Enhancing Consensus Mechanisms in Consortium Blockchains: A Novel Approach with Backpropagation Neural Network, Segmented DAG, and Genetic Node Selection

BHAUMIK TYAGI[1], ANSHIKA BANSAL[2], SARTHAK PASRICHA[3], RAHUL KUMAR[4], DAAMINI BATRA[5]

[1] Jr. Research Scientist,Computer Science Engineering, Delhi, India

[2, 3] Undergraduate Student, Electronics and Communication Engineering, ADGITM, Delhi, India

[4] Graduate Student,The Heritage College, Kolkata, India

[5] Undergraduate Student, Information Technology, ADGITM, Delhi, India

Abstract—Blockchain technology has garnered significant attention for its potential to transform various industries, with a pivotal role played by consensus algorithms in ensuring the security and reliability of blockchain networks. This paper introduces a novel approach to consensus algorithm design, presenting a solution rooted in the Segmented Directed Acyclic Graph (DAG) and Backpropagation (BP) Neural Network for Consortium Blockchain. In contrast to conventional consensus mechanisms, the proposed algorithm harnesses the power of Segmented DAG to structure transaction data, enhancing scalability and reducing latency. The integration of the BP Neural Network further enhances consensus decision-making by incorporating machine learning principles, adapting to changing network dynamics and optimizing transaction validation. Comprehensive experimentation and performance evaluation demonstrate the algorithm's superior throughput and fault tolerance when compared to conventional consensus methods. To enhance the performance of the consortium blockchain consensus, the Practical Byzantine Fault Tolerance (PBFT) consensus, widely used in consortium blockchains, is employed to reduce the number of consensus nodes, and optimize performance. Utilizing the PBFT consensus, high-performance nodes are screened to obtain a reliable and limited set of consensus nodes. A genetic algorithm-based blockchain consensus enhancement scheme is proposed, outlining the fitness function of blockchain nodes and employing the genetic algorithm to iteratively select consensus node groups with outstanding indicators, ultimately forming the nodes participating in consensus.

Indexed Terms—Consortium Blockchain, Genetic algorithm, Backpropagation Neural Network, PBFT, Consensus algorithm, Segmented DAG.

I. INTRODUCTION

Blockchain technology, originally popularized by Bitcoin, has emerged as a transformative and disruptive force with far-reaching applications spanning a multitude of industries. At its core, blockchain offers the fundamental promise of a decentralized and tamper-proof ledger, which ensures secure and transparent transaction management. As the maturity of blockchain technology advances, it increasingly finds resonance in consortium settings, where a consortium, or a group of trusted entities, collaborates to maintain a shared ledger. In such environments, consortium blockchains represent a pragmatic convergence between fully public blockchains (e.g., Bitcoin) and private, centralized databases. Consortium blockchains blend the merits of decentralization with controlled access, underpinning a new era of trust and cooperation.At the heart of every blockchain system lies its consensus algorithm, the pivotal mechanism responsible for achieving consensus among network participants regarding the state of the ledger. Traditional consensus algorithms, such as Proof of Work (PoW) and Practical Byzantine Fault Tolerance (PBFT), have played instrumental roles in establishing the credibility and reliability of blockchain networks. However, in consortium blockchain contexts, they confront a set of distinct challenges that need to be effectively addressed. The unique challenges of consortium blockchains encompass the delicate balance between security, scalability, and adaptability to the ever-evolving dynamics of network participants. While PBFT ensures a high degree of trustworthiness, it encounters scalability limitations as the number of participating nodes increases. Conversely, PoW, renowned for its scalability, comes at the steep cost of excessive energy consumption and is ill-suited for permissioned settings. Tackling these intricate issues necessitates innovative approaches to the design of consensus algorithms.This research endeavors to proactively address these challenges by introducing an innovative consensus algorithm tailored explicitly for consortium blockchain environments. Our proposed algorithm harnesses the synergies of the Segmented Directed Acyclic Graph (DAG) and Backpropagation (BP) Neural Network to enhance the efficiency and adaptability of consensus decision-making. The Segmented DAG imparts structural integrity to the organization of transaction data, yielding substantial scalability improvements. Furthermore, the integration of the BP Neural Network injects machine learning principles, optimizing transaction validation processes and fostering adaptability to the dynamic conditions that characterize blockchain networks. The overarching objective of this research is to present a holistic solution to the consensus predicament in the realm of consortium blockchains. Our ambition is to craft an algorithm that not only rigorously fulfills the demanding security requisites of consortium settings but also successfully surmounts the challenges of scalability and adaptability. Through empirical experimentation and rigorous performance evaluations, we intend to demonstrate the distinctive advantages of our proposed consensus algorithm in terms of enhanced throughput, reduced latency, and heightened fault tolerance.

The consensus algorithm stands as the fundamental technological cornerstone within blockchain systems, serving the pivotal role of orchestrating agreement among distributed nodes [1]. Predominantly, many of the established consensus algorithms employed in consortium blockchains are rooted in Byzantine Fault Tolerance (BFT) [1]. While the BFT algorithm effectively addresses the "Byzantine General" problem, it concurrently introduces its own set of challenges. For instance, with the burgeoning number of participating nodes, the communication complexity of algorithms such as Practical Byzantine Fault Tolerance (PBFT) [2] escalates significantly, subsequently diminishing the overall system throughput. Additionally, PBFT's leader node election process, based on a serial numerical switching approach, heightens the risk of a malicious node ascending to a leadership position, thereby compromising system security.Presently, the performance of blockchain systems is contingent on a multitude of factors, including broadcast communication, information encryption/decryption, consensus mechanisms, and transaction verification mechanisms [18, 19]. Of these factors, the consensus mechanism aims to harmonize the information maintained by participating nodes. However, achieving consensus in a highly decentralized system proves to be a time-intensive endeavor, exacerbated by the incorporation of potentially "uncooperative" nodes, which magnifies the time complexity of the process.

The genetic algorithm [20] is an inherently randomized search and optimization technique inspired by principles derived from evolution and natural genetics. This algorithm excels in navigating complex, extensive, and multimodal landscapes, yielding solutions that approach optimality [21]. The distributed nature of the genetic algorithm aligns harmoniously with the decentralized node structure inherent to blockchain systems. Notably, several experts have explored the application of genetic algorithms to enhance blockchain performance. For instance, Li, Wu, and Chen [22] devised a blockchain-based security architecture for distributed cloud storage, customizing a genetic algorithm to address the placement of file block replicas across multiple users and data centers within a distributed cloud storage environment. Hussein et al. [23] introduced a scalable and resilient system for managing medical records and information via blockchain technology. In this context, the genetic algorithm was harnessed to expedite transaction processing and bolster reliability during the verification request phase.

