Journal of Internet Services and Applications, 2026, 17:1, doi: 10.5753/jisa.2026.5926

Selecting Consensus Algorithm Integrations in a DAG-based Blockchain for IoT Using Genetic Algorithms

Anderson Melo de Morais [**Universidade Federal de Pernambuco** | ***amm6@cin.ufpe.br***]
Fernando Antonio Aires Lins [**Universidade Federal Rural de Pernambuco** | ***fernandoaires@ufrpe.br***]
Nelson Souto Rosa [**Universidade Federal de Pernambuco** | ***nsr@cin.ufpe.br***]

Centro de Informática - CIn, Universidade Federal de Pernambuco, Av. Jorn. Aníbal Fernandes, s/n - Cidade Universitária, Recife - PE, 50740-560, Brazil.

Received: 15 April 2025 • **Accepted:** 01 December 2025 • **Published:** 11 February 2026

Abstract The Internet of Things (IoT) drives technological advances across various sectors by enabling seamless communication among smart devices. However, significant challenges remain regarding the integrity and reliability of the data stored by these devices. Traditional blockchain solutions, such as those based on Proof of Work (PoW), are generally unsuitable for IoT applications due to their high computational resource demands. Although approaches combining multiple consensus algorithms have emerged as alternatives to optimise performance and security, determining the best combination for each scenario remains an open problem. This paper proposes a strategy based on Genetic Algorithms (GAs) to adaptively select and combine consensus algorithms, thus improving blockchain efficiency in IoT environments. The approach was evaluated on a test blockchain, OmniBlock, implemented using a Directed Acyclic Graph (DAG) and designed specifically for evaluation purposes in IoT applications. OmniBlock supports multiple consensus algorithms, including Proof of Authority (PoA), Proof of Stake (PoS), Proof of Work (PoW), Practical Byzantine Fault Tolerance (PBFT), Raft, and others. The consensus algorithm combination is chosen based on performance attributes. All combinations of consensus algorithms evaluated in this work were suggested by GAs; the practical feasibility of each is analyzed empirically. Experimental results indicate that the evolutionary optimization-based strategy performs better across most of the combinations suggested by the GAs.

Keywords: Blockchain, Internet of Things, Consensus Algorithms, Information Security.

1 Introduction

The Internet of Things (IoT) is transforming several sectors by enabling connectivity and communication between smart devices. However, this technological revolution presents significant challenges, particularly in terms of security, scalability, and data management efficiency. One issue is validating and recording transactions securely and quickly, without overloading IoT devices.

Blockchain technology has proven to be a promising solution, offering a decentralised data management architecture. However, traditional blockchains, such as those based on Proof of Work (PoW), are often unsuitable for IoT environments due to their high computational requirements. Currently, many solutions propose combinations between different consensus algorithms [Tapwal *et al*., 2021], [Zhipeng, 2019], [Rasolroveicy and Fokaefs, 2020].

However, most existing combinations are developed for a specific IoT domain or utilise only two or three consensus algorithms. The blockchain developed in this work utilises ten consensus algorithms, chosen from among the most commonly used today, which can be adaptively combined for use in different IoT domains.

Combinations are advantageous in most cases because different algorithms can be used in different phases of the consensus process, thereby improving factors such as block confirmation time and resource consumption. However, they can introduce excessive complexity and interoperability challenges, compromising the blockchain's security and consistency. Properly selecting which algorithms to integrate and how to combine them is crucial to maximising their benefits and minimising their limitations.

This article presents a strategy for selecting and combining consensus algorithms to improve the efficiency and performance of blockchain in IoT environments. The proposed strategy includes a detailed comparative assessment of different combinations, highlighting best practices and critical factors to be considered during their implementation.

The selection strategy uses input matrices that capture the characteristics of the algorithms and the requirements of various IoT domains. Based on these matrices, two evolutionary algorithms (Simple Genetic Algorithm (SGA) and Adaptive Genetic Algorithm (AGA)) are used to explore and optimise the possible combinations between the algorithms. These algorithms select the combination that best meets the requirements of the IoT domain. All the evaluated combinations were suggested by the genetic method, and the viability of each is discussed. This paper is an extension of the research developed by de Morais *et al*. [2024], where the selection strategy consisted of using weighted average calculations to choose combinations. The main advancement presented in this work is using Genetic Algorithms [Holland, 1975], which adapt at runtime and allow for more efficient choices of consensus combinations.

To evaluate the strategy, a blockchain supporting multiple consensus algorithms (e.g., Proof of Authority (PoA), Proof of

Stake (PoS), Proof of Work (PoW), Practical Byzantine Fault Tolerance (PBFT), and Raft), each with its unique advantages and disadvantages, was utilised. This evaluation indicates the blockchain's adaptability, as the combination of these algorithms is designed to optimise its performance and meet the diverse IoT demands.

This paper is organised as follows. Section 2 provides an overview of the main concepts necessary to understand this work. Section 3 presents the strategy for selecting consensus algorithm combinations. Section 4 describes the performance analysis. Section 5 presents papers related to this research topic. Section 6 concludes the paper and presents suggestions for future work.

2 Background

This section introduces the main concepts necessary to understand the contributions of this paper.

2.1 Blockchain for IoT

Blockchain consists of a distributed data structure with a chain of blocks; each block contains a set of encrypted transactions [Monrat *et al*., 2019]. Each block is linked to the previous one through a cryptographic hash, forming a linear sequence of blocks. The main characteristics of blockchain are decentralisation and immutability.

Blockchain decentralisation distributes authority and data among all network participants. The immutable record of data ensures that it cannot be altered or deleted, thereby maintaining the integrity and reliability of the information. When information is recorded, the computational cost of changing or deleting it without the consensus of other participants is currently very high or virtually impossible.

Blockchain technology was initially developed for cryptocurrencies, such as Bitcoin [Nakamoto, 2008], but it has since expanded to areas beyond finance, including education, the judiciary, and the Internet of Things [Kamran *et al*., 2020].

According to Bhushan et al. [Bhushan *et al*., 2021], a blockchain for IoT must meet certain requirements, such as a low transaction confirmation time. Another critical factor is the efficient consumption of computing resources, as most IoT devices rely on batteries or have resource limitations, such as limited memory and processing power. The consensus algorithm must efficiently consume resources such as CPU and RAM.

2.2 Consensus Algorithms

A fundamental feature of blockchain technology is consensus algorithms [Liao and Cheng, 2023]. These algorithms allow participants to agree on the state of the network and promote the integrity of recorded transactions [Monrat *et al*., 2019]. Consensus algorithms, such as those used in blockchain or other decentralised technologies, are essential in environments without a central authority to validate transactions.

The consensus algorithms have been developed to meet the requirements of blockchain. Among the most used are Proof of Work (PoW), popularised by Bitcoin, which requires the resolution of complex cryptographic problems to validate transactions, and Proof of Stake (PoS) [Gaži *et al*., 2019], where cryptocurrencies are used to determine the probability for a participant to create a new block [Wu *et al*., 2019].

The consensus process executes four distinct steps, as shown in Figure 1. The first step is to choose the validator node to start the transaction validation process and create the new block. The second step involves creating a block that contains the values to be recorded on the blockchain. The third step, which is not explicitly executed by some consensus algorithms, such as PoW, is to validate the new block before it is included on the blockchain. The fourth step includes replicating the new block among blockchain participants.

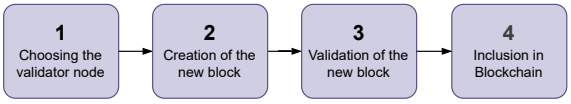

Figure 1. Steps of the consensus process.

2.3 Directed Acyclic Graphs

A Directed Acyclic Graph (DAG) is a data structure consisting of vertices interconnected by directed edges, where the edges cannot form closed loops [Digitale *et al*., 2022]. This feature ensures that there are no loops, allowing for a unidirectional flow of information. DAGs have been widely used in dynamic programming [Mäkinen *et al*., 2019], dependency management systems [Isaacs and Gamblin, 2018], and are now emerging as a promising approach in distributed ledger technologies such as blockchain [Wang *et al*., 2023].

It is possible to implement blockchains using DAG [Wang *et al*., 2023]. Like a blockchain, which utilizes a linear chain of blocks, a DAG-based data structure can also serve as a distributed database to store cryptographic data in an immutable and transparent manner. The primary difference lies in the architecture and consensus algorithms employed.

A DAG usage example is IOTA, a cryptocurrency developed for IoT that uses a DAG called Tangle [Silvano and Marcelino, 2020]. Tangle differs from a traditional blockchain because it utilizes a structure with nodes that represent transactions, rather than a linear chain of blocks. In the blockchain terminology, Tangle comprises blocks that contain only one transaction [Son *et al*., 2020]. Creating a new transaction must solve a cryptographic hash puzzle similar to a traditional blockchain. Subsequently, it is necessary to validate the two previous transactions so that the new transaction can finally be added.

2.4 IoT Domains

IoT encompasses several domains where connected devices collect and analyze data in real-time. In healthcare, IoT enables remote patient monitoring and the use of wearable devices. Smart cities use IoT to manage traffic, public lighting, and security. In agriculture, the use of IoT facilitates crop monitoring and the automation of irrigation systems.

Smart homes leverage IoT for home automation and energy monitoring. The industry utilizes IoT for predictive maintenance and the automation of industrial processes. In

transport and logistics, the IoT enables vehicle tracking and route optimisation. The Internet of Things improves intelligent electrical networks and consumption measurements in the energy sector. Smart commerce uses IoT for inventory management and consumer behaviour analysis.

2.5 Genetic Algorithms

Genetic algorithms (GAs) are a class of optimisation methods inspired by the principles of natural selection and evolution [Holland, 1975]. They work by evolving a population of candidate solutions through iterative processes, such as selection, crossover, and mutation, to seek near-optimal solutions to complex problems. In many applications, they help overcome the limitations of traditional optimisation techniques by maintaining diversity in the population and adapting to changes in the problem structure over time.

