Hindawi
Security and Communication Networks
Volume 2022, Article ID 1060765, 16 pages
https://doi.org/10.1155/2022/1060765

Research Article

A Novel Consensus Algorithm Based on Segmented DAG and BP Neural Network for Consortium Blockchain

Xiaohong Deng,[1,2,3] **Kangting Li**,[2] **Zhiqiang Wang**,[2] **and Huiwen Liu**[1]

[1]*School of Electronics and Information Engineering, Gannan University of Science and Technology, Ganzhou 341000, China*
[2]*School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China*
[3]*Key Laboratory of Cloud Computing and Big Data, Ganzhou 341000, China*

Correspondence should be addressed to Huiwen Liu; 9320070286@jxust.edu.cn

Received 13 February 2022; Accepted 21 March 2022; Published 8 April 2022

Academic Editor: Jiewu Leng

Currently, because of the excellent properties of decentralization, hard tamperability, and traceability, blockchain is widely used in WSN and IoT applications. In particular, consortium blockchain plays a fundamental role in the practical application environment, but consensus algorithm is always a key constraint. Over the past decade, we have been witnessing the obvious growth in blockchain consensus algorithms. However, in the existing consortium blockchain consensus algorithms, there is a limited characteristic of scalability, concurrency, and security. To address this problem, this work introduces a new consensus algorithm that is derived from a directed acyclic graph and backpropagation neural network. First, we propose a partitioned structure and segmented directed acyclic graph as data storage structure, which allows us to improve scalability, throughput, and fine-grained granularity of transaction data. Furthermore, in order to provide the accuracy of node credit evaluation and reduce the possibility of Byzantine nodes, we introduce a novel credit evaluation mechanism based on a backpropagation neural network. Finally, we design a resistant double-spending mechanism based on MapReduce, which ensures the transaction data are globally unique and ordered. Experimental results and security analysis demonstrate that the proposed algorithm has advantages in throughput. Compared with the existing methods, it has higher security and scalability.

1. Introduction

In 2008, Satoshi Nakamoto proposed Bitcoin for the first time, and then the digital currency represented by Bitcoin has developed rapidly and became an integral part of the digital finance field. Blockchain as a core technology of Bitcoin has received extensive research attention, expanding to other fields besides finance; for example, in Wireless Sensor Network (WSN) [1–3], almost all the scenarios of WSNs require an efficient and accurate localization process. However, the main disadvantage of the existing frameworks and algorithms is that they are not so much significant with the trust of the beacon nodes, which are an integral part of WSNs. This must be ensured for localization. At the same time, it is obvious to have the malicious nodes in the hostile environment of WSN operations. Literature [3] provides a secure localization scheme based on trust assessment for WSNs using blockchain technology. While addressing this challenge, it also provides a trust-based framework for secure localization. In the field of the Industrial Internet of Things (IIoT), blockchain technology has become a new way to solve cooperative trust issues. Using blockchain, a tamper-proof system can be built, which can be used as an audit tool for hardware products of the industrial Internet of Things from chip to whole equipment. Blockchain improves the productivity and operational efficiency of IIoT through a smart contract, allowing machines to manage themselves. To apply blockchain technology in IIoT, one of the main issues is to solve the security and efficiency problems of consensus protocols. Literature [4] proposes a reputation-based incentive module that can be implemented on state-of-the-art PoX protocols and can make the PoX protocols achieve better consensus states. This scheme can effectively encourage the cooperative behavior of the nodes in IIoT, which can benefit the network. At the same time, in the field of vehicle edge computing [5, 6], due to the tamper-proof, traceable, and distributed storage properties of blockchain, it can provide a reliable storage environment for its data. However, what affects the blockchain's real entry into practical application is the blockchain's infrastructure. 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 low performance of consensus mechanism, which leads to low operational efficiency of the whole blockchain system.

Consensus algorithm is the core technology in blockchain, used to solve how to reach an agreement between distributed nodes [7]. Most of the existing mainstream consortium blockchain consensus algorithms are based on Byzantine Fault Tolerance [7] (BFT). Although the BFT algorithm can solve the "Byzantine General" problem, it also brings new problems. For example, as the number of nodes increases, the communication complexity of PBFT [8] will increase rapidly, which will directly lead to a decrease in system throughput. In addition, the PBFT elects the leader node by trying to serially switch the number, which will greatly increase the possibility of a malicious node becoming a leader node and lead to poor system security. For this reason, HotStuff [9] uses threshold signature and star communication topology to solve the problem of high communication complexity of PBFT, but there is the possibility of malicious nodes becoming leader nodes, and the HotStuff algorithm adopts a star topology structure, so its algorithm performance is limited by leader node's hardware resource. Besides, Raft [10] uses the communication structure of master-slave nodes to make the system throughput higher under small-scale nodes, but there are problems such as non-Byzantine resistance and throughput limited by the hardware resources of the leader node.

In order to solve the problem of malicious nodes doing evil, researchers proposed to introduce a reputation mechanism into BFT algorithms. For example, Alex et al. [11] proposed a reputation consensus algorithm against "Sybil" attacks, which effectively reduces the risk of malicious nodes becoming leader nodes. However, the algorithm still does not solve the problem that the larger the size of the node, the lower the throughput, and the calculation of the reputation model is relatively fixed, so the scalability is poor. DE Oliverira et al. [12] proposed an adaptive hedging algorithm to change the calculation of the reputation model in a dynamic way. However, the algorithm has poor activity; that is, when the leader node goes down, the algorithm will run abnormally and it is difficult to restore to the normal operating state. In addition, the underlying data structure used by the above consensus algorithm is the traditional chain structure, which will limit the throughput of the system. Directed acyclic graph [13] (DAG) has the characteristics of adapting to high concurrency, which can better solve this problem and greatly increase the throughput of the system. Although traditional DAG has better performance in throughput, it also has problems such as double-spending and high retrieval complexity. In summary, the existing mainstream consortium blockchain consensus algorithms have problems such as poor throughput, poor scalability, and security risks.