This academic discourse underscores the critical role of consensus algorithms within blockchain technology, delves into the challenges associated with prevailing BFT-based algorithms, and explores the potential of genetic algorithms to enhance the performance and resilience of blockchain systems.

The primary contributions of this study can be succinctly encapsulated as follows:

- Development of an innovative blockchain consensus protocol, rooted in the PBFT consensus algorithm, which affords optional node participation in consensus, thereby leading to reduced computational overhead and enhanced performance.
- Deployment of a genetic algorithm to meticulously screen nodes participating in the consensus algorithm, effectively streamlining the complexity inherent in blockchain user interactions.
- Formulation and execution of controlled experiments, comparing the operational data of consortium blockchains with and without the inclusion of our node selection scheme. The findings of our study demonstrate that the blockchain consensus approach, which incorporates the genetic algorithm, outperforms scenarios where all nodes partake in consensus or where node participation occurs without the benefit of the genetic algorithm's screening process. Furthermore, our results underscore the capacity of our proposed algorithm to significantly reduce computational costs relative to alternative solutions.

II. LITERATURE REVIEW

In response to the challenge of mitigating the actions of malicious nodes in blockchain systems, researchers have proposed the incorporation of reputation mechanisms into Byzantine Fault Tolerance (BFT) algorithms. For instance, Alex et al. [3] introduced a reputation-based consensus algorithm designed to counteract "Sybil" attacks, thereby reducing the susceptibility of malicious nodes ascending to leadership roles. However, this algorithm still grapples with a limitation pertaining to diminishing throughput as the number of nodes increases, primarily due to the relatively fixed nature of the reputation model computation. Consequently, its scalability is compromised. In a similar vein, DE Oliverira et al. [4] presented an adaptive hedging algorithm aimed at dynamically altering the calculation of the reputation model. Regrettably, this algorithm exhibits reduced responsiveness, particularly in scenarios where the leader node experiences an outage, resulting in abnormal operation and challenges in restoring normal functionality. Furthermore, it's worth noting that the above-mentioned consensus algorithms rely on traditional chain structures as their underlying data architecture, which imposes limitations on the system's overall throughput.

Directed Acyclic Graph (DAG) [5], conversely, presents qualities that are well-suited to addressing issues of high concurrency, thereby significantly enhancing system throughput. While traditional DAGs offer improved performance in terms of throughput, they are not without their shortcomings, including concerns related to double-spending and elevated retrieval complexity. In summation, contemporary mainstream consortium blockchain consensus algorithms are beset with challenges encompassing diminished throughput, limited scalability, and security vulnerabilities.

In response to the aforementioned challenges, this research introduces a consensus algorithm tailored for consortium blockchains, emphasizing low communication resource consumption, dependable performance, and effortless scalability. The ensuing contributions of this study can be summarized as follows:

- Introduction of a node reputation evaluation mechanism, grounded in the Backpropagation (BP) neural network, aimed at achieving a more precise measurement of a node's creditworthiness. This reputation-driven approach employs the computed reputation values to discern and appoint accounting nodes, thereby mitigating the risk of malicious nodes assuming accounting roles. Furthermore, it enables the selection of multiple nodes with higher reputation scores to partake in the committee's activities, enhancing the security of transaction message verification.
- Development of a partition structure that facilitates nodes' seamless entry and exit, effectively addressing the challenge of scalability. This innovative approach incorporates a segmented Directed Acyclic Graph (DAG) as the underlying data storage framework, consequently remedying the issue of suboptimal throughput traditionally associated with DAGs. Simultaneously, it reduces the retrieval complexity, enhancing the

granularity of data operations by adopting transactions as the fundamental storage unit.

- Introduction of a robust double-spending mechanism, predicated on MapReduce [6], which ensures the global uniqueness of data. This mechanism concurrently resolves the scalability limitations inherent in Byzantine Fault Tolerance (BFT) consensus algorithms and combats the double-spending predicament associated with DAG [7].

The Byzantine fault-tolerant (BFT) consensus mechanism stands as a prevalent choice within consortium blockchains [24]. Its appeal lies in its capacity to uphold robust throughput, with a stringent criterion that limits malicious nodes within a defined threshold. BFT algorithms, including the Practical Byzantine Fault Tolerance (PBFT) [26] and its various derivatives, are among the most frequently adopted consensus mechanisms in consortium blockchains. Their inception was driven by the imperative to design low-latency storage systems capable of reducing algorithmic complexity.The operational procedure of PBFT unfolds across three distinct phases: pre-preparation, preparation, and commitment. A service operation is validated when it garners the approval of more than two-thirds of the network's nodes. Notably, PBFT can accommodate up to 'f' Byzantine malicious nodes within a fault-tolerant blockchain network, where 'f' equals (N − 1)/3, with 'N' representing the total number of participating nodes. This innovation overcomes the inefficiency of the original BFT algorithm, alleviating its algorithmic complexity from an exponential to a polynomial scale, rendering BFT algorithms practicable for real-world system applications. Some of the most widely embraced consortium blockchain platforms, such as Hyperledger's Fabric platform [27], lend their support to the PBFT consensus mechanism.

The Quorum mechanism [28] represents a frequently employed approach in distributed systems, serving to ensure data redundancy and foster eventual consistency within the context of voting algorithms. The fundamental concept underlying this mechanism derives from the "drawer principle," a conventional method often utilized for controlling read-and-write access in distributed systems. Notably, this class of consensus protocol operates without the necessity of inter-node communication. In this model, client requests are directly executed by nodes, and their responses are relayed back to the clients. Once an adequate number of responses are collected, the client subsequently submits the final results. However, it is important to acknowledge that, in the event of a Byzantine fault, resolving such errors typically incurs substantial costs. Furthermore, owing to the absence of a request-sorting mechanism, the Quorum approach encounters challenges in effectively handling contention.

The genetic algorithm [29, 30], initially conceptualized and developed by Holland [31], is a randomized search and optimization technique that draws inspiration from the principles of evolution and natural genetics. This algorithm is characterized by a considerable degree of inherent parallelism. It finds utility in navigating complex, extensive, and multi-modal problem spaces. A notable attribute of the genetic algorithm lies in its ability to provide an approximate optimal solution to the objective or fitness function associated with the optimization problem at hand.Fundamentally, the genetic algorithm treats the entirety of individuals within a given population as objects of optimization and employs a randomized approach to efficiently explore the coded parameter space. The genetic operations of this algorithm encompass selection, crossover, and mutation. Five pivotal elements underpin the genetic algorithm: parameter coding, initial population configuration, fitness function formulation, design of genetic operations, and the establishment of control parameters [32]. These core components collectively define the essence of the genetic algorithm, making it a potent tool for complex problem-solving and optimization. Michalewicz [33] has posited that evolution programs share common attributes with genetic algorithms, although they distinguish themselves by incorporating problem-specific knowledge using "natural" data structures and problem-sensitive "genetic" operators. The primary demarcation between genetic algorithms (GA) and evolutionary programming (EP) resides in their classification. GA is regarded as a weak, problem-independent method, whereas this classification does not apply to EP. Consequently, GA is expected to possess the capability to solve any problem addressable by EP or an evolutionary algorithm (EA). However, the reverse is not necessarily true, as EP or EA may not be equipped to solve all the problems that GA can address. It is important to acknowledge that the generality of GA often comes at the cost of efficiency. Furthermore, when candidate solutions directly exchange information, it implies a deviation from the evolutionary algorithm/programming paradigm [34].