There are many variations in genetic algorithms, including the Simple Genetic Algorithm (SGA) [Vose, 1999] and the Adaptive Genetic Algorithm (AGA) [Jakobović and Golub, 1999]. The Simple Genetic Algorithm employs fixed parameters for mutation and crossover rates throughout its evolution, making it straightforward to implement and understand. However, it can lead to premature convergence if the parameters are not well-tuned.

The Adaptive Genetic Algorithm dynamically adjusts these rates based on the population's current diversity, thus increasing its exploration capabilities when diversity is low and intensifying exploration when diversity is high. The adaptive behaviour of AGA avoids local optima and achieves more robust convergence. Avoiding local optima is crucial because if the algorithm stabilizes on a suboptimal solution, it may not explore other regions of the search space containing the globally optimal solution.

3 Strategy for selecting consensus algorithms using genetic algorithms

This section presents the main contribution of this paper, which is a strategy for choosing the most suitable combinations of consensus algorithms for use in IoT domains.

3.1 Strategy Overview

The strategy proposed in this paper for choosing consensus algorithms aims to improve the choice of these algorithms for IoT environments. The strategy consists of four main activities, which are represented in Figure 2.

Figure 2. Strategy for choosing consensus algorithm combinations.

The first activity of the strategy, indicated in Figure 2, is to develop a decision matrix containing the candidate consensus algorithms to be used in the consensus process. This matrix contains five columns, where the input values are evaluated according to the characteristics of each algorithm, such as time to create new blocks, scalability, latency, throughput, and power consumption. Each value represents how efficient the algorithm is in that aspect; for example, a consensus algorithm with a high value in the energy consumption field is usually efficient during its executions.

The second activity of the strategy, indicated in Figure 2, consists of developing a matrix containing the main IoT domains. This matrix also contains five columns, considering the same previous characteristics: time to create new blocks, scalability, latency, throughput, and power consumption. However, in this matrix, the input values represent each IoT domain that needs a particular characteristic. For example, if the Smart City domain has a high value in the energy consumption field, it needs a consensus algorithm that is highly efficient in this regard to meet its needs.

The third activity, indicated in Figure 2, consists of executing the genetic algorithms AGA and SGA, which will receive as input the matrices defined in the previous activities and, based on the values defined in each matrix, the GAs will make their suggestions for consensus algorithm combinations. The goal of executing this strategy is to define the algorithm combination that best meets the needs of each IoT domain. However, the combination indicated by the GA is not always optimal, and inconsistencies may occur due to the definition of the input data and the GA execution process.

Finally, in the fourth activity, indicated in Figure 2, the consensus algorithm combination suggested by the genetic algorithm will be executed; with it, a new block can be created, and the data can be recorded in the OmniBlock Blockchain, developed with support for running multiple consensus algorithms.

The following sections present the details of each activity defined in Figure 2. The design and development of the OmniBlock Blockchain are presented, along with the genetic algorithms and consensus algorithm combinations provided by the proposed strategy.

3.2 Blockchain Implementation

The OmniBlock Blockchain was developed to support the consensus algorithm selection strategy presented in this paper. This blockchain utilizes Directed Acyclic Graphs (DAGs) and integrates multiple consensus algorithms to mitigate performance challenges associated with consuming computational resources, such as CPU and RAM.

The DAG provides a more flexible and agile structure for validating transactions, allowing parallelisation and reducing latency [Wang *et al*., 2021]. Integrating multiple consensus algorithms aims to strike a balance between the speed of creating new blocks and the efficient use of computational resources, thereby enhancing the consensus process's performance.

Figure 3 shows the overview of the OmniBlock used in this paper. A user, A, wants to record the data of the IoT sensor. A new data block, represented by the number 6, must be created to record this data. At this point, the blockchain will use a combination of consensus algorithms. This choice depends on the IoT domain from which the data originates (e.g., Smart Home, Smart City).

Inspired by Tangle [Silvano and Marcelino, 2020], the DAG-based architecture validates new blocks through two previous nodes, e.g., in Figure 3, node 6, upon creation, must validate nodes 4 and 5 before being confirmed. The implementation was carried out using Docker containers to implement this blockchain, where each container represents a DAG node that stores a copy of all the recorded data.

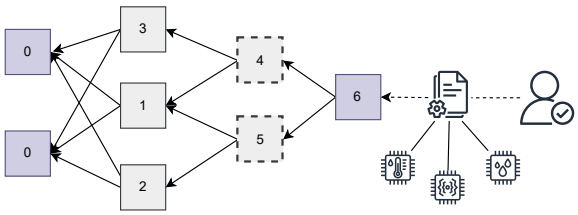

Figure 3. Overview of the OmniBlock Blockchain.

The created Blockchain supports ten consensus algorithms, which are Proof of Work (PoW) [Nakamoto, 2008], Proof of Stake (PoS) [Kiayias *et al*., 2017], Proof of Authority (PoA) [De Angelis *et al*., 2018], Practical Byzantine Fault Tolerance (PBFT) [Castro *et al*., 1999], Raft Consensus Algorithm (Raft) [Ongaro and Ousterhout, 2015], Proof of Luck (PoL) [Milutinovic *et al*., 2016], Proof of Burn (PoB) [P4Titan, 2014], Proof of Importance (PoI) [Bach *et al*., 2018], Delegated Proof of Stake (dPoS) [Larimer, 2014] and Proof of Elapsed Time (PoET) [Salimitari *et al*., 2020].

The consensus algorithms implemented in OmniBlock were selected based on their specific characteristics and suitability for IoT requirements. For example, PoW was chosen for its security against modifications, while PBFT was chosen for its fault tolerance in decentralised networks. In summary, the main criterion for choosing consensus algorithms was 'performance in IoT environments'. This criterion encompasses the ability of the algorithms to meet the specific needs of IoT devices, including efficient power consumption, low latency, minimal block creation time, and robust security.

OmniBlock divides consensus into four sequential steps: (1) validator selection, (2) block creation, (3) block validation, and (4) DAG insertion, and assigns different algorithms to each step based on its characteristics. For example, a lightweight and fast algorithm (such as PoA or PoS) can be used to select a validator, while a more secure one (such as PoW or PBFT) can be employed for block creation and validation.

Adopting a single algorithm for all steps would compromise the balance between performance and security. Using only PoW offers high security, but at the expense of high block creation times and CPU consumption. Using only PoS or PoA results in low latency and reduced resource usage, but it also reduces attack resistance. Therefore, combining two algorithms in each combination allows for the best of what each offers.

3.3 Genetic Algorithm

This section describes the genetic algorithms used in this work: the Simple Genetic Algorithm (SGA) and the Adaptive Genetic Algorithm (AGA). This work applied the standard concepts of SGA and AGA algorithms to the blockchain in the context of the IoT.

The use of a genetic algorithm to select consensus combinations is justified by its ability to explore a space of discrete and multidimensional solutions. In this case, combinations of two algorithms out of ten available were evaluated according to multiple criteria (block creation time, CPU/RAM consumption, and network traffic).

Genetic algorithms operate with candidate populations, crossovers, and mutations that promote diversity in the search. Furthermore, they enable adaptive adjustment of parameters (such as crossover and mutation rates), thereby improving performance in IoT environments.

Simple Genetic Algorithm

SGA is a classic optimisation algorithm that simulates natural evolution. In this algorithm, a population of possible solutions is improved through selection, crossover, and mutation to converge to an approximate solution. Algorithm 1 presents the pseudocode of the SGA proposed in this work.

Algorithm 1 Simple Genetic Algorithm

Require: Domain weights W, population size N, number of generations G, mutation rate μ, crossover rate γ, allowed set for position x (A), full set for position y ($\mathcal{A}$) with $x \in A$, $y \in \mathcal{A}$, and $x \neq y$
Ensure: Best combination (chromosome) C_{best}
1: $P \leftarrow \text{GeneratePopulation}(N)$ ▷ Each chromosome is a pair $[x, y]$ with $x \in A$ and $y \in \mathcal{A}$, ensuring $x \neq y$
2: **for** $g \leftarrow 1$ **to** G **do**
3: **for** each chromosome $c \in P$ **do**
4: Compute fitness $f(c, W)$ ▷ Fitness is computed as the weighted mean of the scores for both algorithms in c
5: **end for**
6: $P_{sel} \leftarrow \text{SelectBest}(P, N/2)$ ▷ Select the top $N/2$ chromosomes with highest fitness
7: $P \leftarrow \text{Reproduce}(P_{sel}, \gamma, \mu)$ ▷ Apply crossover and mutation to generate a new population, ensuring that in each offspring $x \neq y$
8: **end for**
9: $C_{best} \leftarrow \arg\max_{c \in P} f(c, W)$
10: **return** C_{best}

The expected output of the SGA is a pair of consensus algorithms $[x, y]$, which represents a combination of consensus algorithms, where x is different from y. To achieve this, the input parameters used are:

- W: vector of weights extracted from the decision matrix corresponding to the IoT domain;
- N: population size;
- G: total number of generations;
- μ: mutation rate, which is the probability of a gene (i.e., one of the algorithms present in the pair) undergoing mutation;
- γ: crossover rate, or crossover rate, corresponds to the probability of performing a crossover when generating new descendants;

- X: consists of a subset of consensus algorithms (PoA, PoS, PoL, PoI, PoB, dPoS, and PoET) that can be chosen for the first chromosome position;
- Y: complete set of all consensus algorithms available for the combination; and
- Constraint: the chromosome positions x and y must always have different algorithms ($x \neq y$).

The algorithm execution begins by generating an initial population P containing N chromosomes (Line 10). Each chromosome corresponds to a pair $[x, y]$, where x is chosen randomly from the set of allowed algorithms X, and y is chosen randomly from the total set of algorithms Y. A check ensures that if x and y are identical, y will be selected again until they are different. The objective is to find the best combination for each IoT domain defined in W.