In response to the above problems, we propose a consensus algorithm for consortium blockchain with low communication resource consumption, reliable performance, and easy scalability. In the following, the main contributions of this paper are mentioned:

(1) Propose a node reputation evaluation mechanism based on BP neural network, which can measure the node's credit value more accurately. And use the reputation value to select the accounting node to reduce the risk of malicious nodes becoming accounting nodes. In addition, select multiple nodes with higher reputation value to enter the committee to verify the accuracy of transaction messages to improve security.

(2) Design a partition structure for nodes to join and exit freely, which is used to solve the problem of poor scalability. Introduce a segmented DAG as the data storage structure solves the problem of poor throughput while reducing the complexity of retrieval in traditional directed acyclic graphs and improves the fine-grained nature of data operations by using transactions as the basic storage unit.

(3) A resistant double-spending mechanism based on MapReduce [14] is proposed to ensure that the data is globally unique. At the same time, it solves the poor scalability of the BFT consensus algorithm and the double-spending problem in DAG [15].

This paper is organized as follows: In Section 2, we briefly summarize the related knowledge, including the existing mainstream consensus algorithms, BP algorithm, MapReduce, and DAG. In Section 3, we describe the consensus algorithm proposed in this paper in detail, including the credibility evaluation model, the underlying topology, the election of the organizing committee members, and the consensus process. The experimental results and analysis are described in Section 4, which introduces the experimental environment, performance test results, security analysis, and comparison with other pieces of literature. Finally, in Section 5, we outline conclusions and future research directions.

2. Related Knowledge

2.1. 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, typical representative algorithms such as proof of work [16] (PoW) of the Bitcoin system, Proof of Stake [17] (PoS) of Nextcoin, and the Proof of Delegated [18] (DPoS) of EOS v1.0 [19]; the second is the node voting system, typical representative algorithms such as PBFT of Fabric v0.6 [20]; the third is the Paxos-like consensus algorithm, typical representative consensus algorithms such as Paxos [21–23] and Raft of Fabric v1.4.4.

In particular, Paxos is the origin of traditional distributed algorithms. Many consensus algorithms are based on their evolution and development, and Raft also evolved from this idea. However, both Paxos and Raft do not have anti-Byzantine characteristics, so they are not suitable for the blockchain environment. 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). However, HotStuff relies on a synchronized clock to integrate switching views and the consensus process. When the verifier raises an objection during the consensus process, that is, there is a problem in the authentication procedure, the view will be switched after the time expires. Both of the above BFT algorithms' leader nodes are elected according to the view number order for switching. Unfortunately, this way will have the problem of poor security.

Interestingly, the emergence of reputation-based consensus algorithms has solved the problem of leader node election. Literature [12] proposed a new model to replace the proof of work to form a consensus group. The proposed model uses an adaptive hedging method to calculate reputation values for nodes that want to participate in the consensus committee and select nodes with higher reputation values for the consensus committee to reduce the chance of evil nodes. But this method does not take into account the problem of algorithm activity.

Besides, Liu et al. [24] 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 [25] 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 BFT is 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 [26] proposed a PoS consensus mechanism based on reputation. Aiming at the problems of low performance and low security of existing blockchain, 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 election, or election based on some attribute values. In particular, 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 take into account the performance of the entire network.

2.2. Backpropagation Algorithm (BP) [27]. As the core of deep learning [28–30], 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. The 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. The connection of neurons between layers is the weight w, and the target of

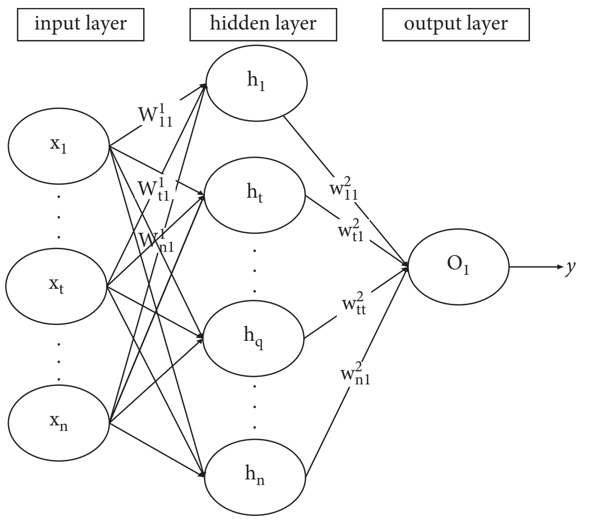

FIGURE 1: Model of BP neural network.

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

$$y = \sum_{i=1}^{n} h_i \times w_{i1}^{2}. \tag{1}$$

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

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

Among them, σ is the activation function, and the common activation functions are ReLU, tanh, sigmoid, and so on. The activation function makes the neural network have a nonlinear fitting ability. X_j is the input value.

And then, the calculation formula of the loss value is shown as follows:

$$\text{loss} = (y - \hat{y})^2. \tag{3}$$

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).

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

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).

$$\Delta w_{11}^{1} = \frac{\partial\,\text{loss}}{\partial O_1}\frac{\partial O_1}{\partial h_1}\frac{\partial h_1}{\partial w_{11}^{1}}. \tag{5}$$

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.

2.3. 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 reexecute 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. Since MapReduce 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.