The inherent structure of the GA chromosome aligns well with the consensus node structure inherent to blockchain technology. Moreover, this paper necessitates the chromosome hybridization process to yield improved outcomes. It is noteworthy that the algorithm introduced in this study operates in conjunction with the consensus algorithm of the consortium blockchain, yet refrains from direct participation in specific consensus computations, thereby having no adverse impact on the computational speed of the consensus process. The rationale for selecting GA over EA is grounded in these considerations.

III. RELATED BACKGROUND

Consensus Algorithm: Generally, a good consensus algorithm can greatly save the time required for the synchronization of the ledger data of the blockchain network nodes, thereby improving the operating efficiency of the entire blockchain system. At present, the consensus algorithms used in the blockchain framework can be roughly divided into three categories: the first is based on the attribute value proof of the node itself.

The PoW algorithm proposed by Satoshi Nakamoto solves the Byzantine problem but uses the method of solving the hash problem to select the accounting nodes, which has the problem of wasting computing power and lower throughput. Afterward, PoS proposes solutions to the problems of excessive waste of PoW resources and slow block generation time, but there are problems such as harmless attacks and long-range attacks. Moreover, DPoS introduces a proxy mechanism, and token holders can elect supernodes as accounting representatives to solve the problem of oligarchy. But when abnormal super nodes appear, the election system cannot solve the problems caused by abnormal and malicious nodes in time. PBFT is a method of state machine copy replication to solve the BFT problem. In PBFT, all replica states are converted in the view, and the leader node selection method is the master node view number modulo the number of nodes. That is, a round of consensus is to take a view as a cycle and switch views when the consensus is completed. Although PBFT reduces the communication complexity in the BFT problem from exponential to polynomial, the nodes need to continuously broadcast message; as the scale becomes larger, the network performance requirements are higher, and the efficiency becomes slower and slower. More precisely, when the leader node in PBFT switches frequently, the complexity will reach O(n3), so this method is only suitable for consortium blockchain. As such, for the problem of high communication complexity, HotStuff adopts the way that all messages are received and distributed by the leader node, which reduces the average communication complexity of PBFT from O(n2) to O(n). In addition, the view switching and consensus process in PBFT are executed separately. If the views are frequently switched, the communication complexity is as high as O(n3).

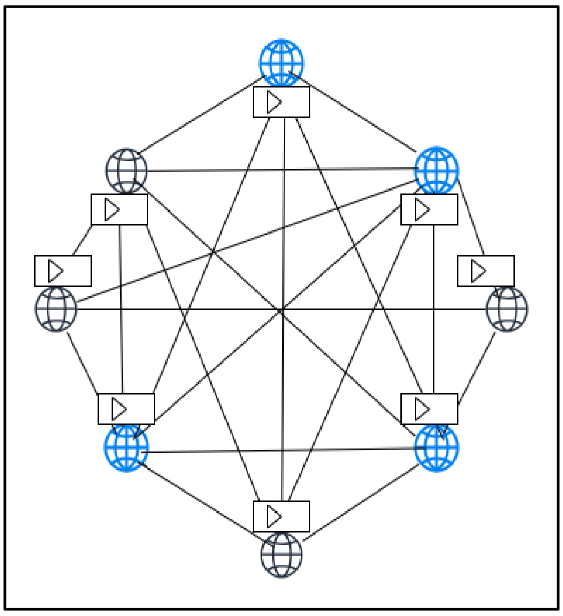
Fig 1: Consortium Blockchain Architecture

Currently, a general classification divides blockchain into three categories, including public blockchain, consortium blockchain, and private blockchain. Among them, the consortium blockchain is widely welcomed due to the features of controlled access node identity and decentralized storage. However, the existing consortium blockchain architecture still suffers from the low performance of the consensus mechanism, which leads to low operational efficiency of the whole blockchain system.

Besides, Liu et al. [8] proposed a consensus mechanism of reputation proof, which solves the problem that the verification node in the blockchain is vulnerable to attack and loses the ability to distinguish honest nodes. By constructing all nodes into a directed weighted graph, the largest weakly connected branch is taken as the set of verification nodes with the highest positivity. Moreover, the “Leader-Rank” algorithm [9] is used to calculate the contribution degree of the verification node according to the out-degree and in-degree of the node. Afterward, it calculates the reliability of the number of valid blocks, valid votes, invalid blocks, and invalid votes created by the verification node and finally calculates the weighted sum of the contribution and reliability to obtain the final comprehensive reputation.

Based on the comprehensive reputation ranking, the leader in the current round of BFTis selected. This reputation-proof mechanism can effectively solve the problem of verifying nodes being manipulated by attacks, but it has the problem of unbalanced weight distribution between contribution and reliability and potential reputation oligarchs. Among them, literature [10] proposed a PoS consensus mechanism based on reputation. Aiming at the problems of low performance and low security of existing blockchains, a master-slave multichain structure is designed to ensure that the block information cannot be tampered with through the anchoring of the master-slave chain. At the same time, a joint consensus mechanism for the main chain is proposed, which uses multiple consensus mechanisms to calculate together in the main-slave chain. However, the use of different algorithms on the master-slave chain will produce a barrel effect, which leads to the problem of high concurrency difficulty.

In summary, from the above consensus algorithm, we can see that the election methods of accounting nodes are mainly randomly selected, fixed elections, or elections based on some attribute values. A good election method of accounting nodes can increase the security of the whole system. Therefore, the election method of the accounting node becomes particularly critical. Generally, the election method not only considers the weight distribution of attributes but also needs to consider the performance of the entire network.

Backpropagation Algorithm (BP) [11]. As the core of deep learning [12–14], the BP algorithm’s function is to calculate the error based on the forward output and then conduct backpropagation to adjust the weights in the neural network based on the error. In brief, the core idea of the BP algorithm is to use gradient descent to find the most suitable weights and bias values so that the fit of the function is optimal. /e BP neural network model is shown in Figure 1. As can be seen, the model is divided into an input layer, a hidden layer, and an output layer. /e connection of neurons between layers is the weight w, and the target of network training is to adjust w to the optimal value. More precisely, we take the adjustment of the first weight w_{11}^1 of the first layer as an example.