For each chromosome $c \in P$, the fitness function $f(c, W)$ is calculated. The fitness function combines the weights (provided for each consensus algorithm) using a weighted average based on the weight vector W. This function evaluates whether a specific combination of consensus algorithms is suitable for meeting the requirements of the IoT domain under consideration.

Next, the SGA algorithm selects the first $N/2$ chromosomes from the current population based on their fitness values. This selection ensures that only the best combinations are transferred to the next phase, reproduction.

In the next phase, the selected chromosome pairs undergo crossover. The crossover operation creates offspring by taking the first gene from one parent and the second gene from the other, ensuring that adjustments are made if the resulting pair has duplicate algorithms.

Each offspring is then subjected to mutation, where a gene (position x or y) can be replaced by an algorithm chosen randomly from the respective set (for x, from X; for y, from Y), again ensuring that $x \neq y$.

Lines 2-8 are repeated for a number of G generations. With each new generation, the population evolves ideally toward better combinations (higher fitness).

After executing G generations, the SGA algorithm selects the chromosome with the highest fitness value. This selection will be indicated as the best consensus algorithm combination for the IoT domain under consideration. Finally, the value of C_{best} is returned.

Adaptive Genetic Algorithm

AGA differs from SGA in that it dynamically adjusts crossover and mutation rates based on population performance and diversity. This adjustment allows the search process to adapt to the problem characteristics and avoid premature convergence.

In the implementation of AGA, used in this work, the mutation and crossover rates were set to 0.05 and 0.5, respectively. According to Lobo *et al*. [2007], a mutation rate around 0.05 is considered moderate, as it ensures the introduction of variations without destroying the good characteristics already evolved. A crossover rate of approximately 0.5 is often used to promote intensive recombination of genetic traits. Algorithm 2 represents the AGA used in this work.

Algorithm 2 Adaptive Genetic Algorithm

Require: Domain weights W, population size N, number of generations G, initial mutation rate μ, initial crossover rate γ, allowed set for position x (A), full set for position y ($\mathcal{A}$) with $x \in A$, $y \in \mathcal{A}$, and $x \neq y$
Ensure: Best combination (chromosome) C_{best}
1: $P \leftarrow \text{GeneratePopulation}(N)$ ▷ Each chromosome is a pair $[x, y]$, where x is chosen from A and y from $\mathcal{A}$, ensuring $x \neq y$
2: **for** $g \leftarrow 1$ **to** G **do**
3: **for** each chromosome $c \in P$ **do**
4: Compute fitness $f(c, W)$
5: **end for**
6: $D \leftarrow \text{ComputeDiversity}(P)$
7: Adjust μ and γ based on D
8: $P_{sel} \leftarrow \text{SelectBest}(P, N/2)$
9: $P \leftarrow \text{Reproduce}(P_{sel}, \gamma, \mu)$ ▷ Perform crossover and mutation, ensuring $x \neq y$ in each offspring
10: **end for**
11: $C_{best} \leftarrow \arg\max_{c \in P} f(c, W)$
12: **return** C_{best}

Also, the expected result in AGA is a pair of algorithms $[x, y]$. For this, the input parameters used are:

- W: A vector of weights corresponding to the requirements of the specific IoT domain.
- N: The population size.
- G: The number of generations.
- μ (Mutation Rate): The initial probability of mutating a gene.
- γ (Crossover Rate): The initial probability of performing crossover.
- A (Allowed Set for x): A restricted set of consensus algorithms (e.g., PoA, PoS, PoL, PoI, PoB, dPoS, PoET) that can be chosen for the first gene.
- $\mathcal{A}$ (Full Set for y): The complete set of consensus algorithms for the second gene.
- Constraint: Ensure that in every chromosome $[x, y]$, $x \neq y$.

The algorithm starts by generating an initial population P of N chromosomes. Each chromosome is a pair $[x, y]$, where x is randomly chosen from the allowed set A and y is randomly chosen from the complete set $\mathcal{A}$, with the condition that $x \neq y$.

For each chromosome $c \in P$, the fitness is computed $f(c, W)$, which is typically defined as the weighted average of the scores associated with the two algorithms in the chromosome.

Then, the diversity D of the current population is computed (i.e., by measuring the number of unique fitness values). Based on D, the mutation rate μ and the crossover rate γ are adjusted. If the diversity is low, increase μ and decrease γ to promote exploration. If diversity is high, decrease μ and increase γ to refine promising solutions.

Next, the top $N/2$ chromosomes P are selected based on their fitness to form a mating pool P_{sel}. A new population P is generated by crossover and mutation on the chosen parents. The crossover function swaps the second gene between the

parents, and the mutation function changes one gene, ensuring that the new gene does not duplicate the other gene on the chromosome.

Steps 2 to 5 are repeated for G generations. Finally, the chromosome C_{best} with the highest fitness value is returned as the best consensus combination.

3.4 Combinations Selection

After implementing the OmniBlock Blockchain and its consensus algorithms, and also implementing the genetic algorithms, the combination selection strategy introduced in Section 3 was executed. The strategy's objective is to indicate the best combination of algorithms for an IoT scenario or domain. The proposed strategy follows the four activities described in Figure 2.

The following sections outline each step of the selection strategy.

3.4.1 Definition of the weights of the consensus algorithm matrix

The first activity of the strategy is to define a matrix containing all the algorithms supported by the blockchain, as shown in Table 1. The columns of this matrix consist of weights numbered from 1 to 10, representing the characteristics of each algorithm. Five characteristics were considered to define a blockchain for IoT: Time to create a new block, scalability, latency, throughput, and power consumption.

In blockchains for IoT, the time to create a new block is the interval between the submission of a transaction and its effective inclusion in the ledger; scalability is the network's ability to maintain performance and reliability as the number of nodes or transactions grows; latency refers to the delay between the submission of a transaction and its confirmation by validators; throughput is the number of transactions or blocks processed per second by the network; and energy consumption encompasses the use of computing resources (CPU, RAM, and battery) required to execute the consensus protocol on power-constrained devices.

To define the weights for the first column, *Time to create new blocks*, the values obtained in the performance evaluation conducted in de Morais *et al*. [2023a], where the times of each consensus algorithm were measured. Weights close to 10 were assigned to the algorithms that created the fastest blocks, for example, PBFT and Raft, while weights close to 1 were assigned to the slower algorithms, such as PoW and PoET.

To define the weights of the remaining columns, de Morais *et al*. [2023b] was used, which analyzes the main consensus algorithms presented in the literature.

In Table 6 of de Morais *et al*. [2023b], *Scalability*, *Latency*, *Throughput* and *Energy Consumption* are qualitatively classified as *High*, *Medium* and *Low*. To define the weights, an equivalence was performed: the algorithms with concept *High* were divided into *Very High* (10-8) and *High* (6-7). *Medium* (5). Those with concept *Low* were divided into *Low* (4-3) and *Very Low* (2-1).

3.4.2 Defining the weights of the IoT domain matrix

The second activity creates a decision matrix containing the main IoT domains, as shown in Table 2. This matrix has the same columns as the consensus matrix (time to create a new block, scalability, latency, throughput, and power consumption). However, the weights represent the degree to which each domain requires that characteristic. For example, a weight of 10 in the "scalability" column indicates that the domain requires a consensus algorithm that supports high scalability. A weight of 1 represents that scalability is not as relevant in that domain, and so on for the other columns.

To define the weights of this matrix, several approaches were analysed in each of the domains, and empirical analyses allowed the identification of how each of the dimensions of the columns in Table 2 is treated in the literature.

Eight domains were considered, and a relevant paper was chosen related to each one of them: Smart Home [Lee *et al*., 2020]; Healthcare [Zaabar *et al*., 2021]; Smart Agriculture [Lei *et al*., 2022]; Industrial IoT [Zubaydi *et al*., 2023]; Smart Cities [Paul *et al*., 2021]; Smart Logistics [Ugochukwu *et al*., 2022]; Smart Energy [Khan *et al*., 2023]; and Internet of Vehicles [Javed *et al*., 2020].

The values in Table 2 were defined on 10 levels, representing the need for each characteristic in each domain. The value is 10 for critical need, 8 or 9 for very high need, 6 or 7 for high need, 5 for medium need, 3 or 4 for low need, and 1 or 2 for very low need.

The decision matrices were chosen on a scale of 1 to 10 to provide sufficient granularity for the genetic algorithm to optimise combinations. With ten weight levels, subtle variations in the characteristics of each consensus combination (such as block time, latency, or energy consumption) can be more accurately represented, enriching the search space and facilitating the algorithm's convergence to balanced solutions.

Furthermore, this additional granularity helps avoid sudden jumps in fitness assessment, promoting fine-tuning and a selection process more sensitive to performance differences between combinations.

3.4.3 Execution of the genetic algorithms to select the combinations

After defining the weights of the consensus algorithm matrix and the IoT domain matrix, the third activity of the strategy involves executing the genetic algorithms to determine which combination is most suitable for each domain. For this, the matrices of Table 1 and Table 2 were used as input for the SGA and AGA.

The strategy of this work is to combine pairs of consensus algorithms (e.g., PoX and PoY), where the first algorithm of the pair is used in stage 1 of the consensus, selection of the validator node, and the second algorithm of the pair will be used in the following stages: creation, validation, and insertion of the new block into the blockchain.

After performing the executions, the indications of the most suitable consensus algorithm combinations for each IoT domain are obtained. Table 3 presents the results obtained in this stage of the strategy for choosing the combinations.

Table 1. Characteristics analysed in each consensus algorithm.

