC (accessed on 21 August 2024) and https://codeocean.com/capsule/2547976/tree/v1 (https://doi.org/10.24433/CO.4994364.v2) (accessed on 21 August 2024).

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

Abbreviations

The following abbreviations are used in this manuscript:

AvgBH	Average Block Health
CR	Conformity Rate
EBM	Empty Blocks Mined
FR	Final Reward
GR	Generated Reward
IR	Initial Reward
LR	Leader Reward
MN	Malicious Node
NBH	Node's Block Health
NPN	Non-Participating Node
P	Penalties
PEB	Preventing Empty Blocks
PN	Participating Node
PoS	Proof of Stake
PoW	Proof of Work
PR	Participating Reward
SR	Saved Reward
TF	Transaction Fees

Appendix A. Normal Approach (Approach 1)

The approach, referred to as the 'normal' approach, involves allocating the appropriate rewards (LR) to the leader node, along with the total transaction fees (TF) for the current round (Formula (A1)). However, other participants do not receive any rewards, even

though their participation incurs costs (in terms of processing, storage, network, etc.) and ensures the proper functioning of the Blockchain system they adhere to.

$$FR_{node} = \begin{cases} LR_{node} + TF & \text{, if node = Leader} \\ 0 & \text{, if node = Participant} \end{cases} \quad \text{(A1)}$$

Appendix B. Improved Approach (Approach 2)

As explained in Figure 2, The final reward (FR) of each node can be calculated as follows (Formula (A2)):

$$FR_{node} = \begin{cases} LR_{node} - P_{node} & \text{, if node = Leader} \\ PR_{node} - P_{node} & \text{, if node = Participant} \end{cases} \quad \text{(A2)}$$

The different elements used to allow the FR calculation are detailed in the following: Calculating the Participating Reward (PR) which is based on the GR (Generated Reward) collected. The latter will not be consumed in its entirety because the system must distribute part of the GR for the nodes that will participate in the next mining cycles. This specificity is taken into account in Formula (A3) in which we multiplied the initial GR by DP (Distributed Part) to finally have just the GR ready to be distributed to the participants equitably. The value of DP must also be greater than 0 and less than 1 ($DP \in]0, 1]$ with $DP = 1 - SP$).

$$PR_{node} = \frac{DP \times GR}{\sum node} \quad \text{(A3)}$$

Formula (A4) summarizes how *Generated Reward* GR is built:

$$GR = TF + PrevGR + \sum PrevP \quad \text{(A4)}$$

As already illustrated previously in Figure 2, Generated Reward (GR) is built with:

- Transaction fees (TF),
- The sum of the penalties debited from the nodes in the previous round of mining ($PrevP_{node}$),
- Part of the GR from the previous mining cycle ($PrevGR$) (Formula A5).

SP refers to the "Saved Part" and represents a ratio of the GR part that will be hoarded and used in future mining cycles. Its value must be greater than 0 and less than 1 ($SP \in]0, 1]$).

$$PrevGR_{round} = SP * GR_{round-1} \quad \text{(A5)}$$

The node score value can be included between 0 (the node does not comply with the protocol rules) and 1 (the node respects all protocol rules).

Based on the score assigned to each participating node, the system will be able to calculate the penalties (P_{node}) to be collected from non-conforming nodes (Formula (A6)). These penalties will be subtracted from the value of future node rewards while respecting the role played by the latter in the mining cycle:

- If the node is designated as Leader, the penalties will be deducted from Leader Reward (LR);
- For other participating nodes, penalties will be deducted from Participating Reward (PR).

$$P_{node} = \begin{cases} LR_{node} \times (1 - Score_{node}) & \text{, if node = Leader} \\ PR_{node} \times (1 - Score_{node}) & \text{, if node = Participant} \end{cases} \quad \text{(A6)}$$