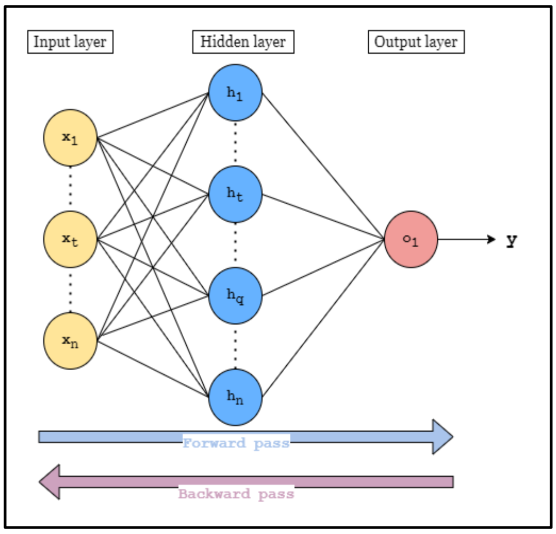

Fig 2: Model of BP neural network

First, the output of O_1 needs to be calculated forward, as shown in formula (1).

$$y = \sum_{i=1}^{n} h_i \times w_{i1}^2 \quad (1)$$

In formula (1), h_i is the neural unit of the hidden layer, and the calculation formula for h_1 is shown as follows:

$$y = \sigma\left(\sum_{j=1}^{n} X_j \times w_{j1}^1\right) \quad (2)$$

Among them, σ is the activation function, and the common activation functions are ReLU, tanh, sigmoid, and so on. /e activation function makes the neural network have a nonlinear fitting ability. Xj is the input value. And then, the calculation formula of the loss value is shown as follows:

$$loss = (y - \hat{y})^2 \quad (3)$$

Among them, the meaning of loss is to measure the difference between the predicted value y and the true value$\hat{y}$. The adjustment method of w_{11}^1 is shown in formula (4).

$$w_{11(2)}^1 = w_{11(1)}^1 - lr \times \Delta w_{11}^1 , \quad (4)$$

where lr is a real number between 0 and 1 and $w_{11(2)}^{1}$ represents the second-round adjustment value of w_{11}^{1}. The result is the first round w_{11}^{1} minus the gradient multiplied by lr, and the calculation method is shown in formula (5).

$$\Delta w_{11}^{1} = \frac{\partial loss}{\partial O_1}\frac{\partial O_1}{\partial h_1}\frac{\partial h1}{\partial w_{11}^{1}} \qquad (5)$$

Finally, repeat formula (4), and after multiple iterations of updating, w_{11}^{1} completes the adjustment. Because the neural network can independently adjust the advantages of characteristic nodes, we use it to predict the reputation value of each node.

MapReduce. Undoubtedly, when an information system has a huge amount of data, the data needs to be divided and processed separately. MapReduce is a computing architecture that uses functional programming ideas to divide a calculation into two calculation processes, Map and Reduce.More precisely, MapReduce can divide a large computing task into multiple small computing tasks and then assign each small computing task to the corresponding computing node in the cluster and always track the progress of each computing node to decide whether to re-execute the task.

Finally, the calculation results on each node are collected and output. Its working principle is shown in Figure 2. In the chaotic and disorderly color data, it first performs the Map operation on the color data, then splits the key-value data structure, and sends it to different computing units performing the Reduce operation, which mainly counts and sorts the number and types of colors. Finally, the results of the types and quantities of colors are summarized. SinceMapReduce has the characteristics of multinode collaboration and deduplication of data, this paper will use this architecture to solve the poor scalability of consensus algorithms and the double-spending problem in DAG.

Directed Acyclic Graph. Particularly, the emergence of DAG has transformed the ledger form from a single chain to a directed acyclic graph pattern, avoiding the limitations of serialized writes that exist in single chains and allowing the ledger to support high concurrency. In fact, in the blockchain represented by Bitcoin, except for the genesis block, each block has one and only one predecessor block and one successor block, and the blocks form a single chain. Conversely, if two blocks are reserved at the same time, it will cause the blockchain to fork. According to the longest chain principle, only one block will be retained on the main chain, and the other will be discarded. Hash Graph[15] framework that uses a Para chain DAG uses a gossip algorithm and virtual voting to confirm that the entire transaction is globally ordered in an asynchronous environment, and there will be a long voting process in the virtual voting stage. For this reason, it will result in more rounds of voting to confirm that the transaction is valid and reliable.This paper mainly discusses the attack methods involved in these three layers. Common attack methods [16] are as follows:

1) Double-spending attacks: When a new transaction enters the block to obtain enough confirmations, and the length of the attacker's side chain exceeds the main chain, the attacker's side chain becomes the main chain. So, the first transaction initiated by the attacker was determined to be invalid, and the double-spending attack was successful.
2) 51% attack: for this paper, an attacker who has more than half of the reputation value is a 51% attack.
3) Solar eclipse attack: an attacker tries to isolate a group or one node, isolate it from communication with other nodes, and prevent it from obtaining the latest world state.
4) Denial of service attack: the node deliberately does not actively participate in the calculation process. In this paper, it can be considered that the calculation process forwards heartbeat packets too much, does not respond or repeatedly sends double-spending transactions, and almost does not send normal transactions. In brief, the proportion of normal transactions that is less than 50% is considered a denial of attack.This paper applies the above attack model to the security considerations of the proposed algorithm.