2.4. 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. However, in a distributed ledger based on DAG, as shown in Figure 3, the basic unit of each ledger can reference one or more predecessor units and can be referenced by one or more subsequent units at the same time. As such, this structural difference enables DAG-based ledgers to support concurrent operations, and multiple nodes can add transactions or block units to the ledgers at the same time, thereby greatly improving system throughput. However, although the traditional DAG has better performance in throughput, there are other problems. For example, using the Iota [31] framework with DAG as the bottom layer, transactions in this framework require a large number of Markov Monte Carlo random walks and a small amount of proof of work to add to the ledger. This method is too complicated, and the transactions in Iota are not globally ordered, so it cannot completely resist the double-spending problem, and the retrieval time is long. In addition, another HashGraph [32] framework that uses a parachain 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. In summary, the existing DAG framework has

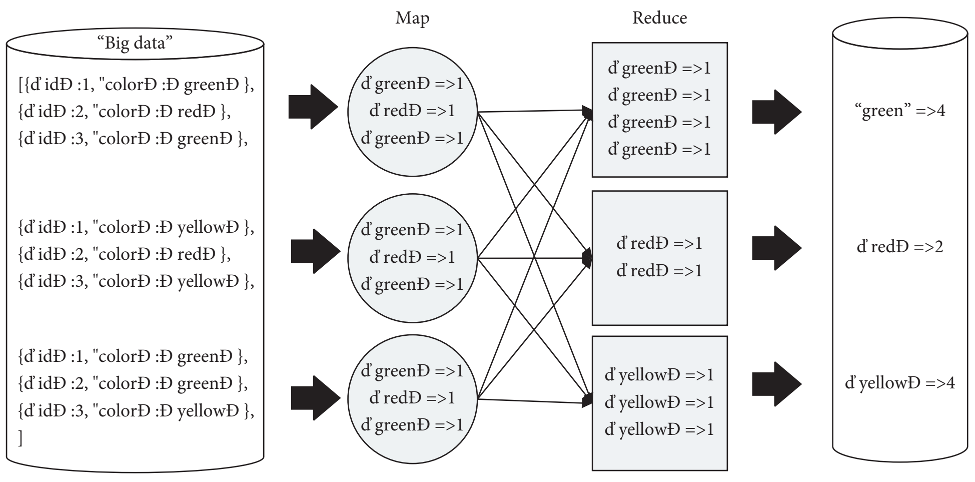
Map
Reduce
"Big data"
[{ď idĐ :1, "colorĐ :Đ greenĐ },
{ď idĐ :2, "colorĐ :Đ redĐ },
{ď idĐ :3, "colorĐ :Đ greenĐ },
{ď idĐ :1, "colorĐ :Đ yellowĐ },
{ď idĐ :2, "colorĐ :Đ redĐ },
{ď idĐ :3, "colorĐ :Đ yellowĐ },
{ď idĐ :1, "colorĐ :Đ greenĐ },
{ď idĐ :2, "colorĐ :Đ greenĐ },
{ď idĐ :3, "colorĐ :Đ yellowĐ },
]
ď greenĐ =>1
ď redĐ =>1
ď greenĐ =>1
ď greenĐ =>1
ď redĐ =>1
ď greenĐ =>1
ď greenĐ =>1
ď redĐ =>1
ď greenĐ =>1
ď greenĐ =>1
ď greenĐ =>1
ď greenĐ =>1
ď greenĐ =>1
ď redĐ =>1
ď redĐ =>1
ď yellowĐ =>1
ď yellowĐ =>1
ď yellowĐ =>1
"green" =>4
ď redĐ =>2
ď yellowĐ =>4

Figure 2: Working principle of MapReduce.

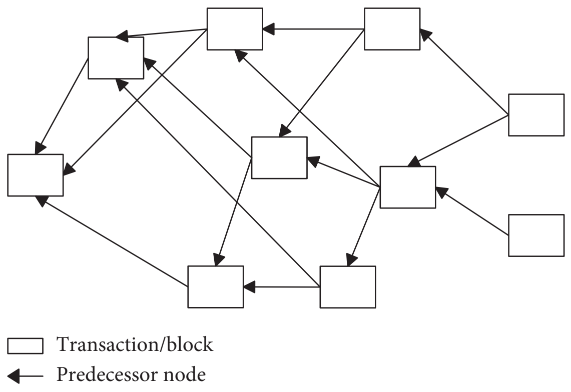

Figure 3: Architecture of directed acyclic graph.

problems such as double-spending, high search complexity, and long time to add to the ledger. We propose a segmented DAG to solve the above problems, the details of which will be introduced in subsequent chapters.

2.5. Attack Model. The blockchain system is a network composed of cooperation between nodes. In order to subdivide the functions of the blockchain system, the blockchain is usually divided into a six-layer structure, which includes data layer, network communication layer, consensus layer, incentive layer, contract layer, and application layer. The proposed algorithm in this paper mainly involves the data layer, network communication layer, and consensus layer. The data layer is mainly based on a certain data structure to store data, the network communication layer is responsible for broadcasting and verifying transactions, and the purpose of the consensus layer is to allow nodes to coordinate and cooperate to achieve a consensus on data consistency. This paper mainly discusses the attack methods involved in these three layers. Common attack methods [33] are as follows:

(1) Double-spending attacks: the main situations of common double-spending attacks are as follows: (1) When a new transaction enters the block to obtain a sufficient number of 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) For the Naive DAG, when the transaction enters the ledger, it relies on the PoW algorithm to calculate the weight to choose to eliminate the double-spending transaction, but there is a situation that is not timely.

(2) 51% attack: for this paper, an attacker who has more than half of the reputation value is 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.

3. Proposed Algorithms

3.1. Reputation Evaluation Model. In the traditional consortium blockchain consensus algorithm, the nodes participating in the consensus generally switch sequentially or randomly, which easily leads to malicious nodes deliberately doing evil. In the following research, the reputation consensus algorithm has been proposed, node's reputation is measured by the behavior of nodes participating in the consensus, and the accounting nodes are selected in turn by the size of the reputation value. In this way, it better solves the problem of nodes doing evil. However, most of them use simple linear formulas to evaluate the reputation of nodes. Unfortunately, linear algorithms cannot effectively extract the behavioral characteristics of nodes, so they cannot make full use of the characteristics and assign corresponding weights. For this reason, we propose a reputation evaluation model based on the BP algorithm. Since the neural network can approximate the properties of any function from arbitrary precision [34], it has strong feature extraction capabilities. We use some attributes of the node as feature vectors to consider the reputation value of the node. The reputation value is used to quantify the possibility that the node is a Byzantine node. The node attributes are shown in Table 1.