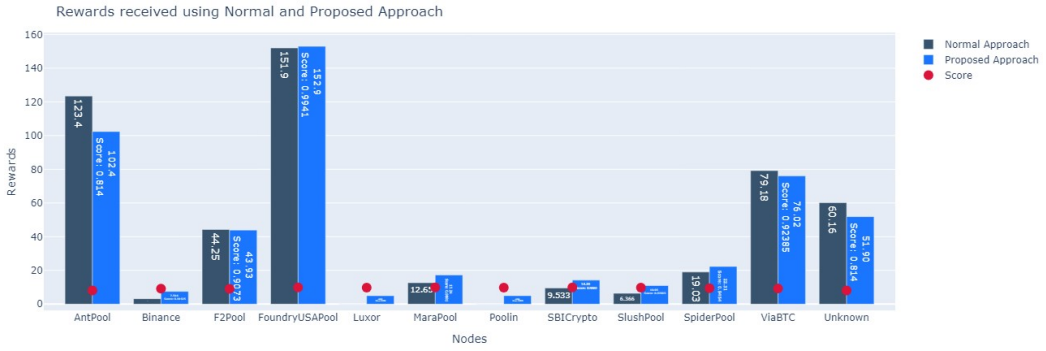
Once the compliance rate for each rule of the Blockchain protocol is calculated, it will be assigned a coefficient (which differs according to the criticality of the rule compared to the other rules of the protocol) so that the assembly is used to calculate the score of the node ($Score_{node} \in [0, 1]$) as defined in Formula (A7).

$$Score_{node} = \frac{\sum_{rule=1}^{\infty}(Coefficient_{rule} \times ConformityRate_{rule})}{\sum Coefficient} \quad \text{(A7)}$$

The calculation of the compliance rate ($ConformityRate_{rule} \in [0, 1]$) of each rule is presented in Formula (A8). The closer the *ConformityRate* value of a rule is to 1, the more that rule is respected by the participant.

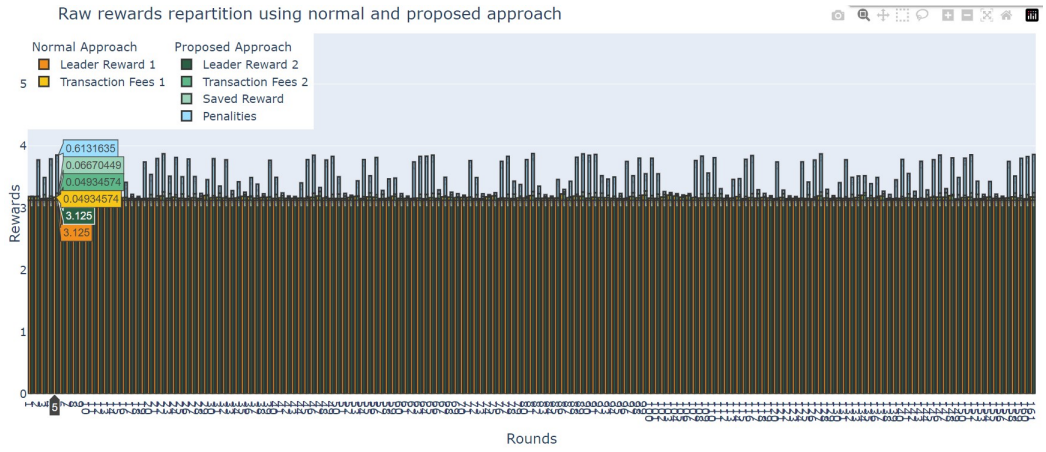
$$ConformityRate_{rule} = \frac{RequirementRespected}{\sum Requirements} \quad \text{(A8)}$$

Appendix C. Comparative Analysis of Bitcoin Raw Rewards Distribution: Normal vs. Proposed Approaches



Node	Score	Normal Approach	Proposed Approach
AntPool	0.814	123.431	102.389
Binance	0.919	3.169	7.514
F2Pool	0.907	44.249	43.928
FoundryUSAPool	0.994	151.935	152.898
Luxor	0.977	0.000	4.962
MaraPool	0.988	12.655	17.245
Poolin	0.983	0.000	4.994
SBICrypto	0.986	9.533	14.198
SlushPool	0.977	6.366	10.953
SpiderPool	0.946	19.031	22.312
ViaBTC	0.924	79.177	76.016
Unknown	0.814	60.164	51.895

Figure A1. Rewards Distribution Using Normal and Proposed Approaches in Bitcoin Dataset



Round	Leader Reward 1	Transaction Fees 1	Leader Reward 2	Transaction Fees 2	Saved Reward	Penalities
1	3.125	0.060	3.125	0.060	0.000	0.000
2	3.125	0.060	3.125	0.060	0.006	0.584
3	3.125	0.027	3.125	0.027	0.065	0.275
4	3.125	0.032	3.125	0.032	0.037	0.599
5	3.125	0.049	3.125	0.049	0.067	0.613

Figure A2. Overview of Bitcoin Raw Rewards Repartition Using Normal and Proposed Approaches

Table A1. Overview of Bitcoin Rewards and Penalties Distribution Across Rounds.