IV. METHODOLOGY

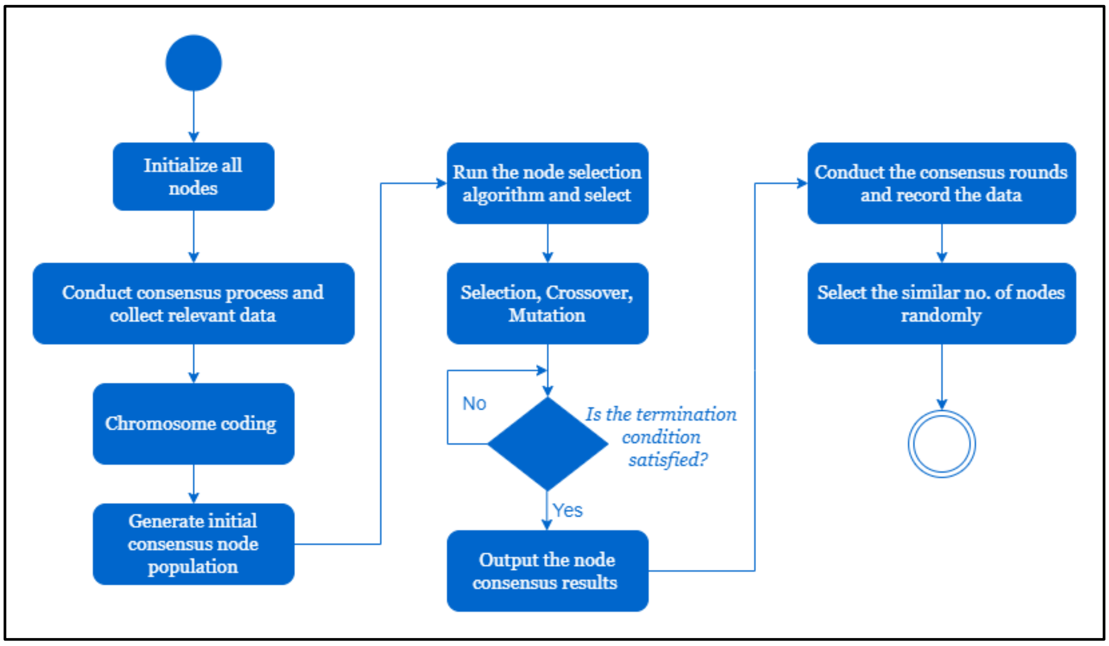
Initialize all nodes
Conduct consensus process and collect relevant data
Chromosome coding
Generate initial consensus node population
Run the node selection algorithm and select
Selection, Crossover, Mutation
No
Is the termination condition satisfied?
Yes
Output the node consensus results
Conduct the consensus rounds and record the data
Select the similar no. of nodes randomly

Fig 3: Overall experimental process

Topology Considerations: In contrast to the sequential processing paradigm of chain structures, Directed Acyclic Graphs (DAG) present an inherently suitable platform for accommodating natural high concurrency. To address the challenges of intricate retrieval processes and prolonged transaction ledger integration times, we introduce a novel approach known as "segmented DAG." Figure 3 illustrates this concept, depicting a segmented DAG structure.In this representation, shaded blocks signify organizing committee blocks, each encompassing the group signature, timestamp, reputation record, and hash pointer group pertinent to the organizing committee. Conversely, white blocks represent ordinary nodes, housing signatures, timestamps, transaction data, and hash pointer groups specific to ordinary nodes. The numerical annotations within the grey and white blocks denote their respective sequence. A black block, functioning as a fast index block array, includes its timestamp, hash pointer, and the block hash associated with its own block. The genesis block (denoted as 0) serves as the immutable organizing committee block, with subsequent blocks linked to it through hash pointers.

The method of connection entails the random selection of the nearest 'n' timestamped transaction data. Periodically, or upon the generation of a predefined number of blocks, a fixed organizing committee block is introduced. This approach fosters the verification and deduplication of transaction information between two organizing committee blocks by the former, enhancing system efficiency.

Moreover, concerning data retrieval, in contrast to retrieving all transactions within the original DAG, we segment the DAG according to the temporal dimension. This enables rapid indexing based on timestamps through reliance on the black block, substantially reducing search complexity.

In a physical context, DAG structures may manifest in two primary forms: the adjacency matrix and the adjacency list. Given the inherent sparsity of the adjacency matrix, we opt for the storage format of the adjacency list. As depicted in Figure 5, the leftmost section features an array of fast index blocks, housing timestamps and hash pointers. The central grey segment represents committee blocks, while the white section represents original blocks. Both committee and white blocks contain transaction information, hash values, and hash pointers, with connections to the corresponding transaction information blocks. This architectural approach enhances efficiency in data storage and retrieval within the DAG structure.

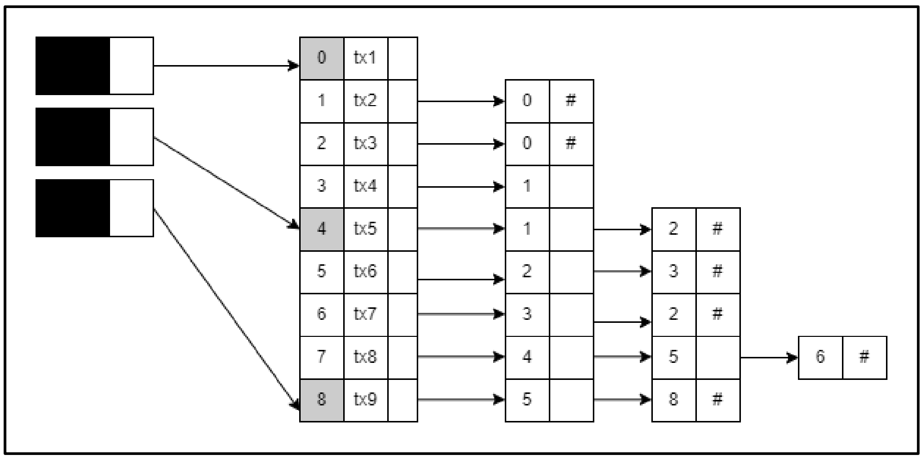

Fig 4: The storage structure of the segmented Directed Acyclic Graph

Aiming at the problem of double-spending that is difficult to eliminate in DAGs, we propose a resistant double-spending mechanism based on MapReduce, as shown in Figure 6. First, use n organizing committee nodes to accept the client's transaction operations, then divide all ordinary nodes into several partitions, and select several ordinary nodes with higher reputation values or organizing committee nodes in different partitions for transaction message statistics. Furthermore, the ordinary node sends the transaction message to the organizing committee node. In addition to verifying the correctness of the transaction, the organizing committee node not only verifies the transaction's correctness but also removes the double-spending transaction message according to the reputation value of the node. Finally, the results are summarized to the leader node for verification. Afterward, the leader packs the transaction message, sends it to the remaining nodes of the organizing committee, and sends it to the other ordinary nodes through the gossip protocol [17].