The characteristics of nodes in the blockchain mainly include two aspects. (1) Security features: they contain the number of maliciously sending false transaction messages E_Tx, the node's historical reputation value H_Rep, and the node's online time On_Time. (2) Performance characteristics: they contain the throughput TPS of the node, the number of effective forwarding transactions C_ETx of the node, and the average delay forwarding time D_AFT, where E_Tx, C_FTx, and C_ETx are discrete values, and the rest are continuous values.

In this paper, we constructed a four-layer BP neural network, in which the input layer contains six neurons, corresponding to six features, and the two-layer hidden layer contains 1,000 neurons. The activation function uses the ReLU function, and the last output layer uses the sigmoid function to map the reputation value between 0 and 1. Construct reputation evaluation as shown in formula (6).

$$rep_j^i = \begin{cases} F\left(X_j^{i-1}\right) rep_j^{i-1} \geq 0 \text{ and evil} = 0, \\ -\sum_{i=0}^{\text{latest}} rep_j^i rep_j^{i-1} \geq 0 \text{ and evil} = 1, \\ rep_j^{i-1} + \alpha e^{-\text{count}} rep_j^{i-1} < 0, \end{cases} \tag{6}$$

where evil = 0 means that the node is not doing evil and evil = 1 means that the node is doing evil. rep_j^i is the reputation value of the j node in round i, and X_j^{i-1} is the feature vector of the j node in round i-1. X_j = [E_Tx,H_Rep, On_Time,TPS,C_ETx,D_AFT]; X_j needs to undergo dimensionless processing. F(x) is the neural network model, and when rep_j^{i-1} is greater than or equal to 0 and the node is not doing evil, use F(x) to predict the reputation value of the node. However, when rep_j^{i-1} is greater than or equal to 0 and the node has malicious behavior, the node's reputation is the negative number of the node's accumulated reputations from the first time to the latest; in contrast, when rep_j^{i-1} is less than 0, the node's reputation is calculated by adding an exponential function related to the number of evils and the reputation of the previous round, where $\alpha \in (0, 1)$, and count is the number of evils.

3.2. Underlying Topology

3.2. Underlying Topology. Compared with the serial processing of chain structure, DAG is more suitable for natural high concurrency. We propose a segmented DAG to solve the problem of high retrieval complexity and long time for transactions to be added to the ledger. As shown in Figure 4, the shaded block is the organizing committee block, which contains the organizing committee's group signature, timestamp, reputation record, and hash pointer group. The white blocks represent ordinary nodes, which contain the signatures, timestamps, transaction information, and hash pointer groups of ordinary nodes. The numbers in the grey and white blocks represent the sequence. The black block is a fast index block array, which contains a timestamp, a hash pointer, and the block hash of its own block.

The genesis block 0 is fixedly generated as organizing committee block, and subsequent blocks are connected to it by hash pointers. The connection method is to randomly select the nearest n timestamp transaction data. A fixed organizing committee block is generated every fixed time or the corresponding number of blocks. The transaction information between two organizing committee blocks is verified and deduplicated by the former. In addition, in terms of retrieval, compared to the retrieval of all transactions in the original DAG, we divide the DAG according to the time dimension, only relying on the black block to quickly index according to the timestamp. As such, the search complexity is greatly reduced.

DAG has two forms in physical structure: one is an adjacency matrix, and the other is an adjacency list. Since the adjacency matrix is a sparse matrix, it will waste a lot of space, so the storage form of the adjacency list is adopted. As shown in Figure 5, the leftmost is an array of fast index blocks, which contains timestamps and hash pointers, the grey part in the middle is the committee block, and the white part is the original block. Both the committee block and the white block contain transaction information, hash value, and hash pointer and are connected to the corresponding transaction information block.

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 a number of 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 [35].

3.3. Consensus Algorithm Process

3.3.1. Algorithm for Election of Organizing Committee Members. The method for electing members of the organizing committee is shown in Algorithm 1. The members

Table 1: Characteristic attributes of node.

Characteristic symbol	Explanation	Value range
E_Tx	Number of maliciously sending false transaction messages	$[0, +\infty]$
H_Rep	Node's historical reputation	$[-\infty, 1]$
On_Time	Node's online time	$[0, +\infty]$
C_ETx	The number of effective forwarding transactions by the node	$[0, +\infty]$
D_AFT	Average delay forwarding time	$[0, +\infty]$
TPS	Node's throughput	$[0, +\infty]$

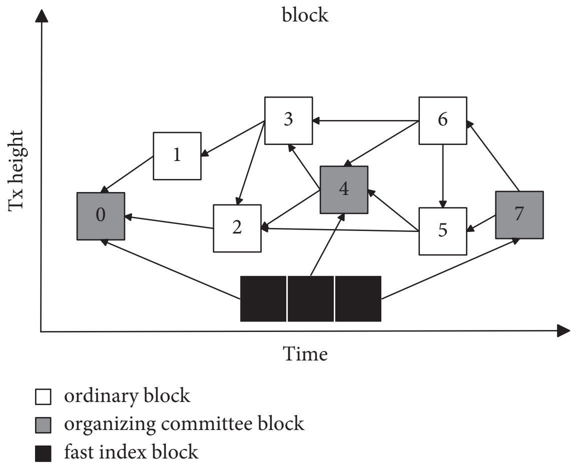

Figure 4: Topology diagram of segmented DAG.