	Time to create new blocks	Scalability	Latency	Throughput	Power consumption
PoW	2	8	4	5	6
PoS	3	7	5	6	9
PoA	5	8	4	2	4
PBFT	8	6	1	7	10
Raft	9	6	4	3	9
PoET	1	5	2	5	2
PoL	3	2	5	9	7
PoB	2	7	4	6	9
PoI	5	6	2	1	4
dPoS	4	3	1	5	6

Table 2. Characteristics analysed in each IoT domain.

	Time to create new blocks	Scalability	Latency	Throughput	Power consumption
Smart Home	6	3	7	5	9
Healthcare	9	2	5	3	7
Smart Agriculture	3	9	3	6	8
Industrial IoT	8	8	4	7	1
Smart Cities	9	7	5	7	8
Smart Logistics	7	3	9	4	3
Smart Energy	8	8	3	6	2
Internet of Vehicles	9	1	4	3	9

Table 3. Combination according to genetic algorithms

	SGA	AGA
Smart Home	PoS and PoW	PoS and PoL
Healthcare	PoS and Raft	PoS and PoL
Smart Agriculture	PoS and PoB	PoS and PoW
Industrial IoT	PoA and PoW	PoET and PoA
Smart Cities	PoB and PBFT	PoI and Raft
Smart Logistics	PoET and PoW	PoA and PoL
Smart Energy	PoA and PoW	PoI and Raft
Internet of Vehicles	PoL and PBFT	PoI and PBFT

As can be seen in Table 3, the combinations suggested by SGA and AGA are different; this is because in AGA, the mutation and crossover rates vary and are dynamically adjusted during the execution of the algorithm. This dynamic adjustment enables AGA to explore more possible combinations and present results that differ from those of SGA. In the following sections, the results of the genetic algorithms will be evaluated to compare which presented more efficient results.

Considering the four stages of the consensus process: (1) validator selection, (2) block creation, (3) block validation, and (4) blockchain insertion. Different algorithms are assigned to each stage based on their characteristics. For example, algorithms that are efficient in validator selection, such as PoA and Raft, are used in stage 1; other algorithms, efficient in block creation and validation, such as PoW or PBFT, are used in subsequent stages.

3.5 Consensus Algorithm Combinations

The consensus algorithms were selected based on their specific characteristics and suitability to IoT requirements. For example, PoA was chosen due to its reduced transaction confirmation time [Wang *et al*., 2022], PoW for its security [Abellán Álvarez *et al*., 2024], and PoS for its power efficiency [Abellán Álvarez *et al*., 2024].

PoB was chosen because it presents low resource consumption for block validation [Karantias *et al*., 2020]. PoI was selected for its ability to reduce latency, increasing the speed of block confirmation [Xiao *et al*., 2021]. dPoS promotes more democratic and efficient block management [Larimer, 2014]. PoET seeks to minimise power consumption [Bowman *et al*., 2021].

PBFT was selected for its Byzantine fault tolerance capabilities and security in distributed networks [Meshcheryakov *et al*., 2021], while Raft was selected for its speed in reaching consensus in distributed environments [Fu *et al*., 2021]. The primary criterion for selecting the algorithms was ”performance in IoT environments.” This criterion encompasses the ability of the algorithms to meet the specific needs of IoT devices, including energy efficiency, low latency, and short block creation times.

The following sections describe how the combinations proposed by genetic algorithms work, as shown in the Table 3.

3.5.1 PoS and PoL

This combination combines two algorithms that consume little computational resources. While PoS promotes the economical use of participants’ coins, PoL provides randomness and fairness to the consensus process.

In this process, PoS is initially used to select the node with the highest participation, making it eligible to serve as a validator. This node then uses PoL to generate a new block, basing its creation on a luck mechanism that equitably distributes the opportunity to create blocks.

This combination's advantage lies in reducing power consumption, as neither algorithm requires intensive computation, and in promoting a fairer and more evenly distributed consensus among network nodes.

This combination may present some drawbacks, as PoS tends to favor participants with larger coin holdings, leading to the centralization of power and hindering the participation of new nodes. Furthermore, while PoL introduces luck into the selection process, the random element can compromise predictability and control over block creation times, which can be problematic in applications with strict latency or synchronisation requirements.

3.5.2 PoS and Raft

In this combination, PoS was chosen for its efficiency in consuming computational resources [Abellán Álvarez *et al*., 2024]. Raft was chosen to handle failures and improve data consistency and integrity on the blockchain [Fu *et al*., 2021].

In the first step of consensus, the initial node is chosen among those with the largest number of coins (stakes), according to the PoS algorithm. In the following steps, the selected node executes the Raft algorithm to create a new block, validate it, and add it to the blockchain.

Raft offers a consensus process that prioritises consistency and availability. This strategy contributes to the integrity of the blockchain, ensuring that all nodes are synchronized, even in adverse conditions, such as failures or the presence of malicious nodes.

A large number of nodes in the network can significantly impact the Raft algorithm's performance, which is a particular concern in IoT environments that require high scalability. In the Raft consensus process, the other network participants must vote on the new block before it is included in the blockchain. Therefore, if the number of participants is large, the block confirmation time may be high.

3.5.3 PoS and PoB

In this combination, PoS is adopted for its ability to select validators based on the amount of tokens, promoting honest commitment from participants. PoB is used because requiring the burning of tokens makes the production of new blocks more expensive for malicious agents.

PoS is used in the initial consensus stage to choose the node with the highest stake, which becomes the validator. This selected node uses PoB to generate a new block, burning a pre-established amount of coins as the creation cost.

This combination reduces energy consumption and makes block creation economically costly for attackers, increasing the security and integrity of the blockchain.

The main limitation of the PoS algorithm is the potential for power concentration among participants with large stakes, which can compromise decentralisation. The PoB algorithm, on the other hand, requires the permanent burning of tokens, which can reduce liquidity and discourage continued participation, especially in systems with a limited coin supply. Furthermore, the coin-burning model can be seen as economically inefficient in the long run.

3.5.4 PoS and PoW

PoS was chosen for its power efficiency and the ability to create new blocks based on the amount of currency held by the participants. PoW was selected based on its resistance to modification, which improves the immutability and integrity of the blockchain [Abellán Álvarez *et al*., 2024].

The validation node is selected from those offering the largest coins (stakes) in the first consensus step. In the second step, the selected node executes PoW to create a new block, which is then added to the blockchain.

In this combination, the validation node selection criterion is through financial participation due to using PoS. This approach reduces PoW's need for intense computational activity, making block creation more efficient and sustainable.

The key point to consider in this combination is that the PoW algorithm can consume a significant amount of energy and computational resources, which is particularly critical in IoT environments where devices have limited resources.

3.5.5 PoA and PoW

The combination of Proof of Authority (PoA) and Proof of Work (PoW) combines the low latency of PoA [Wang *et al*., 2022] with the security against modifications of PoW [Abellán Álvarez *et al*., 2024].

PoA is used in the first step of consensus to select the validation node with the highest authority; the chosen node performs the second step of consensus, using PoW to create a block and incorporate it into the DAG in the last step.

This combination combines the results of PoA and PoW into a more efficient block creation process. It speeds up transaction confirmation times compared to solutions that employ only one of these algorithms separately.

However, this combination can present some drawbacks. PoA's efficiency can be compromised if malicious actors gain authority, while PoW can increase battery and computational resource consumption.

3.5.6 PoET and PoA

This combination combines Poet's low power consumption with Poa's reliability. While PoET uses a random waiting time mechanism to choose the node that will act as a validator fairly, PoA ensures that this node, recognised by its authority in the network, is responsible for creating the new block.

In the first stage of consensus, PoET is applied to determine which node, after waiting a random waiting time, becomes eligible to act as a validator. After this selection, the chosen node uses PoA to generate the block, using its identity and reputation to validate and incorporate the new block into the blockchain.

The main advantage of this combination is the reduction in computational resource consumption, as it utilises two algorithms known for their efficiency in this regard, while maintaining block creation speed.

The PoET-PoA combination may have some limitations. PoA tends to centralise authority in a few nodes, which can compromise the system's decentralisation and create points of failure. PoET's randomness can lead to unpredictable

variations in block creation times, which can affect blockchain performance in high-demand scenarios.

3.5.7 PoB and PBFT

PBFT provides a fast and fault-tolerant consensus process in distributed environments. PoB incentivises validators to engage in honest behaviour through token burning.

PoB generates a block candidate in the first consensus stage, where the node that wants to create the block must burn a predetermined amount of tokens. This block then undergoes a validation process using PBFT, in which a group of participating nodes performs verifications until consensus is reached on the block's integrity, before it is included in the blockchain.

The advantage of this combination lies in improved security, as PoB discourages malicious behaviour by imposing an economic cost on block creation. At the same time, PBFT ensures fast and reliable validation even in the presence of failures or compromised nodes [Khan *et al*., 2022].

However, the cost associated with token burning can represent a barrier to participation for nodes without financial resources. At the same time, the PBFT algorithm can hinder scalability, increasing latency and consensus complexity in environments with a large number of nodes. Therefore, the PoB-PBFT combination may be less suitable for many devices and high-traffic scenarios, where scalability and minimizing computational costs are top priorities.

3.5.8 PoI and Raft

This combination combines validator selection based on the PoI's importance (which takes into account metrics such as stake, activity, and interaction on the network) with Raft's distributed consensus and fault tolerance.

In the consensus process, PoI is used in the initial stage to identify and select the node with the highest importance, assigning it the responsibility of validating and proposing the new block. The chosen node then uses Raft to lead the consensus, consistently generating, validating, and inserting blocks into the blockchain, even in the face of potential failures.

This combination results in efficient, metrics-driven selection of importance (PoI) with an agile and reliable consensus algorithm (Raft). It provides a gain in decentralisation and scalability, essential features for IoT environments that require efficient data management.

While the PoI-Raft combination offers strengths, such as faster block confirmation, it also has limitations. Verifying node importance can be complex and susceptible to manipulation, thereby compromising the fairness of the selection process. Furthermore, while efficient in moderate-sized networks, Raft can face security issues and increased latency when applied to large IoT networks due to the communication overhead between nodes [Howard and Mortier, 2020].