V. IMPLEMENTATION

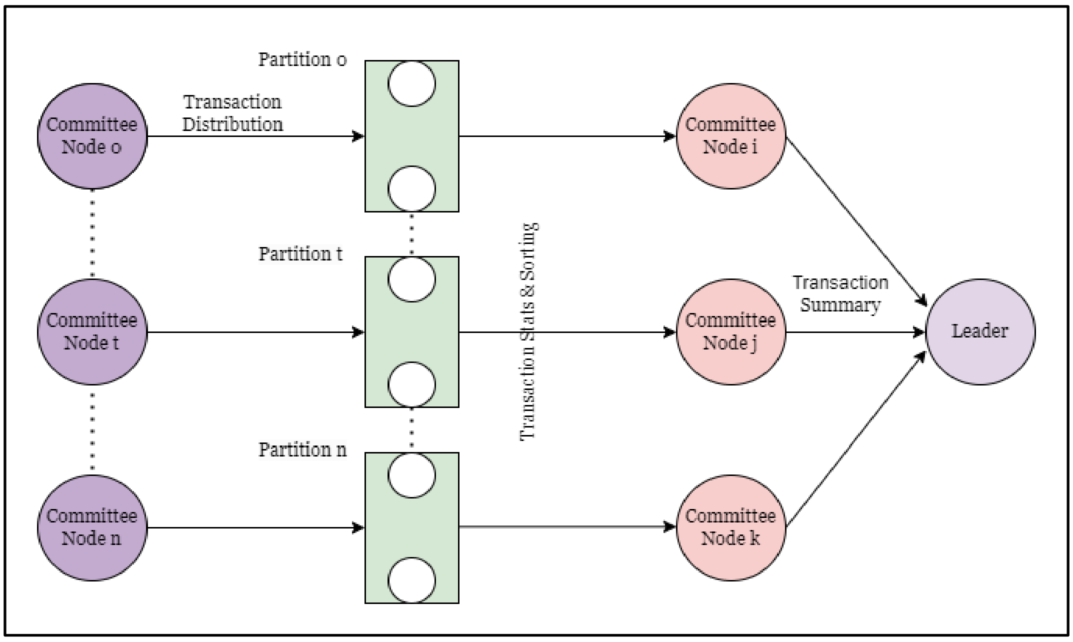

Fig: MapReduce-based Resistant double-spending mechanism

Resistant Double-Spending Mechanism Based on MapReduce. The mechanism used in this paper to remove double-spending transactions based on MapReduce is shown in Algorithm 3. First, ‘n’ organizing committee nodes monitor transaction messages, and the backup leader section group is responsible for monitoring the status of the organizing committee nodes, scheduling, and removing some malicious nodes. Second, each organizing committee node will divide the transaction message into m parts and then send the m parts to m organizing committee nodes with higher reputation value for the map. The map operation will traverse the transaction message and return the data in k-v format. The key is the hash of the transaction message, and the value is the reputation value of the node that sent the transaction. These m organizing committee nodes do the MD5 operation of the key modulate r and then send the k-v to the corresponding r organizing committee nodes. Third, after these r organizing committee nodes receive the corresponding transaction message, they will merge the messages, shuffle the messages according to the size of the value, remove the double-spending operation, and then return the deduplicated transaction message to the leader. Finally, the leader node receives the message of the organizing committee node, performs verification, packs, and returns the DAG structure transaction message.

CONSENSUS ALGORITHM & PROCESS

ALGORITHM 1: Consensus process based on Map Reduce Resistant double-spending mechanism

```
Input:  X (the feature vector of the node), Random seed
Output: flag1 (whether the committee members are successfully elected), flag2 (the consensus result)
1.  Calculate the reputation value of the node in each round using the function formula(X), denoted as Vec.
2.  Sort the reputation values in Vec to determine the reputation ranking, resulting in Sorted_Vec.
3.  Initialize the flag—> flag1 as false.
4.  VRF—> Gen Rand_Num. Utilize the Verifiable Random Function (VRF) to generate a random number and
    corresponding proof, with a specified seed.
5.  Verify the correctness of the generated random number and the accompanying proof within the configured
    time frame (Config_time). If both verifications are successful, and the time condition is met:
6.  a. Proceed to the selection process by invoking the function Choose(Random_number).
7.  b. Set flag1 to True to indicate that the node has been chosen to join the committee.
8.  If the verification process fails or the time condition is not met, return to Step 4.
9.  End the conditional statement.
10. Calculate the reputation value of the node in each round using the function formula(X), denoted as Vec.
11. Sort the reputation values in Vec to determine the reputation ranking, resulting in Sorted_Vec.
12. Utilize the Verifiable Random Function (VRF) to select a leader node, backup node, and follower node based
    on the provided seed. The results are stored in (leader, leader_backs, follower).
13. Gather verification transactions (TX) from followers and sort them, resulting in Tx_sorted.
14. Utilize MapReduce to collect and pack transactions from followers, removing duplicate transactions to form
    a Block.
15. Initialize the flag flag2 as false.
16. Check if the reputation collected in the Block is greater than or equal to half of the total reputation and the
    time condition is met (Time <Config_time).
17. a. If the condition is met, set flag2 to True.
18. b. If the "Term of Service" is satisfied, return to Step 10.
19. c. If the condition is not met, clear the committee (Clean(committee)).
20. If the condition in Step 14 is not met, proceed to Step 15.
21. Change the leader node (Change(leader)).
22. Return to Step 10 to restart the transaction collection.
23. End the conditional statement.
24. Return the value of flag2 as the output of the algorithm.
```

The algorithm does not change the blockchain system process and only needs to provide some relevant consensus data, which are usable, and available to calculate the algorithm.

ALGORITHM 2: Overall Process of the Genetic Algorithm

Input: Number of iterations *iteration_num*, number of consensus nodes *CHROMOSOME_NUM*, total number of nodes *NODE_NUM*.

Output: Data after iteration *temp_evaluation*, consenus node, consenus data, *init_group.*

```
1.  def main ():
2.      evaluation=sys.maxsize
3.  iteration_num = 0
4.  service.init_group (Const.CHROMOSOME_NUM.value,Const.NODE-NUM.value)
5.  for chromosome in init_group:
6.  service.init_group(Const.CHROMOSOME_NUM.value,Const.NODE_NUM.value)
7.  whileevaluation>Const.FITNESS_VAR_LIMIT.valueanditeration_num<Const.ITERATION_NUM.value:
8.  divide_group = Service.divide(init_group, Const.DIVIDE_NUM.value)
9.  new_group = Service.divide_new_group(divide_group, Const.PRO.value)
10.     for chromosome in list(new_group.values())
11. mutated_chromosome = Service.mutated(chromosome)
12.     for chromosome in list(new_group.values())
13. new_group[chromosome.fitness] = Util.check_gene(chromosome, Util.part_in_node gene
    (Const.NODE_NUM.value))
14. temp_evaluation = Service.evaluate(list((new_group.values)()))
15. db.commit()
16. change_node_status(init_group)
```

Initialization: The m chromosomes form a population, and each chromosome initializes the genes. Every chromosome uses the genes to calculate fitness values via the following fitness function:

$Pl_0 = \{\{chr_1, fv_1\}, \{chr_2, fv_2\}, ..., \{chr_m, fv_m\}\}$ (1)

In Eq. (1), $Pl_0$is the 0th generation population: chr_1 and fv_1 are the i^{th}chromosome, calculated the fitness value according to the gene; $ch_i = \{g_1, g_2, ..., g_N\}$, where $g_j$represents the j^{th} gene of chromosome i, and binary code indicates whether the node is selected; N is the total number of consensus nodes; gjindicates the mark of the j^{th} consensus node, which is a Boolean quantity initialized to a random value; and m is a pre-set constant.