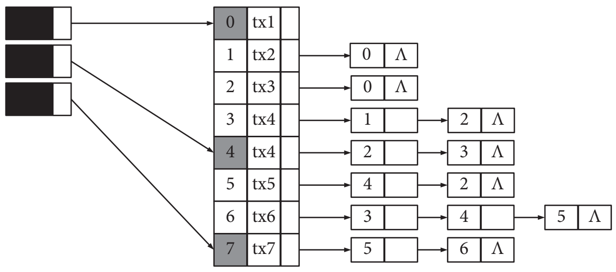

Figure 5: The storage structure of the segmented DAG.

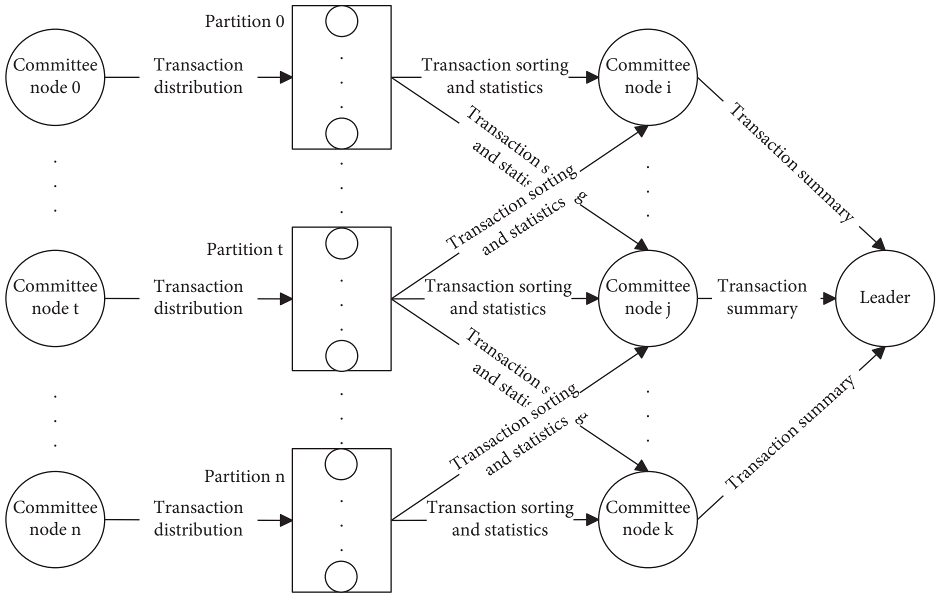

Figure 6: Resistant double-spending mechanism based on MapReduce.

who enter the committee in the first round are fixedly selected. Besides, the subsequent election of the organizing committee can be described as follows. First, input the characteristics into formula (6), and obtain the reputation of each round of nodes; second, rank the reputation, and the leader of the organizing committee uses a verifiable random function [36] (verifiable random functions (VRF)) to generate verifiable random numbers, which are used to randomly select nodes with higher reputation values. Finally, the nodes in the group verify whether the random number and the binomial distribution pass. If the verification of the VRF function fails or overtime, the organizing committee is requested to perform random sampling again and regenerate the random number through VRF; if it passes, the organizing committee will synchronize the data of random numbers and select several nodes to enter the organizing committee.

3.3.2. Consensus Process. In this paper, nodes are divided into ordinary nodes and organizing committee nodes, and the reputation value of the node is obtained by the reputation model. The overall consensus process is shown in Algorithm 2. First, the organizing committee randomly elects a leader node and preselects a backup leader node. Generally, the organizing committee selects the backup leader with the highest reputation value. The backup leader node is normally responsible for collecting transaction messages and monitoring the status of the leader and follower nodes. Once the leader node goes down or acts maliciously, the backup node starts to take over. The remaining nodes are follower nodes, which are responsible for collecting and verifying transaction data and sorting them. After a period of time, the results are returned to the leader node. Second, according to the results, the leader node removes the double-spending data according to the entry degree and distributes the relevant information to the organizing committee nodes to wait for a reply. Last, if more than 1/2 of the reputation node replies are received (the reputation value of 1/2 is the maximum error tolerance threshold of this algorithm, which will be proved by subsequent experiments), then gossip protocol broadcasts to other ordinary nodes to achieve global unification. In contrast, if not received or timed out, immediately switch the leader to enter the next round. Or when the term of the committee is reached, all the members of the organizing committee will be replaced; else go directly to the next round.

3.3.3. 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 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 and 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 leader. Finally, the leader node receives the message of the organizing committee node, performs verification, packs, and returns the DAG structure transaction message.

4. Experimental Results and Analysis

4.1. Experimental Environment. In order to test the performance of the consensus algorithm proposed in this paper, we designed several simulation experiments. The operating system selected for the experimental environment is centos, using Python and PyTorch to write consensus algorithms, using flask to write web program interfaces, and using docker containers to load web programs to simulate nodes. In addition, we used Alibaba Cloud server to simulate the experiment of multimachine multinode stress test, used siege to carry out stress test and throughput, and set the concurrency to 280 transactions/s. It should be noted that, without a special statement, we set the number of organizing committee nodes to one-half of the number of summary points, set the appointment period of the organizing committee to 2 min, and set the delay waiting threshold to be random between 3 s and 4 s. In order to better carry out quantitative experiments, we used transactions with tags. The transaction tags are normal, empty, malicious, and double spend, which represent normal transactions, empty transactions (heartbeat packets), malicious transactions, and double spend transactions, respectively. The purpose of the experiment is to test the performance and security of the consensus algorithm in an ideal environment or a Byzantine environment. The performance includes the error of the neural network, the ability to process transactions in the consensus process, and the delay in completing the consensus. On the other hand, the purpose of security is to test the system operation under the condition of a hypothetical adversary attack.