3.5.9 PoET and PoW

Proof of Elapsed Time (PoET) distributes the opportunity to create blocks fairly through random waiting times. Proof of Work (PoW) provides security and resistance to various types of attacks.

In the consensus process, PoET is initially used to determine the node with the shortest waiting time, making it eligible to propose a new block. This winning node then uses PoW to validate and consolidate the new block.

This combination aims to balance power efficiency and security by combining Poet's lightweight and fair approach with Pow's attack resistance.

The reliance on specific hardware for proper PoET implementation may limit its applicability in resource-constrained IoT environments. Furthermore, even with the potential reduction in resource consumption through PoET, PoW still imposes high computational resource requirements, which may affect scalability and operational efficiency in high-demand networks.

3.5.10 PoA and PoL

This combination combines the rapid selection of validators based on PoA's authority with the fairness of PoL's luck mechanism. PoA is used to identify trustworthy nodes, while PoL uses a random method that fairly distributes the opportunity to create blocks.

In the consensus process, PoA is initially applied to select a few nodes with the highest authority and reputation in the network, qualifying them as validators. Then, PoL is used to determine, based on randomness, which node will actually create the new block in the blockchain.

The main advantage of this combination is that it avoids overloading the most reputable nodes during block creation. By using the PoA algorithm to choose a group of potential validators and PoL to distribute block creation opportunities fairly, the consensus process becomes less centralised and more resistant to attacks and manipulation.

Despite these advantages, the PoA-PoL combination can have disadvantages. For example, using PoA tends to concentrate validation authority in a few nodes, which can increase the risk of centralisation. Furthermore, the random component of PoL can lead to inconsistent wait times for block creation, which in turn impacts the system's predictability.

3.5.11 PoL and PBFT

In this combination, PoL uses a luck mechanism to distribute the opportunity to create blocks fairly, while PBFT ensures fast and fault-tolerant consensus among nodes, even in adverse environments.

In the first consensus stage, PoL is initially used to define the candidate node that will randomly create a new block. This node then uses PBFT to conduct block validation, allowing a group of trusted nodes to verify and reach consensus before including the block in the blockchain.

The main advantage of this combination lies in the combination of fairness and security: PoL is used for a fair distribution of block creation opportunities, and PBFT is used for resilient consensus.

This combination, while combining simplicity and fairness in node selection, can face limitations, such as the randomness of PoL. Although efficient, PoL can generate unpredictable response times, which compromise the predictability of the

network's response time. Because the PBFT algorithm relies heavily on communication between validator nodes, it suffers from scalability issues in large networks, resulting in increased latency and resource consumption.

3.5.12 PoI and PBFT

This combination combines importance-based validator selection (PoI), which evaluates the relevance of nodes based on metrics such as stake, activity, and network interaction, with the fault tolerance of PBFT. While PoI identifies nodes that demonstrate the highest honesty and performance, PBFT provides fast and secure consensus, even in the presence of malicious nodes.

In the first consensus step, PoI is initially applied to select the node with the highest importance, qualifying it as a candidate validator. This node then uses PBFT to validate the new block, ensuring that a group of trusted nodes reaches consensus before the block is added to the blockchain.

The main advantage of this combination is the combination of performance-based validator selection (PoI) with a resilient consensus algorithm (PBFT), which increases the security and decentralisation of the network Khan *et al*. [2022]. The combination of these algorithms may be suitable for Internet of Things environments that require high reliability and performance.

Despite offering a fault-tolerant approach, the PoI-PBFT combination has limitations. PoI relies on reputation indicators, which can be manipulated or may not accurately reflect the true relevance of nodes, affecting validator selection. Furthermore, PBFT, although efficient in smaller networks, faces scalability challenges and increased latency in environments with a large number of participants.

4 Evaluation

This section presents the experimental evaluation of the proposed strategy. The methodology used for the performance evaluation was based on Jain [1991].

4.1 Objectives, Parameters, and Metrics

The objective of this performance evaluation is to compare the performance and computational resource consumption of the consensus algorithm combinations suggested by genetic algorithms.

The primary components under study are various blockchain configurations and their combinations. The system consists of a linear and a DAG-based blockchain, both of which are implemented using Docker containers. Consensus algorithms were implemented in Node.js.

The experiments were conducted on a cluster assembled using Docker Swarm. Table 4 shows the machines' configuration in this cluster. A total of 9 machines were used, totalling 104 GB of RAM.

The primary service provided by the system involves creating and registering a new block on the blockchain, which is accomplished by a single user sending data to the network. The registered data was obtained in partnership with the Startup Semine[1], and consists of humidity and temperature data of the air and soil collected on an agricultural property.

Table 4. Docker Swarm Cluster Configuration

Quantity	Processor	RAM
4	Intel Core i5-3470	16 GB
2	Intel Core i5-5200	8 GB
2	Intel Core i3-4160	8 GB
1	Intel Core i3-3217	8 GB

Due to resource limitations, errors and failures will not be studied. Therefore, the study will be limited only to creating and correctly inserting blocks in the blockchain. The metrics chosen for this evaluation were:

- *Time to create new blocks*.
- *CPU usage*;
- *RAM usage*;
- *Network traffic*.

The system parameters that affect the time it takes to create new blocks on the blockchain are the CPU speed of the cluster hosting the implementation, the size of the message that will be stored in the block, and the bandwidth of the Docker Swarm network.

In a performance evaluation, factors and parameters that, when varied, will influence the system's performance are considered. In this evaluation, two factors were considered to obtain performance variations. The first factor was the blockchain configuration, which varied between linear and DAG-based.

Another factor considered was the consensus algorithms, which varied in each test to allow the estimation of performance values in each configuration. The consensus algorithms used were: PoW, PoS, PoA, PBFT, Raft, PoI, PoB, PoL, dPoS and PoET.

4.2 Project

The workload consists of a program sending data to be recorded in a new block and requesting the execution of consensus algorithm combinations. This program also monitors the time between sending the request and confirming the block's inclusion.

The CPU, RAM, and network traffic consumption metrics are collected from each of the machines in the cluster using the cAdvisor[2] tool and are monitored by Prometheus[3].

Each of the combinations performed for the experiments created a new block. The procedure was repeated 30 times to calculate the time needed to create new blocks and to obtain the average time.

In each execution, CPU, RAM, and network traffic consumption were also collected. The monitoring time for each experiment was 5 minutes, and 30 samples were collected for each metric, with an interval of 3 seconds between each.

[1] More information can be obtained at https://semine.ag/
[2] cAdvisor
[3] Prometheus

4.3 Results

The evaluations performed on each combination of the consensus algorithm are described below.

4.3.1 Linear Blockchain Performance

The first experiment implemented a traditional blockchain with linear blocks, each containing 1000 nodes and utilizing only a single consensus algorithm.

The algorithms chosen for this test were Proof of Work (PoW), Proof of Stake (PoS), Proof of Authority (PoA), Practical Byzantine Fault Tolerance (PBFT), Raft Consensus Algorithm (Raft), Proof of Luck (PoL), Proof of Burn (PoB), Proof of Importance (PoI), Delegated Proof of Stake (dPoS) and Proof of Elapsed Time (PoET).

These algorithms were chosen because they are widely recognized and utilized in real-world blockchain scenarios, such as Bitcoin and Ethereum.

This initial experiment aims to generate reference values that allow us to verify whether the performance of the DAG-based blockchain with algorithm combinations has improved compared to the traditional model.

A Docker container implements each of the blockchain nodes. Table 5 shows the result of this experiment and the values obtained for each metric. For example, for PoW, the average block creation time was close to 2.7 seconds, CPU usage reached 58%, and RAM usage reached 23.3 MB. Network traffic was 8,88 MB/s sent and 8,87 MB/s received.

Table 5. Performance evaluation of a linear blockchain

	Time to create new blocks (ms)	CPU (%)	RAM (MB)	Network traffic (sent/received MB/s)
PoW	2713,7	58%	23,3	8,88 / 8,87
PoS	1342,9	34%	28,4	12,8 / 11,4
PoA	1221,9	30%	40,8	11 / 9,09
PBFT	1232,1	26%	25,4	17,95 / 18,59
Raft	1102,6	27,8%	25,6	18,98 / 16,58
PoL	2242,7	48%	32,4	13,9 / 14,8
PoB	2336,9	39%	33,3	10,4 / 9,92
PoI	1295	37,8%	48,3	9,72 / 8,3
dPoS	1093,6	37,2%	34	11,2 / 10,1
PoET	2152,1	56%	36,4	13,1 / 14,4

4.3.2 Performance of the DAG-based blockchain

The second experiment used the OmniBlock, a DAG-based blockchain. For this experiment, the combinations presented in Table 3, which were suggested by the genetic algorithms, were evaluated. The results for each of the metrics are presented below.

Time to create new blocks

The first metric analysed was the time it takes to create new blocks. This metric is crucial in IoT-focused blockchain applications, as most smart devices and applications require rapid responses. Furthermore, in applications that handle user data, it is essential to record data quickly to prevent potential attacks and security breaches.

The time to create new blocks was compared by varying the combinations in a blockchain containing 1000 nodes. Figure 4 presents the results of this experiment.

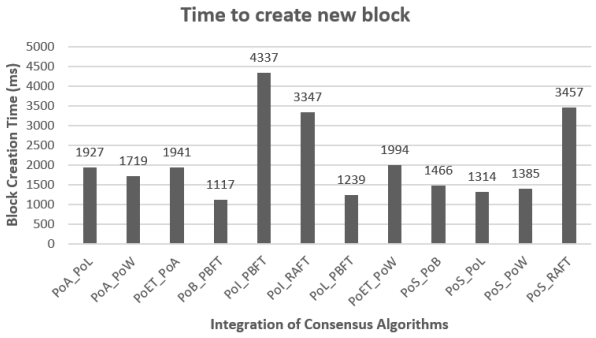

Figure 4. Time comparison to create new blocks using 1000 nodes in a DAG-based blockchain.