Round	Leader Reward 1	Transaction Fees 1	Leader Reward 2	Transaction Fees 2	Saved Reward	Penalties
1	3.125	0.060	3.125	0.060	0.000	0.000
2	3.125	0.060	3.125	0.060	0.006	0.584
3	3.125	0.027	3.125	0.027	0.065	0.275
4	3.125	0.032	3.125	0.032	0.037	0.599
5	3.125	0.049	3.125	0.049	0.067	0.613
6	3.125	0.053	3.125	0.053	0.073	0.064
7	3.125	0.027	3.125	0.027	0.019	0.030
8	3.125	0.040	3.125	0.040	0.008	0.585
9	3.125	0.029	3.125	0.029	0.063	0.058
10	3.125	0.030	3.125	0.030	0.015	0.588
11	3.125	0.028	3.125	0.028	0.063	0.612
12	3.125	0.028	3.125	0.028	0.070	0.615
13	3.125	0.032	3.125	0.032	0.071	0.063
14	3.125	0.034	3.125	0.034	0.017	0.029
15	3.125	0.034	3.125	0.034	0.008	0.023
16	3.125	0.041	3.125	0.041	0.007	0.242
17	3.125	0.030	3.125	0.030	0.029	0.037
18	3.125	0.028	3.125	0.028	0.010	0.024
19	3.125	0.027	3.125	0.027	0.006	0.584
20	3.125	0.030	3.125	0.030	0.062	0.324
21	3.125	0.030	3.125	0.030	0.042	0.601
22	3.125	0.069	3.125	0.069	0.067	0.613
23	3.125	0.033	3.125	0.033	0.075	0.281
24	3.125	0.051	3.125	0.051	0.039	0.600
25	3.125	0.033	3.125	0.033	0.069	0.277
26	3.125	0.028	3.125	0.028	0.038	0.599
27	3.125	0.040	3.125	0.040	0.067	0.276
28	3.125	0.030	3.125	0.030	0.038	0.042
...	...	...	...	...	...	...
159	3.125	0.037	3.125	0.037	0.039	0.600
160	3.125	0.024	3.125	0.024	0.039	0.614
161	3.125	0.053	3.125	0.053	0.070	0.615

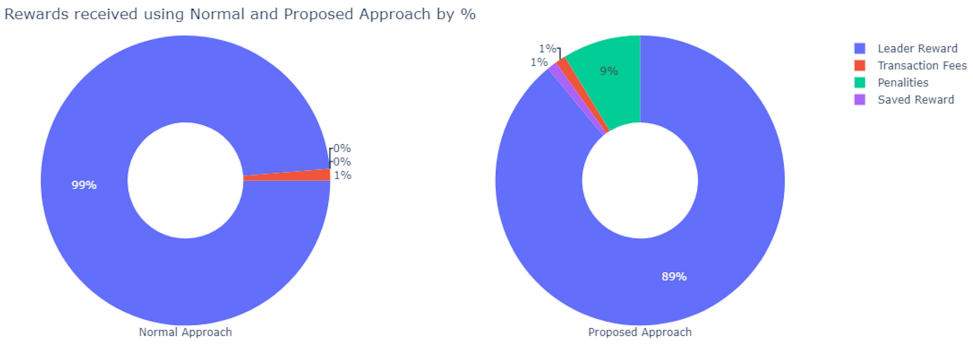


Figure A3. Repartition of Bitcoin Raw Rewards Using Normal and Proposed Approaches, Represented in Percentage.