4.2. Performance Test

4.2.1. Regressor Performance. To test the accuracy of the regressor, 500 node's index data and their reputation evaluation results are selected. According to the method of the reputation evaluation model, the selected index data is first standardized to facilitate the neural network processing. More precisely, the data of 400 nodes are used as training data each time, and the data of the remaining 100 nodes are used as verification data. Besides, the learning rate is set to 0.001 in the experiment.

```
Input: X (the feature vector of the node)
Output: flag1 (whether the committee members are successfully elected)
 (1) Vec = formula(X)# Get the reputation value of the node in each round
 (2) Sorted_Vec = Sort(Vec) # Reputation ranking
 (3)    flag1 = false
 (4) Random_number, proof = VRF(seed)# VRF function to generate random numbers and evidence
 (5) if(Verify(Random) = = True&&Verify(proof) = = True&&Time < Congfig_time): #Verify that the random number and evidence
     are correct at the specified time
 (6) flag1 = Choose(Random_number)# Select nodes to enter the committee and set flag to True
 (7) else:
 (8)        Go To Step4
 (9) end else
(10) end if
(11) return flag1
```

Algorithm 1: Election of organizing committee members.

```
Input: Random seed
Output: Consensus result flag2
 (1) (leader, leader_backs, follower) = VRF(seed)# Select leader node, backup node, and follower node
 (2) Tx_sorted = Sort(gather_follower(TX))# Collect verification transactions and sort them
 (3) Block = MapReduce(gather_follower(Tx_sorted))# Remove duplicate transactions and pack
 (4) flag2 = false
 (5) if(Collect(Block)≥1/2reputation&&Time < Congfig time) # Collect more than half of the reputation value at a fixed time
 (6)        flag2 = Broadcast(Block) # Broadcast block success is True
 (7)        if Check(Term of Service) = = True # Is it in the service cycle
 (8)              Go to step1 # Repeat step 1
 (9)        end if
(10)        else
(11)              Clean(committee)# Clearance Committee
(12)
(13)        end else
(14) else:
(15)     Change(leader) # Switch leader node
(16)     Go to Step1# Go back to step 1 and restart transaction collection
(17) end else
(18) end if
(19) return flag 2
```

Algorithm 2: Consensus process.

```
Input: transaction message msg
Output: the transaction set block and malicious node number after deduplication
 (1) msg_i = follower_i.watch(msg) i in range(0, n-1)# There are n organizing committee nodes to monitor transaction messages
 (2) msg_ij = Deivide(msg_i) j in range(0, m-1)# Divide the transaction message msg_i of each node into m parts
 (3) key,value = node_j.map(msg_ij) j in range(0, m-1) # Map the transaction message, use the transaction hash as the key, and the
     value is the reputation value of the node that sent the transaction
 (4) key_temp = md5(key) mod r # Do the md5 operation of the key and modulate r, and send the key and value to the corresponding
     follower_t
 (5) msg_t = follower_t.reduce(key_temp, value) t in range(0, r-1) #The follower node receives the corresponding transaction
     message, merges the messages, shuffles according to the size of the value, removes the double-spending operation, and
     returns the deduplicated transaction message
 (6) if msg_t is db_tx or error_msg:
 (7)    nodeid = t
 (8) end if
 (9) block = leader.collect(msg_t) # The leader node accepts the follower node message and performs verification and packaging
(10) return block, nodeid # Return DAG transaction message
```

Algorithm 3: Resistant double-spending mechanism based on MapReduce.

The change of loss function is shown in Figure 7. As can be seen from the data in Figure 7, as the number of pieces of training increases, the loss function has a sharp downward trend and finally tends to a stable value of about 0.1. However, the loss on the verification set is stable at around 0.067. These results provide substantial evidence for the original assumptions that the neural network model has learned the corresponding features and the error with the true value is small.

4.2.2. Throughput Test. In order to test the throughput of different consensus algorithms in a multimachine multinode environment, four hosts are configured with 25, 50, 75, 100, 125, 150, 175, and 200 docker simulation nodes. Each consensus algorithm takes the average value of the transactions per second (TPS) of 100 rounds of consensus. That is, the total transaction volume in 100 rounds divides the time taken for 100 rounds of consensus.

The test results are shown in Figure 8. The results reveal that the four algorithms all increase the TPS when the node size is less than 100. The proposed algorithm is slightly worse than HotStuff and Raft, but better than PBFT. However, when the node size is greater than 100, the TPS of PBFT decreases rapidly, while the TPS of Raft and HotStuff grows more slowly, and the proposed algorithm grows approximately linearly and is better than the other three algorithms. This is because, in the case of a smaller scale, the star topology used by Raft and HotStuff is faster. However, as the scale becomes larger, the performance of PBFT becomes lower and lower as the communication becomes more complicated. Since HotStuff and Raft use a star topology communication method, the throughput of the entire system is limited by the IO device of the master node. In particular, when the number of nodes increases, once the traffic exceeds or reaches the maximum processing capacity of the main node's IO, TPS will begin to decrease. In contrast, the performance of the proposed consensus algorithm will increase linearly with the increase in the number of nodes. This is because the MapReduce architecture is used to process tasks on multiple nodes, which greatly weakens the hardware performance limitations of a single node.

4.2.3. Response Time. Generally, the delay of the blockchain system can be defined as the time difference between the client submitting the transaction request and the client receiving the response result. In the experiment, we tested the delays of PBFT, the proposed algorithm, Raft, and HotStuff, when the number of nodes in the whole network is 25, 50, 75, 100, 125, 150, 175, and 200, respectively. Each algorithm tests the average delay of 100 rounds of consensus results.