The combinations that created blocks faster were PoB + PBFT and PoL + PBFT, respectively. These combinations combine PBFT's agility and fault tolerance with PoB and PoL's security, which can be a promising solution for IoT applications.

CPU usage

The next metric to examine is CPU usage. In IoT applications, devices often have limited computing resources or depend on batteries. When a device performs intensive tasks, such as calculations or data processing, it requires more energy to power its components, including the CPU.

Therefore, when implementing blockchain-based solutions, it is crucial to assess the CPU consumption associated with creating new blocks to prevent overloading the device's processing capabilities.

To analyse this metric, only the central node was considered, the one that validated the new block during the test. Only the validator node showed significant CPU consumption values. Figure 5 shows the result of this evaluation, considering a blockchain with 1000 nodes.

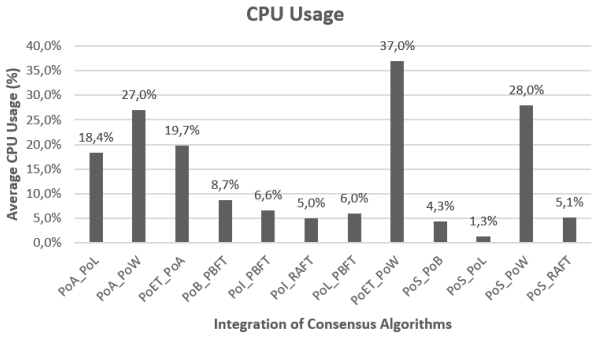

Figure 5. CPU usage comparison to create new blocks using 1000 nodes in a DAG-based blockchain.

The analysis reveals that the combinations that required the highest CPU consumption were those using the PoW algorithm. Proof of Work is an algorithm that demands more

computational resources due to its operation. However, its use offers a higher level of security, as the computational difficulty of breaking it is higher [Abellán Álvarez *et al*., 2024].

In contrast, the combinations that present the lowest CPU usage values are those that incorporate algorithms such as PBFT, PoS, and Raft. This indicates the viability of using these combinations in IoT environments.

RAM usage

Another metric to analyse is the average RAM usage. In IoT scenarios, managing memory resources is crucial to prevent overloads caused by insufficient memory availability.

The experiments were carried out for all combinations of the consensus algorithms, and the average was calculated considering all nodes present on the blockchain. Figure 6 shows the average memory consumption for the blockchain that contains 1000 nodes.

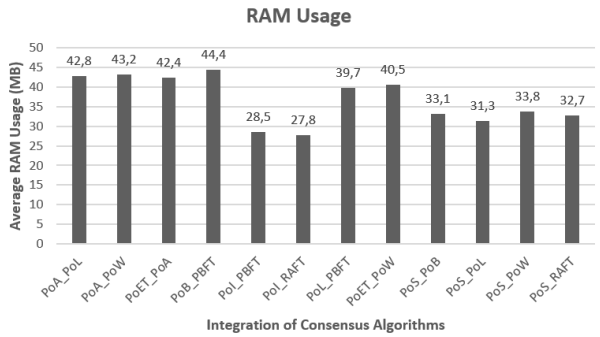

Figure 6. RAM usage comparison to create new blocks using 1000 nodes in a DAG-based blockchain.

As shown in the graph, PoB + PBFT had the highest RAM usage. However, PoI + PBFT had the lowest memory consumption.

Network Traffic

The last metric considered in the performance evaluation is network traffic. This metric is particularly relevant in blockchain solutions for IoT, as in scenarios with intense device data generation, it is essential to assess whether there is a risk of network overload, which can cause slowdowns in the blockchain.

As in previous experiments, all combinations were evaluated considering the blockchain with 1000 nodes. For each combination, the number of megabits per second (MB/s) sent and received was considered. Figure 7 presents the network traffic averages obtained in this evaluation.

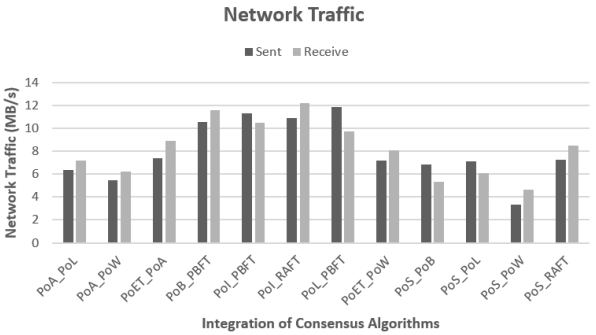

Figure 7. Comparison of average network traffic for the blockchain with 1000 nodes.

The combinations that recorded the highest values in this experiment used PBFT and Raft. These algorithms perform many checks on a new block before inserting it into the blockchain.

Although this evaluation focused on ideal conditions without failures or attacks, real IoT environments are subject to node failures, network partitions, and malicious behaviour. In these scenarios, lightweight consensus-based combinations (e.g., PoS + Raft) tend to recover quickly from failures but may lack robustness if multiple leaders fail simultaneously. Combinations that include Byzantine fault-tolerant protocols (such as PoS + PBFT) support a greater presence of malicious nodes but impose much greater communication overhead.

Other combinations (e.g., PoB + PBFT) reduce block creation latency but generate high network traffic due to PBFT's multiple rounds of messaging. These trade-offs underscore the need to strike a balance between fault-tolerant models and resource costs (CPU, memory, and network traffic) when selecting the most suitable combination for each application context.

4.3.3 Analysis and Discussion of the Combinations

After the performance evaluation, it is evident that the Blockchain OmniBlock, with its multiple combinations of consensus algorithms, outperforms the linear blockchain with isolated consensus algorithms (Table 5).

For example, the CPU consumption of the combinations was better than all the values of the algorithms presented in Table 5. The time to create new blocks was better in 62.5% of the times when the combinations were used, compared to the isolated algorithms. However, based on the experiments performed, it is not possible to determine whether the improvement was due to the combination of algorithms or the DAG structure used.

Some consensus algorithms (such as Raft) can offer faster block creation times in specific scenarios, but combinations offer benefits that neither can provide alone in many cases. Combining two algorithms, for example, one based on stakes and another based on Byzantine fault tolerance or proof-of-work, achieves a balance between speed, security, resource consumption, and resilience.

Based on the experiments' results and the metrics considered, some consensus algorithm combinations are more promising for data integrity in IoT. If the application requires fast responses and a low transaction confirmation time, PoB

and PBFT, as well as PoL and PBFT combinations, are the most suitable options, as they have the shortest time values for creating new blocks.

Some IoT applications utilize devices with limited processing capacity and require minimal CPU consumption. According to the performance evaluation, most of the suggested combinations present low CPU consumption (less than 10%) for this scenario. Only five combinations presented higher average CPU consumption values: PoET and PoW, PoS and PoW, PoA and PoW, PoET and PoA, and PoA and PoL. These combinations utilize algorithms known for their high computational resource consumption, such as PoW, making them potentially unsuitable for IoT scenarios.

In many cases, it is also necessary to keep RAM usage low to avoid overloading the blockchain nodes. According to Figure 6, in this evaluation, the combinations that presented the lowest average RAM consumption were PoI and Raft, and PoI and PBFT. This result is due to the fact that these combinations utilize algorithms known for efficiently utilizing computational resources, such as PBFT and Raft.

A blockchain solution for IoT must also support a large amount of data traveling through the network. Among the evaluated consensus algorithms, the combinations that supported the highest average network traffic were PoI and Raft. The highest network traffic values occur in combinations that use the PBFT and Raft algorithms.

The scalability of consensus algorithm combinations is crucial to meeting the growing demand for transactions in IoT environments. Combinations such as PoI and Raft and PoB and PBFT offer interesting solutions. PoI and Raft provide horizontal scalability, enabling the network to expand with additional nodes due to Raft's efficiency in achieving distributed consensus. The PoB and PBFT combination offers scalability with improved power efficiency and reduced computational resource usage, making it ideal for scenarios involving large volumes of data.

However, PoW combinations tend to be less scalable due to high resource consumption, making them more appropriate for scenarios where security is a priority over performance and growth. The scalability challenge in IoT blockchain lies in balancing security, efficiency, and low resource consumption to expand the network without compromising its performance.

4.4 Comparison of Genetic Algorithms

Due to variations in mutation and crossover rates, the SGA and AGA genetic algorithms present different combination suggestions, as shown in Table 3. Therefore, based on the performance evaluation of each combination, it is possible to compare and analyze which genetic algorithm presented more efficient results.

Table 6 compares the suggestions of the genetic algorithms. According to the table, AGA was more efficient in 75% of the suggestions presented.

For this analysis, the values of each metric were compared for each of the combinations provided by AGA and SGA, respectively. For example, for the Smart Home IoT domain, the combination suggested by SGA was PoS and PoW, and the suggestion by AGA was PoS and PoL. Based on the performance evaluation, the PoS and PoL combination proved to be more efficient in the four metrics considered: block creation time, CPU and RAM usage, and network traffic.

Table 6. Comparison of Genetic Algorithms

	SGA	AGA
Smart Home		✓
Healthcare		✓
Smart Agriculture	✓	
Industrial IoT		✓
Smart Cities		✓
Smart Logistics		✓
Smart Energy		✓
Internet of Vehicles	✓	

The Simple Genetic Algorithm (SGA) yielded better results in only two IoT domains: Smart Agriculture and Internet of Vehicles. In agriculture, for example, the SGA suggestion was PoS and PoB, which was more efficient in the metrics of CPU and RAM usage and network traffic, compared to the AGA suggestion, which was PoS and PoW, which was more efficient only in the metric of block creation time.