Consensus: This step is the consensus of the consortium blockchain. In each round of consensus, a chromosome will be selected from the current population Pl_0 in order and according to the gene expression of the chromosome to participate in the consensus. One chromosome only determines q consensuses in the step. A population performs a total of $q \times m$ consensuses. After the m consensus rounds, Eq. (1) calculates the fitness value for every chromosome in the current population Pl_0.

Generating a new population: According to the fitness value from small to large, the current population Pl_0 of chromosomes is divided into three groups: *G1, G2, and G3*. For the three groups, the chromosomes are selected randomly according to their selection probabilities, Pro_1, Pro_2, and Pro_3, respectively, and then added to the population Pl_0. If population Pl_1 does not reach the expected number of rounds m, the unselected chromosomes in the three groups with the highest fitness value will be selected. In addition, if population Pl_1 still does not reach the expected number of rounds m, the remaining chromosomes will be calculated using the two chosen chromosomes in population Pl_0 via the crossover method, which is the widely used single-point crossover method.

Termination conditions: The current population Pl_1 is evaluated, and the difference in the fitness values S^2 of the population is calculated, which is the variance of the population's chromosome fitness values:

$$S^2 = \frac{1}{m}\sum_{i=1}^{m}[F(x)_i - M]^2 \qquad (2)$$

$$M = \frac{1}{m}\sum_{i=1}^{m}[F(X)i \qquad (3)$$

where M is the F(X) average of m chromosomes, and i is the maximum number of iterations. rS2 is the variance of the fitness value set in advance. When S2 ≤ rS2 or the number ofiterations in the population is greater than the preset value 'i', the genetic algorithm will end, and the optimal fitness value chromosome is the resulting output. Otherwise, we return to step 2 "Consensus" to continue to execute the algorithm.

VI. RESULT & DISCUSSION

Some key results are discussed below:

- Algorithm Activity (Active): The "YES" entry in this column indicates that the proposed algorithm is actively functioning, meaning it is operational and participates in the consensus process.
- Resistant Double-Spending: The term "HIGH" suggests that the proposed algorithm has a high degree of resistance to double-spending, a critical security concern in blockchain systems. This implies that it effectively prevents or mitigates the risk of a single unit of cryptocurrency being spent more than once.
- Complexity: The algorithm's complexity is denoted as "O(n^2)," where 'n' typically represents the number of nodes or some other relevant parameter. This notation implies that the algorithm exhibits quadratic complexity, meaning that its execution time and resource usage increase quadratically with the size of the input data or network.
- Fine-Grained: The "YES" entry in this column indicates that the proposed algorithm offers fine-grained control or precision, likely concerning data operations or transaction processing. Fine-grained control means that the algorithm can manage and manipulate data with a high level of detail or granularity.
- High Concurrency: The "YES" entry in this column signifies that the proposed algorithm supports high concurrency. It can efficiently handle multiple tasks or transactions concurrently, which is crucial for enhancing system performance and throughput.
- Scalability: The "HIGH" entry in the scalability column suggests that the proposed algorithm exhibits strong scalability characteristics. It can adapt and expand as the network or the demands on the system grow, ensuring its continued efficiency and performance.

Complexity and scalability analysis:
Due to many uncertainties, there is no effective quantitative analysis method for genetic algorithm parameters, such as accuracy, feasibility, and computational complexity. This is one of the drawbacks of the genetic algorithm. This paper analyzes the algorithm's complexity according to the proposed fitness function. There are m chromosomes in total, and each chromosome needs to perform q consensuses in each round. The algorithm performs at most t rounds, and one consensus of PBFT has O(n2) complexity. In summary, the complexity of this algorithm is as follows:

$$O(n) = m \times q \times O(n2) \times t$$

The algorithm is scalable to a certain extent. Theoretically, it can be applied to any consortium blockchain with several nodes more significant than 3n + 1. However, the running speed will decrease with the number of nodes.

CONCLUSION

In summary, consortium blockchain architecture has emerged as the preferred choice for various blockchain applications. However, it grapples with throughput limitations attributable to its reliance on traditional chain structures. While the introduction of Directed Acyclic Graph (DAG) has indeed bolstered system throughput, it concurrently introduces new challenges, characterized by heightened algorithmic complexity and concerns regarding double-spending.In response to these challenges, this research advances a consensus algorithm engineered to provide high concurrency and scalability within consortium blockchains. A pivotal innovation involves the design of a segmented DAG structure, strategically positioned to enhance system throughput while simultaneously mitigating the time complexity associated with global retrieval. The implementation of a resistant double-spending mechanism, rooted in the MapReduce architecture, effectively safeguards the global uniqueness of transactions. Moreover, the consensus algorithm proposed in this study finds particular relevance in the realm of large-scale sensor networks within the Internet of Things. It amplifies the security of computational processes and augments the scalability of device clusters. An additional layer of security is introduced through the election of trustworthy nodes via a reputation model founded on the principles of Backpropagation (BP) neural networks. Empirical results from simulation experiments affirm the superior performance of the algorithm outlined in this research. However, it is imperative to acknowledge certain aspects warranting further in-depth exploration:

1. Underlying Topology: The existing DAG structure, while delivering high concurrency and parallel capabilities surpassing those of traditional chains, grapples with security issues and presents challenges in resolving the double-spending problem.
2. Network Fragmentation: The concept of network segmentation and autonomy offers the potential to reduce communication scale and expedite consensus processes.
3. Hybrid Consensus: A discernible trend in the realm of consensus algorithms is the integration of hybrid approaches. Investigating the fusion of proof-like algorithms and Byzantine Fault Tolerance (BFT) technology holds promise as a meaningful research direction.

REFERENCES

[1] M. Seyed, M. Amirhossein, and B. Alireza, “A survey of blockchain consensus algorithms performance evaluation criteria,” Expert Systems with Applications, vol. 154, no. 16, pp. 113385–113406, 2020.

[2] H. Sukhwani, J. M. Martinez, and X. Chang, “Performance modeling of PBFT consensus process for permissioned blockchain network (hyperledger fabric),” in Proceedings of the 36th Symposium on Reliable Distributed Systems (SRDS), Hong Kong, China, September 2017.

[3] A. Biryukov and F. D. ReCon, “sybil-resistant consensus from reputation,” Pervasive and Mobile Computing, vol. 61, no. 1, pp. 1574–1192, 2020.