The result is shown in Figure 9. Figure 9 illustrates that the delay of PBFT increases sharply with the increase of the number of nodes, while the proposed algorithms, Raft, and HotStuff keep the delay low. The reason for the above phenomenon is that the PBFT communication time is O(n2), so as the number of nodes increases, the access delay will increase greatly. Since both HotStuff and Raft use a star

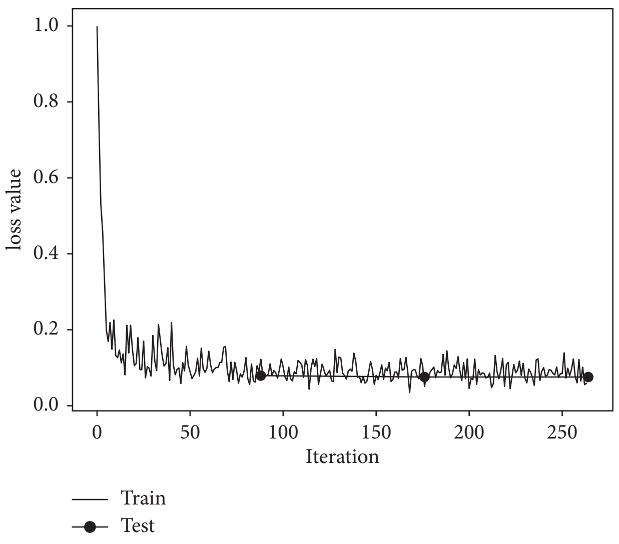

Figure 7: Change curve of loss function.

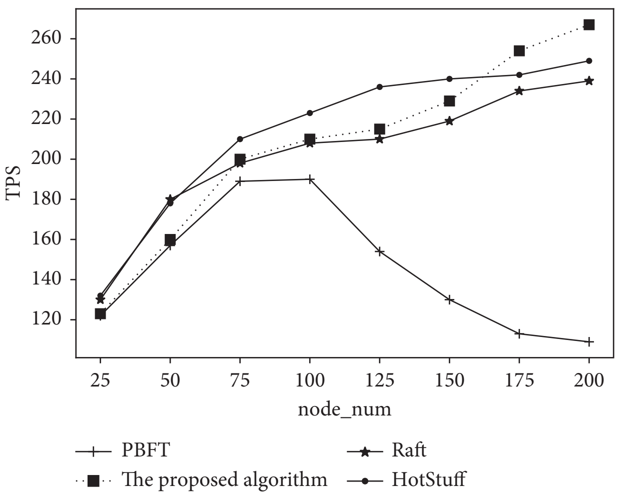

Figure 8: Throughput comparison.

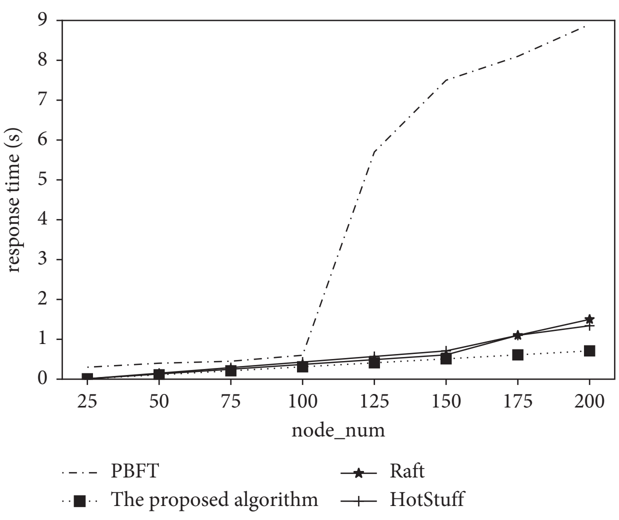

Figure 9: Algorithm delay comparison.

topology for communication, the client and the leader directly perform read and write operations, this method will enter a bottleneck period after being limited by hardware resources, and the delay will slowly increase in the later period. However, the algorithm in this paper has a small workload of the leader node due to the simultaneous write mechanism of the partitions. Therefore, it has lower latency.

4.2.4. Comparison of Retrieval Speed of Different Storage Structures. In this paper, the retrieval transaction scale is set between 0 and 20,000 transactions, and the purpose is to compare the retrieval speed in different storage structure scenarios of chain structure, simple DAG, and segmented DAG structure. The experimental results are shown in Figure 10.

As can be seen from Figure 10, the retrieval speed of traditional chained and naive DAGs increases linearly with the number of transactions. The segmented DAG retrieval does not follow the trend of linear growth in the number of transactions. The reason for the above phenomenon is that the chained data structure is searched in order. Even if the tree search method is used, it is limited to within the block. When searching for a transaction, the external memory needs to be transferred into the memory, so it takes longer. Simple DAG requires a BFS or DFS search method, so with the increase of transaction messages, the retrieval time will also increase accordingly. However, the segmented DAG used in this paper is stored in the form of a hash table and can be retrieved in chronological order, so the retrieval time is greatly reduced.

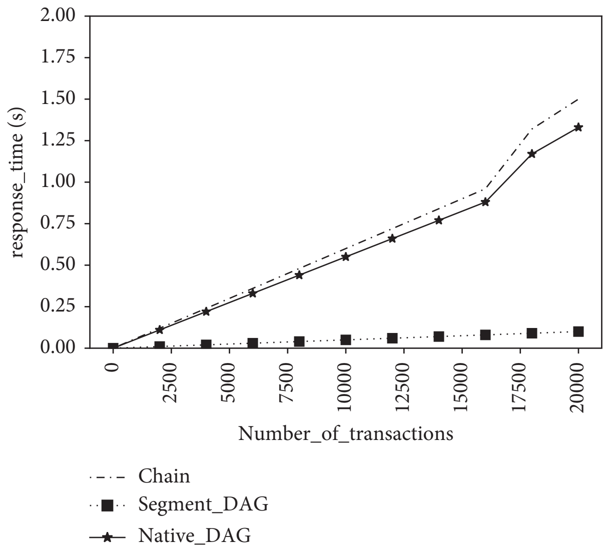

Figure 10: Comparison of retrieval speed of different storage structures.