References

1. Padmavathi, U.; Rajagopalan, N. Concept of blockchain technology and its emergence. In *Research Anthology on Convergence of Blockchain, Internet of Things, and Security*; IGI Global: Hershey, PA, USA, 2023; pp. 21–36.
2. Pragasam, T.T.N.; Thomas, J.V.J.; Vensuslaus, M.A.; Radhakrishnan, S. CEAT: Categorising Ethereum Addresses' Transaction Behaviour with Ensemble Machine Learning Algorithms. *Computation* **2023**, *11*, 156. [CrossRef]
3. Bellaj, B.; Ouaddah, A.; Bertin, E.; Crespi, N.; Mezrioui, A.; Bellaj, K. BTrust: A New Blockchain-Based Trust Management Protocol for Resource Sharing. *J. Netw. Syst. Manag.* **2022**, *30*, 64. [CrossRef]

4. Ren, Y.S.; Ma, C.Q.; Chen, X.Q.; Lei, Y.T.; Wang, Y.R. Sustainable finance and blockchain: A systematic review and research agenda. *Res. Int. Bus. Financ.* **2023**, *64*, 101871. [CrossRef]
5. Zouina, M.; Outtai, B. Towards a distributed token based payment system using blockchain technology. In Proceedings of the 2019 International Conference on Advanced Communication Technologies and Networking (CommNet), Rabat, Morocco, 12–14 April 2019; IEEE: New York, NY, USA, 2019; pp. 1–10.
6. Yazdinejad, A.; Dehghantanha, A.; Parizi, R.M.; Srivastava, G.; Karimipour, H. Secure intelligent fuzzy blockchain framework: Effective threat detection in iot networks. *Comput. Ind.* **2023**, *144*, 103801. [CrossRef]
7. Bobde, Y.; Narayanan, G.; Jati, M.; Raj, R.S.P.; Cvitić, I.; Peraković, D. Enhancing Industrial IoT Network Security through Blockchain Integration. *Electronics* **2024**, *13*, 687. [CrossRef]
8. Al-Sumaidaee, G.; Alkhudary, R.; Zilic, Z.; Swidan, A. Performance analysis of a private blockchain network built on Hyperledger Fabric for healthcare. *Inf. Process. Manag.* **2023**, *60*, 103160. [CrossRef]
9. Ouaguid, A.; Hanine, M.; Chiba, Z.; Abghour, N.; Ghazal, H. Analysis of Blockchain Integration in the e-Healthcare Ecosystem. In Proceedings of the 2023 6th International Conference on Advanced Communication Technologies and Networking (CommNet), Rabat, Morocco, 11–13 December 2023; IEEE: New York, NY, USA, 2023; pp. 1–8.
10. Bidry, M.; Ouaguid, A.; Hanine, M. Enhancing E-Learning with Blockchain: Characteristics, Projects, and Emerging Trends. *Future Int.* **2023**, *15*, 293. [CrossRef]
11. Ghorab, A.S.; Rasheed, R.S.; Salah, M.S.; AbuSamra, A.A. PalCert: A Blockchain-based Certificate Attestation and Verification System for HEIs in Palestine. In *Information and Communication Technology in Technical and Vocational Education and Training for Sustainable and Equal Opportunity: Business Governance and Digitalization of Business Education*; Springer: Singapore, 2024; pp. 123–139.
12. Hao, Y.; Helo, P.; Tsoniotis, N.; Toshev, R. Blockchain-Based Supply Chain System in Automotive Industry forSmall-and Medium-Sized Manufacturing. In *Blockchain Driven Supply Chains and Enterprise Information Systems*; Springer: Singapore, 2023; pp. 151–171.
13. Ouaguid, A.; Fathi, F.; Zouina, M.; Ouzzif, M.; Abghour, N. Androscanreg 2.0: Enhancement of Android Applications Analysis in a Flexible Blockchain Environment. *Int. J. Softw. Innov. (IJSI)* **2022**, *10*, 1–28. [CrossRef]