[4] M. T. d. Oliveira, L. H. A. Reis, D. S. V. Medeiros, R. C. Carrano, S. D. Olabarriaga, and D. M. F. Mattos, “Blockchain reputation-based consensus: a scalable and resilient mechanism for distributed mistrusting applications,” Computer Networks, vol. 179, no. 10, pp. 107367–107380, 2020.

[5] A. Bakhtiar, F. Syahirul, and D. /anh, “Big data directed acyclic graph model for real-time COVID-19 twitter stream detection,” Pattern Recognition, vol. 123, no. 3, pp. 108404– 108416, 2022.

[6] J. Dean and S. Ghemawat, “MapReduce,” Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.

[7] K. Lyudmila, K. Dmytro, and N. Andrii, “Decreasing security threshold against double spend attack in networks with slow synchronization,” Computer Communications, vol. 154, no. 6, pp. 75–81, 2020.

[8] N. A. Lin, Z. H. Chen, and G. K. Liu, “Mechanism for proofof-reputation consensus for blockchain validator nodes,”Journal of XidianUniverSity, vol. 47, no. 5, pp. 61–66, 2020.

[9] Q. Li, T. Zhou, L. L¨u, and D. Chen, “Identifying influential spreaders by weighted LeaderRank,” Physica A: Statistical Mechanics and Its Applications, vol. 404, no. 2, pp. 47–55, 2014.

[10] H. Liu, S. Li, and W. Lv, “Master-slave multiple-blockchain consensus based on credibility,” Journal of Nanjing University of Science and Technology, vol. 44, no. 3, pp. 325–331, 2020.

[11] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[12] X. Zhu, Y. Luo, A. Liu, M. Z. A. Bhuiyan, and S. Zhang, “Multiagent deep reinforcement learning for vehicular computation offloading in IoT,” IEEE Internet of things Journal, vol. 8, no. 12, pp. 9763–9773, 2021.

[13] M. Chen, T. Wang, S. Zhang, and A. Liu, “Deep reinforcement learning for computation offloading in mobile edge computing environment,” Computer Communications, vol. 175, no. 11, pp. 1–12, 2021.

[14] Y. Liu, Y. X. Lan, and B. Y. Li, “Proof of Learning (PoLe): empowering neural network training with consensus building on blockchains,” Computer Networks, vol. 2011, no. 2, pp. 108594–108603, 2021.

[15] Hedera, “/e swirldshashgraph consensus algorithm: fair, fast, byzantine fault tolerance,” 2022, https://docs.hedera. com/guides/core-concepts/hashgraph-consensus-algorithms.

[16] Y. Wen, F. Lu, Y. Liu, and X. Huang, “Attacks and countermeasures on blockchains: a survey from layering perspective,” Computer

Networks, vol. 191, no. 8, pp. 107978–107994, 2021.

[17] T. Yu and J. Xiong, “Distributed consensus-based estimation and control of large-scale systems under gossip communication protocol,” Journal of the Franklin Institute, vol. 357, no. 14, pp. 10010–10026, 2020.

[18] Chen S, Zhang J, Shi R, Yan J (2018) A comparative testing on performance of Blockchain and relational database: foundation for applying Blockchain into current business systems. In: Streitz N, Konomi S (eds) Distributed, ambient and pervasive interactions: understanding humans. Springer-Cham, Las Vegas, pp 21–34. https://doi.org/10.1007/978-3-319-91125-0_2

[19] Scherer M (2017) Performance and scalability of blockchain networks and smart contracts. Master’s thesis, Umeåe University,Department of Computing Science, Sweden. https://www.divaportal.org/smash/get/diva2:1111497/fulltext01.pdf

[20] Davis L (1991) Handbook of genetic algorithms, 1st edn. Van Nostrand Reinhold, New York

[21] Maulik U, Bandyopadhyay S (2000) Genetic algorithm-based clustering technique. Pattern Recogn 33(9):1455–1465. https://doi.org/10.1016/S0031-3203(99)00137-5

[22] Li J, Wu J, Chen L (2018) Block-secure: Blockchain based schemefor secure P2P cloud storage. Inf Sci 465:219–231. https://doi.org/10.1016/j.ins.2018.06.071

[23] Hussein AF, ArunKumar N, Ramirez-Gonzalez G, Abdulhay E,Tavares JMR, de Albuquerque VHC (2018) A medical recordsmanaging and securing blockchain based system supported by agenetic algorithm and discrete wavelet transform. CognSyst Res52:1–11. https://doi.org/10.1016/j.cogsys.2018.05.004

[24] Bano S, Sonnino A, Al-Bassam M, Azouvi S, McCorry P, Meiklejohn S, Danezis G (2017) Consensus in the age of blockchains. arXiv:1711.03936

[25] Xiao Y, Zhang N, Lou W, Hou YT (2020) A survey of distributed consensus protocols for Blockchain networks. IEEE CommunSurv Tutor 22(2):1432–1465. https://doi.org/10.1109/COMST.2020.2969706

[26] Castro M, Liskov B (2002) Practical byzantine fault tolerance and proactive recovery. ACM Trans ComputSyst 20(4):398–461.https://doi.org/10.1145/571637.571640

[27] Androulaki E, Barger A, Bortnikov V, Cachin C, Christidis K, Caro AD, Enyeart D, Ferris C, Laventman G, Manevich Y, Muralidharan S, Murthy C, Nguyen B, Sethi M, Singh G, Smith K, Sorniotti A, Stathakopoulou C, Vukoli´c M, Cocco S. W, Yellick J (2018) Hyperledger fabric: A distributed operating system for permissioned blockchains. In: Proceedings of the 13th EuroSys conference, pp 1–15. https://doi.org/10.1145/3190508.3190538

[28] Herlihy M (1986) A quorum-consensus replication method for abstract data types. ACM Trans ComputSyst 4(1):32–53. https://doi.org/10.1145/6306.6308

[29] Ribeiro Filho JL, Treleaven PC, Alippi C (1994) Genetic-algorithm programming environments. Computer 27(6):28–43. https://doi. org/10.1109/2.294850

[30] Goldberg DE, Holland JH (1988) Genetic algorithms and machine learning. Mach Learn 3(2):95–99. https://doi.org/10. 1023/A:1022602019183

[31] Holland JH (1992) Genetic algorithms. Sci Am 267(1):66–73. http://www.jstor.org/stable/24939139

[32] Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection, 1st edn. MIT Press, Cambridge

[33] Michalewicz Z (1994) Conclusions. In: Michalewicz Z (eds) genetic algorithms + data structures = evolution programs. Springer, Berlin, pp 287–308. https://doi.org/10.1007/978-3-662-07418-3_14

[34] Simon D (2013) Evolutionary optimization algorithms, 1st edn. Wiley, New Jersey