5 Related Work

According to Khan *et al*. [2022], the blockchain fundamental properties, such as decentralisation, security, and immutability, are promising for IoT applications. However, traditional consensus algorithms are designed for extensive computing and communication environments, which are unsuitable for resource-constrained IoT devices. The authors developed a CONIoT ontology to systematically classify and analyse the main consensus algorithms, evaluating their applicability in the IoT. The paper discusses the characteristics of consensus algorithms, such as PoW, PoS, and PBFT, among others, and their efficiencies in terms of power consumption, latency, and processing capacity.

Rasolroveicy and Fokaefs [2020] proposes using blockchain to improve security and efficient consumption of computational resources in IoT. The authors develop a self-adaptive strategy that dynamically selects the most suitable consensus algorithm for IoT systems based on the MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge) architecture. Dynamic reconfiguration enables the adjustment of consensus algorithm properties, such as the number of validation nodes and validation time, to optimize system performance. Experimental results demonstrate that different consensus algorithms, such as PoW, Proof of Elapsed Time (PoET), PBFT, and Raft, exhibit significant performance variations as the workload changes.

Tapwal *et al*. [2021] proposes a dynamic blockchain called A-Blocks for recording data from the Industrial Internet of Things (IIoT). According to the authors, traditional blockchain solutions use a single consensus algorithm, which may be unsuitable for the diverse and dynamic nature of IIoT data. A-Blocks dynamically selects the most appropriate consensus algorithm based on the attributes of the incoming data. Its operation occurs in two phases: categorising data into groups based on their characteristics (such as variability, sampling rate, and range of values) and selecting the appropriate consensus algorithm (PoW, PoS, or PBFT) for each data group.

Van Meerten *et al*. [2023] proposes a testing methodology for blockchain systems, using a case study on the XRP Ledger of the Ripple blockchain. The authors explore different message delivery orders of the consensus protocol and design a genetic algorithm to guide the generation of random test cases, thereby discovering concurrency bugs more efficiently. The work utilizes genetic algorithms to automate the generation of test cases during performance evaluation.

Aghroud *et al*. [2023] analyzes the challenges of choosing consensus algorithms for IoT networks. The authors develop a strategy to select the most suitable consensus algorithms for IoT. Algorithms must meet four criteria: low computing power, due to hardware and energy limitations of IoT devices; low latency, typically in the order of milliseconds, to meet the demands of real-time applications; high data security, using cryptographic techniques to improve the integrity of transactions; and efficiency in the use of storage capacity, to handle the large amount of data generated without overloading the devices. The algorithms discussed are PoA, PoS, PoW, PBFT, Proof of Elapsed Time (PoET), and Proof of Importance (PoI).

Yang *et al*. [2020] propose a consensus algorithm based on a genetic algorithm to improve the efficiency of the Practical Byzantine Fault Tolerant (PBFT) algorithm in selecting the primary node. The authors utilize genetic algorithms to select the optimal primary node that will initiate the consensus process in PBFT, considering criteria such as the minimum number of errors or failures and the highest transaction efficiency among other backup nodes, with the goal of reducing system delay and enhancing the consensus process efficiency in the blockchain.

Many related works still have significant limitations. Most solutions rely on a single consensus algorithm or adopt dynamic reconfigurations that impose high overhead on measurement, analysis, and message exchange (e.g., in MAPE-K architectures), which can degrade latency and device resource consumption. Few proposals support run-time consensus selection; they are generally optimised for a specific domain (such as IIoT) and do not react to changes in the data generation pattern. Thus, a lightweight, responsive, and adaptable algorithm selection strategy is needed that can handle multiple IoT contexts without introducing excessive processing or communication overhead.

Table 7 compares the related works considered in this research. This paper advances the state of the art by enabling multiple consensus algorithms on a DAG-based blockchain. This fact enables improvements in block creation time and computational resource consumption, which are essential for IoT systems. The other papers that utilize adaptive mechanisms are developed for a specific purpose; this work aims to provide a solution that adapts to various IoT scenarios.

Furthermore, the combination of consensus algorithms makes blockchain more adaptable to various scenarios and more resilient to attacks, thereby contributing to the integrity of IoT data records. The main contribution of this work is the development of a strategy for selecting the optimal combination of consensus algorithms for IoT systems using genetic algorithms, which enables greater efficiency and versatility in the consensus process.

6 Conclusions and Future Work

This paper presents a novel strategy for selecting and integrating consensus algorithms into blockchains currently used for IoT Systems. By constructing input matrices that contain the performance characteristics of multiple consensus algorithms and the requirements of different IoT domains, the proposed approach employs Simple Genetic Algorithms (SGA) and Adaptive Genetic Algorithms (AGA) to automatically explore and optimize potential combinations.

Experimental results demonstrate that the evolutionary optimisation-based strategy increases blockchain efficiency by reducing block creation time, decreasing the consumption of computational resources such as CPU and RAM, and ensuring higher adaptability in distributed IoT settings. The work extends previous research by integrating multiple consensus mechanisms and by providing a systematic method to guide selection for each IoT-specific scenario.

As future work, the decision matrices can be further refined to incorporate additional performance metrics and more dynamic weighting schemes based on real-world IoT usage patterns. It is also important to assess whether the performance improvement was achieved due to the use of DAG or combinations of consensus algorithms. Furthermore, expanding the evaluation to include a broader set of consensus algorithms and exploring other generic algorithm approaches could lead to even more robust combinations. In addition, other important metrics can be considered, such as MB/Block, as well as security metrics, to provide a more comprehensive assessment of the proposed solution. Another important future activity is conducting performance tests involving multiple users accessing the system simultaneously, in order to more accurately reflect realistic operational scenarios. Finally, the strategy will be implemented in real-world IoT environments, and the scalability and practical impact of the proposed approach will be evaluated in various IoT environments.

Acknowledgements

The authors thank Startup Semine for providing IoT sensor data, which was used for registration in the Blockchain developed in this work.

Funding

This research was funded by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES. Process number: 88887.627616/2021-00.

Authors' Contributions

A.M. de Morais implemented this study. F.A.A Lins and N.S. Rosa contributed to the supervision and review of this paper. A.M. de Morais is the primary contributor and writer of this manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no known competing financial

Table 7. Comparison between related works.

	Use of DAG	Use of Genetic Algorithm	Combination of Consensus Algorithms	Algorithms choice strategy
[Khan *et al*., 2022]				✓
[Rasolroveicy and Fokaefs, 2020]				✓
[Van Meerten *et al*., 2023]		✓		
[Tapwal *et al*., 2021]			✓	✓
[Aghroud *et al*., 2023]				✓
[Yang *et al*., 2020]		✓		
This Paper	✓	✓	✓	✓

interests or personal relationships that could have appeared to influence the work reported in this paper.

Availability of data and materials

The source code used in this work is available at: Click Here.

The datasets generated and/or analysed during the current study are available in: Click Here.

References

Abellán Álvarez, I., Gramlich, V., and Sedlmeir, J. (2024). Unsealing the secrets of blockchain consensus: A systematic comparison of the formal security of proof-of-work and proof-of-stake. *arXiv e-prints*, pages arXiv–2401. DOI: 10.48550/arXiv.2401.14527.

Aghroud, M., Oualla, M., and El Bermi, L. (2023). The criteria for adapting a blockchain consensus algorithm to iot networks. In *Computer Sciences & Mathematics Forum*, volume 6, page 9. MDPI. DOI: 10.3390/cmsf2023006009.

Bach, L. M., Mihaljevic, B., and Zagar, M. (2018). Comparative analysis of blockchain consensus algorithms. In *2018 41st international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 1545–1550. Ieee. DOI: 10.23919/mipro.2018.8400278.

Bhushan, B., Sahoo, C., Sinha, P., and Khamparia, A. (2021). Unification of blockchain and internet of things (biot): requirements, working model, challenges and future directions. *Wireless Networks*, 27:55–90. DOI: 10.1007/s11276-020-02445-6.

Bowman, M., Das, D., Mandal, A., and Montgomery, H. (2021). On elapsed time consensus protocols. In *Progress in Cryptology–INDOCRYPT 2021: 22nd International Conference on Cryptology in India, Jaipur, India, December 12–15, 2021, Proceedings 22*, pages 559–583. Springer. DOI: 10.1007/978-3-030-92518-5$_2$5.

Castro, M., Liskov, B., *et al*. (1999). Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186. DOI: 10.5555/296806.296824.

De Angelis, S., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., Sassone, V., *et al*. (2018). Pbft vs proof-of-authority: Applying the cap theorem to permissioned blockchain. In *CEUR workshop proceedings*, volume 2058. CEUR-WS. DOI: 10.5281/zenodo.1169272.

de Morais, A. M., Lins, F. A. A., and Rosa, N. S. (2023a). Integration and evaluation of blockchain consensus algorithms for iot environments. In Barolli, L., editor, *Advanced Information Networking and Applications*, pages 1–13, Cham. Springer International Publishing. DOI: 10.1007/978-3-031-28451-9$_1$.

de Morais, A. M., Lins, F. A. A., and Rosa, N. S. (2023b). Survey on integration of consensus mechanisms in iot-based blockchains. *JUCS - Journal of Universal Computer Science*, 29(10):1139–1160. DOI: 10.3897/jucs.94929.

de Morais, A. M., Lins, F. A. A., and Rosa, N. S. (2024). Selecting blockchain consensus algorithms integrations for iot-based enviroments. In *Proceedings of the 13th Latin-American Symposium on Dependable and Secure Computing*, pages 116–125. DOI: 10.1145/3697090.3697100.

Digitale, J. C., Martin, J. N., and Glymour, M. M. (2022). Tutorial on directed acyclic graphs. *Journal of Clinical Epidemiology*, 142:264–267. DOI: 10.1016/j.jclinepi.2021.08.001.

Fu, W., Wei, X., and Tong, S. (2021). An improved blockchain consensus algorithm based on raft. *Arabian Journal for Science and Engineering*, 46(9):8137–8149. DOI: 10.1007/s13369-021-05427-8.