14. Amallah, M.A.; Abghour, N.; Moussaid, K.; El Omri, A.; Rida, M. Review on Blockchain and Access Control Systems. In *Advances on Smart and Soft Computing, Proceedings of ICACIn 2021, Casablanca, Morocco, 24–25 May 2021*; Springer: Singapore, 2022; pp. 235–246.
15. Lawrence, L.; Shreelekshmi, R. VECDSigL: Video integrity verification using elliptic curve digital signature links. *Softw. Impacts* **2023**, *15*, 100474. [CrossRef]
16. Abirou, M.; Abghour, N. A Review of Blockchain and the Benefits for Digital Marketing-Related Applications of Blockchain Integration. In *Advances on Smart and Soft Computing, Proceedings of ICACIn 2021, Casablanca, Morocco, 24–25 May 2021*; Springer: Singapore, 2022; pp. 355–365.
17. El Akrami, N.; Hanine, M.; Flores, E.S.; Aray, D.G.; Ashraf, I. Unleashing the Potential of Blockchain and Machine Learning: Insights and Emerging Trends from Bibliometric Analysis. *IEEE Access* **2023**, *11*. [CrossRef]
18. Xiong, H.; Chen, M.; Wu, C.; Zhao, Y.; Yi, W. Research on Progress of Blockchain Consensus Algorithm: A Review on Recent Progress of Blockchain Consensus Algorithms. *Future Internet* **2022**, *14*, 47. [CrossRef]
19. King, S.; Nadal, S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *Self-Publ. Paper* **2012**, *19*, 1.
20. Dziembowski, S.; Faust, S.; Kolmogorov, V.; Pietrzak, K. Proofs of space. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 16–20 August 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 585–605.
21. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. In *Decentralized Business Review*; Bitcoin: Nashville, TX, USA, 2008.
22. Abooleet, S.; Kinnett, S. A Systematic Review of Blockchain-based Loyalty Programs. In Proceedings of the 56th Hawaii International Conference on System Sciences, Honolulu, HI, USA, 3–6 January 2023.
23. Chen, Y.; Mandler, T.; Meyer-Waarden, L. Three decades of research on loyalty programs: A literature review and future research agenda. *J. Bus. Res.* **2021**, *124*, 179–197. [CrossRef]
24. The Loyalty Report 2019. 2020. Available online: https://cdn2.hubspot.net/hubfs/352767/TLR%202019/Bond_US%20TLR19%20Exec%20Summary%20Launch%20Edition.pdf (accessed on 20 June 2024).
25. Bitcoin Fee-to-Reward Ratio, Explained. 2023. Available online: https://cointelegraph.com/explained/what-is-bitcoins-fee-to-reward-ratio (accessed on 20 June 2024).
26. Ouaguid, A.; Abghour, N.; Ouzzif, M. Towards a new reward and punishment approach for blockchain-based system. In Proceedings of the 2019 International Conference on Systems of Collaboration Big Data, Internet of Things & Security (SysCoBIoTS), Casablanca, Morocco, 12–13 December 2019; IEEE: New York, NY, USA, 2019; pp. 1–7.
27. Larimer, D. Delegated proof-of-stake (dpos). *Bitshare Whitepaper* **2014**, *81*, 85.
28. Sompolinsky, Y.; Wyborski, S.; Zohar, A. PHANTOM GHOSTDAG: A scalable generalization of Nakamoto consensus: September 2, 2021. In Proceedings of the 3rd ACM Conference on Advances in Financial Technologies, Arlington, VA, USA, 26–28 September 2021; pp. 57–70.
29. Slimcoin—A Peer-to-Peer Crypto-Currency with Proof-of-Burn—"Mining without Powerful Hardware". 2014. Available online: https://github.com/slimcoin-project/slimcoin-project.github.io/raw/master/whitepaperSLM.pdf (accessed on 22 February 2020).

30. Gai, F.; Wang, B.; Deng, W.; Peng, W. Proof of reputation: A reputation-based consensus protocol for peer-to-peer network. In Proceedings of the Database Systems for Advanced Applications: 23rd International Conference, DASFAA 2018, Gold Coast, QLD, Australia, 21–24 May 2018; Proceedings, Part II 23; Springer: Cham, Switzerland, 2018; pp. 666–681.
31. Bentov, I.; Lee, C.; Mizrahi, A.; Rosenfeld, M. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y. In *ACM Sigmetrics Performance Evaluation Review*; Association for Computing Machinery: New York, NY, USA, 2014; Volume 42, pp. 34–37.
32. Nguyen, D.C.; Ding, M.; Pham, Q.V.; Pathirana, P.N.; Le, L.B.; Seneviratne, A.; Li, J.; Niyato, D.; Poor, H.V. Federated learning meets blockchain in edge computing: Opportunities and challenges. *IEEE Int. Things J.* **2021**, *8*, 12806–12825. [CrossRef]
33. Matsunaga, T.; Zhang, Y.; Sasabe, M.; Kasahara, S. An Incentivization Mechanism with Validator Voting Profile in Proof-of-Stake-Based Blockchain. *IEICE Trans. Commun.* **2022**, *105*, 228–239. [CrossRef]
34. Xiao, Y.; Zhang, N.; Li, J.; Lou, W.; Hou, Y.T. Distributed consensus protocols and algorithms. *Blockchain Distri. Syst. Secur.* **2019**, *25*, 40.
35. Gervais, A.; Karame, G.O.; Wüst, K.; Glykantzis, V.; Ritzdorf, H.; Capkun, S. On the security and performance of proof of work blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 3–16.
36. Tripathi, A.K.; Jain, A.; Kumar, A.; Singh, S.; Shukla, B. Exploring consensus algorithms: A comprehensive examination and comparative analysis. *AIP Conf. Proc.* **2024**, *3168*, 020036.
37. Proof-of-Stake Rewards and Penalties. 2024. Available online: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/rewards-and-penalties/ (accessed on 20 June 2024).
38. Slashing Penalties—The Long Term Evolution of Proof of Stake (POS). 2021. Available online: https://novuminsights.com/post/slashing-penalties-the-long-term-evolution-of-proof-of-stake-pos/ (accessed on 20 June 2024).
39. McKinney, W. Pandas: A foundational Python library for data analysis and statistics. In *Python for High Performance and Scientific Computing*; O'Reilly Media: Sevastopol, CA, USA, 2011; Volume 14, pp. 1–9.
40. Van Der Walt, S.; Colbert, S.C.; Varoquaux, G. The NumPy array: A structure for efficient numerical computation. *Comput. Sci. Eng.* **2011**, *13*, 22–30. [CrossRef]
41. Plotly Open Source Graphing Library for Python. 2023. Available online: https://plotly.com/python/ (accessed on 20 June 2024).
42. McKinney, W. Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference, Austin, TX, USA, 28 June–3 July 2010; Volume 445, pp. 51–56.
43. PyIncentiveBC: A Python Module for Simulation of Incentivization Mechanism Implemented in Blockchain-Based Systems. 2024. Available online: https://codeocean.com/capsule/2547976/tree/v2 (accessed on 19 August 2024).
44. Blockchair. 2024. Available online: https://gz.blockchair.com/bitcoin/blocks/ (accessed on 11 August 2024).
45. What Is Block Health? 2024. Available online: https://mempool.space/docs/faq#what-is-block-health (accessed on 11 August 2024).
46. Empty Block Report: A Data Driven Analysis of the Bitcoin Empty Block Phenomenon. 2024. Available online: https://research.mempool.space/empty-block-report/ (accessed on 11 August 2024).
47. Ouaguid, A. A Blockchain-Based Security Analysis Framework for Mobile Applications. Ph.D. Thesis, Hassan II University, Casablanca, Morocco, 2021. [CrossRef]
48. Xu, J.; Wang, C.; Jia, X. A Survey of Blockchain Consensus Protocols. *ACM Comput. Surv.* **2023**, *55*, 278. [CrossRef]
49. Dickson-Karn, N.M.; Orosz, S. Implementation of a Python Program to Simulate Sampling. *J. Chem. Educ.* **2021**, *98*, 3251–3257. [CrossRef]
50. Stamenković, S.; Jovanović, N.; Chakraborty, P. Evaluation of simulation systems suitable for teaching compiler construction courses. *Comput. Appl. Eng. Educ.* **2020**, *28*, 606–625. [CrossRef]
51. PyIncentiveBC Source Code. 2024. Available online: https://github.com/ouaguid/pyIncentiveBC (accessed on 19 August 2024)
52. Clyburne-Sherin, A.; Fei, X.; Green, S.A. Computational reproducibility via containers in social psychology. *Meta-Psychology* **2019**, *3*, MP.2018.892. [CrossRef]