Gaži, P., Kiayias, A., and Zindros, D. (2019). Proof-of-stake sidechains. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 139–156. IEEE. Available at:`https://eprint.iacr.org/2018/1239`.

Holland, J. H. (1975). Adaptation in natural and artificial systems. *The University of Michigan Press*. DOI: 10.7551/mitpress/1090.001.0001.

Howard, H. and Mortier, R. (2020). Paxos vs raft: Have we reached consensus on distributed consensus? In *Proceedings of the 7th Workshop on Principles and Practice of Consistency for Distributed Data*, pages 1–9. DOI: 10.48550/arXiv.2004.05074.

Isaacs, K. E. and Gamblin, T. (2018). Preserving command line workflow for a package management system using ascii dag visualization. *IEEE transactions on visualization and computer graphics*, 25(9):2804–2820. DOI: 10.1109/tvcg.2018.2859974.

Jain, R. (1991). *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*, volume 1. Wiley New York. Book.

Jakobović, D. and Golub, M. (1999). Adaptive genetic algorithm. *Journal of computing and information technology*, 7(3):229–235. Avaialble at:`https://scispace.com/pdf/adaptive-genetic-algorithm-30uwb4qvir.pdf`.

Javed, M. U., Rehman, M., Javaid, N., Aldegheishem, A., Alrajeh, N., and Tahir, M. (2020). Blockchain-based secure data storage for distributed vehicular networks. *Applied Sciences*, 10(6):2011. DOI: 10.3390/app10062011.

Kamran, M., Khan, H. U., Nisar, W., Farooq, M., and Rehman, S.-U. (2020). Blockchain and internet of things: A bibliometric study. *Computers & Electrical Engineering*, 81:106525. DOI: 10.1016/j.compeleceng.2019.106525.

Karantias, K., Kiayias, A., and Zindros, D. (2020). Proof-of-burn. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*, pages 523–540. Springer. DOI: 10.1007/978-3-030-51280-4_28.

Khan, A. A., Laghari, A. A., Rashid, M., Li, H., Javed, A. R., and Gadekallu, T. R. (2023). Artificial intelligence and blockchain technology for secure smart grid and power distribution automation: A state-of-the-art review. *Sustainable Energy Technologies and Assessments*, 57:103282. DOI: 10.1016/j.seta.2023.103282.

Khan, M., den Hartog, F., and Hu, J. (2022). A survey and ontology of blockchain consensus algorithms for resource-constrained iot systems. *Sensors*, 22(21):8188. DOI: 10.3390/s22218188.

Kiayias, A., Russell, A., David, B., and Oliynykov, R. (2017). Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual international cryptology conference*, pages 357–388. Springer. DOI: 10.1007/978-3-319-63688-7_12.

Larimer, D. (2014). Delegated proof-of-stake (dpos). *Bitshare whitepaper*, 81:85. Available at:`https://docs.bitshares.org/en/master/technology/dpos.html`.

Lee, Y., Rathore, S., Park, J. H., and Park, J. H. (2020). A blockchain-based smart home gateway architecture for preventing data forgery. *Human-centric Computing and Information Sciences*, 10(1):9. DOI: 10.1186/s13673-020-0214-5.

Lei, M., Xu, L., Liu, T., Liu, S., and Sun, C. (2022). Integration of privacy protection and blockchain-based food safety traceability: Potential and challenges. *Foods*, 11(15):2262. DOI: 10.3390/foods11152262.

Liao, Z. and Cheng, S. (2023). Rvc: A reputation and voting based blockchain consensus mechanism for edge computing-enabled iot systems. *Journal of Network and Computer Applications*, 209:103510. DOI: 10.1016/j.jnca.2022.103510.

Lobo, F., Lima, C. F., and Michalewicz, Z. (2007). *Parameter setting in evolutionary algorithms*, volume 54. Springer Science & Business Media. DOI: 10.1007/978-3-540-69432-8.

Mäkinen, V., Tomescu, A. I., Kuosmanen, A., Paavilainen, T., Gagie, T., and Chikhi, R. (2019). Sparse dynamic programming on dags with small width. *ACM Transactions on Algorithms (TALG)*, 15(2):1–21. DOI: 10.1145/3301312.

Meshcheryakov, Y., Melman, A., Evsutin, O., Morozov, V., and Koucheryavy, Y. (2021). On performance of pbft blockchain consensus algorithm for iot-applications with constrained devices. *IEEE Access*, 9:80559–80570. DOI: 10.1109/access.2021.3085405.

Milutinovic, M., He, W., Wu, H., and Kanwal, M. (2016). Proof of luck: An efficient blockchain consensus protocol. In *proceedings of the 1st Workshop on System Software for Trusted Execution*, pages 1–6. DOI: 10.1145/3007788.3007790.

Monrat, A. A., Schelén, O., and Andersson, K. (2019). A survey of blockchain from the perspectives of applications, challenges, and opportunities. *IEEE Access*, 7:117134–117151. DOI: 10.1109/access.2019.2936094.

Nakamoto, S. (2008). A peer-to-peer electronic cash system. *Bitcoin*, 4. Available at:`https://bitcoin.org/bitcoin.pdf`.

Ongaro, D. and Ousterhout, J. (2015). The raft consensus algorithm. *Lecture Notes CS*, 190:2022. Available at:`https://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro`.

P4Titan (2014). A peer-to-peer crypto-currency with proof-of-burn "mining without powerful hardware". Available at:`https://slimcoin.info/whitepaperSLM.pdf` acesso em 22/03/2024.

Paul, R., Ghosh, N., Sau, S., Chakrabarti, A., and Mohapatra, P. (2021). Blockchain based secure smart city architecture using low resource iots. *Computer Networks*, 196:108234. DOI: 10.1016/j.comnet.2021.108234.

Rasolroveicy, M. and Fokaefs, M. (2020). Dynamic reconfiguration of consensus protocol for iot data registry on blockchain. In *Proceedings of the 30th Annual International Conference on Computer Science and Software Engineering*, pages 227–236. Avaialble at:`https://dl.acm.org/doi/10.5555/3432601.3432632`.

Salimitari, M., Chatterjee, M., and Fallah, Y. P. (2020). A survey on consensus methods in blockchain for resource-constrained iot networks. *Internet of Things*, 11:100212. DOI: 10.36227/techrxiv.12152142.v1.

Silvano, W. F. and Marcelino, R. (2020). Iota tangle: A cryptocurrency to communicate internet-of-things data. *Future generation computer systems*, 112:307–319. DOI: 10.1016/j.future.2020.05.047.

Son, B., Lee, J., and Jang, H. (2020). A scalable iot protocol via an efficient dag-based distributed ledger consensus. *Sustainability*, 12(4):1529. DOI: 10.3390/su12041529.

Tapwal, R., Deb, P. K., Misra, S., and Pal, S. K. (2021). Amaurotic-entity-based consensus selection in blockchain-enabled industrial iot. *IEEE Internet of Things Journal*, 9(14):11648–11655. DOI: 10.1109/jiot.2021.3131501.

Ugochukwu, N. A., Goyal, S., Rajawat, A. S., Islam, S. M., He, J., and Aslam, M. (2022). An innovative blockchain-based secured logistics management architecture: utilizing an rsa asymmetric encryption method. *Mathematics*, 10(24):4670. DOI: 10.3390/math10244670.

Van Meerten, M., Ozkan, B. K., and Panichella, A. (2023). Evolutionary approach for concurrency testing of ripple blockchain consensus algorithm. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 36–47. IEEE. DOI: 10.1109/icse-seip58684.2023.00009.

Vose, M. D. (1999). *The simple genetic algorithm: foundations and theory*. MIT press. Book.

Wang, Q., Li, R., Wang, Q., Chen, S., and Xiang, Y. (2022). Exploring unfairness on proof of authority: Order manipulation attacks and remedies. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 123–137. ACM. DOI: 10.48550/arxiv.2203.03008.

Wang, Q., Yu, J., Chen, S., and Xiang, Y. (2023). Sok: Dag-based blockchain systems. *ACM Computing Surveys*, 55(12):1–38. DOI: 10.1145/3576899.

Wang, T., Wang, Q., Shen, Z., Jia, Z., and Shao, Z. (2021). Understanding characteristics and system implications of dag-based blockchain in iot environments. *IEEE Internet of Things Journal*, 9(16):14478–14489. DOI: 10.1109/jiot.2021.3108527.

Wu, M., Wang, K., Cai, X., Guo, S., Guo, M., and Rong, C. (2019). A comprehensive survey of blockchain: From theory to iot applications and beyond. *IEEE Internet of Things Journal*, 6(5):8114–8154. DOI: 10.1109/jiot.2019.2922538.

Xiao, B., Jin, C., Li, Z., Zhu, B., Li, X., and Wang, D. (2021). Proof of importance: A consensus algorithm for importance based on dynamic authorization. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 510–513. DOI: 10.1145/3498851.3499007.

Yang, C., Wang, T., and Wang, K. (2020). A consensus mechanism based on an improved genetic algorithm. *Open Access Library Journal*, 7(9):1–6. DOI: 10.4236/oalib.1106713.

Zaabar, B., Cheikhrouhou, O., Jamil, F., Ammi, M., and Abid, M. (2021). Healthblock: A secure blockchain-based healthcare data management system. *Computer Networks*, 200:108500. DOI: 10.1016/j.comnet.2021.108500.

Zhipeng, F. (2019). Research on blockchain hybrid consensus algorithm based on internet of things. *Int. J. Internet Things Big Data*. DOI: 10.21742/ijitbd.2019.4.1.05.

Zubaydi, H. D., Varga, P., and Molnár, S. (2023). Leveraging blockchain technology for ensuring security and privacy aspects in internet of things: a systematic literature review. *Sensors*, 23(2):788. DOI: 10.3390/s23020788.