4.3. Security Analysis

4.3.1. Attack Model Analysis

(1) Double spend attack. As mentioned in Section 2.5, in view of the first case, the underlying DAG topology we proposed will not have chain bifurcation, and there will be no two segmented DAGs at the same time. Only when a solar eclipse attack occurs, will the network splits make two segmented DAGs. In this case, you need to master the 50% reputation value; however, it is almost nonexistent in the subsequent experimental verification. For the second case, we designed the MapReduce architecture to ensure that the transactions in the DAG will be deduplicated and then sorted. In addition, the leader node will perform deduplication again, so the second case can also be avoided.

(2) 51% attack and eclipse attack (reputation cumulative split attack).

The segmented DAG structure we used is still updated based on the longest DAG structure. Assuming that the solar eclipse attack is successful, there will be multiple DAGs in an asynchronous environment. This phenomenon shows the Poisson distribution [16]. At a specific time, something will happen randomly at any time. More precisely, when this time period is divided into very small time slices, it can be considered that, within each time slice, the event may or may not happen. However, it is almost impossible to consider situations that occur more than once because the time slice can be divided into small enough time slices. The Poisson distribution formula is as follows:

$$P(X = i) = \binom{n}{i}\left(\frac{\lambda}{n}\right)\left(1 - \frac{\lambda}{n}\right)^{n-i}. \tag{7}$$

In formula (7), when $n \longrightarrow \infty$, $\binom{n}{i}/n^i \longrightarrow 4/i!$, $(1 - (\lambda/n)) \longrightarrow e^{-\lambda}$, it can be derived as formula (8).

$$\begin{aligned} P(X = i) &= \binom{n}{i}\left(\frac{\lambda}{n}\right)^{i}\left(1 - \frac{\lambda}{n}\right)^{n-i}, \\ &= \frac{e^{-\lambda}\lambda^{i}}{i!}. \end{aligned} \tag{8}$$

Formula (8) represents the probability that the attacker succeeds in the i block, where $i = z$, $\lambda = q_z$, and the calculation method of q_z is shown in formula (9). Among them, q is the probability of an attacker's successful attack, and p is the probability of an honest node that normally generates a transaction block.

$$q_z = \begin{cases} 1, \text{ If the attacker is the longest chain}, \\ \left(\dfrac{q}{p}\right)^{z}, \text{ If the attacker is behind by z blocks}. \end{cases} \tag{9}$$

We set the organizing committee node groups with 10%, 33%, 49%, and 50% reputation values to exist in an asynchronous network environment and set z from 0 to 50 and then try to attack the segmented DAG in the state of the real environment.

The results of the experiment are shown in Figure 11. As can be seen, the node group with 10% and 30% reputation values may have a larger drop and approximate to 0 as more blocks fall behind; the attacker with 50% reputation value will have a successful attack. In addition, the attacker with 49% reputation value tends to 0 as more blocks are created. As such, we set the threshold of reputation value to 50% in Algorithm 2 because mastering 51% reputation value will attack successfully and split the network. However, this is almost impossible. First of all, in the procedure of the election of the group committee, any normal node can enter the group committee, and controlling the group committee is almost to control all the nodes. Secondly, the communication between the nodes is local P2P, so the link is multichannel, and the possibility of splitting the network with the increase of the group committee nodes is almost 0, so this situation is almost impossible to exist.

(3) Denial of service attack.

In the experiment, we tested the reputation value changes of four nodes, node 1 to node 4, with different attribute values, and the results are shown in Figure 12.

More specifically, set the adjustable parameter α to 1, and set node 1 and node 2 to maintain good and medium characteristic attribute values, respectively. Besides, node 3 transforms from relatively medium attribute values (approximates attribute values of node 2) to better attribute values (approximates attribute values of node 1), and node 4 is set as a malicious node and begins to deny service in the fourth round. As the results of Figure 12 show, the reputation value of node 1 and node 2 has no obvious change, while the reputation value of node 3 is slowly increasing. In the fourth round, node 4 sends malicious information, and its reputation value changes to the inverse of the sum of its historical reputation values. The reason for the stable changes in the reputation value is because the neural network predicts the reputation value with high accuracy, and the reputation value results brought by similar feature attribute values are all approximately the same. Particularly, nodes with good characteristic attribute values will be given corresponding good scores, so the increase or decrease of reputation value will not be obvious, so it has good stability. In contrast, for the calculation of the reputation value of a malicious node, its reputation value changes directly to the inverse of the sum of its historical reputation values. In addition, as the number of malicious actions increases, an exponential function is used to calculate the follow-up reputation value, and the follow-up reputation growth rate is approximately zero. Undoubtedly, nodes with a reputation value less than 0 will not have the right to participate in the organizing committee and sort and count transactions.

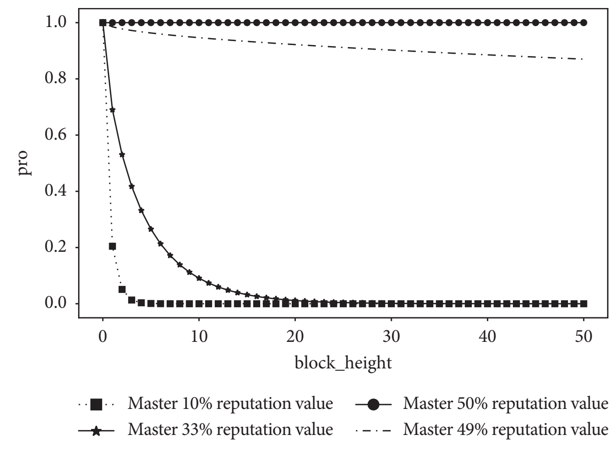

Figure 11: Cumulative reputation attacks.

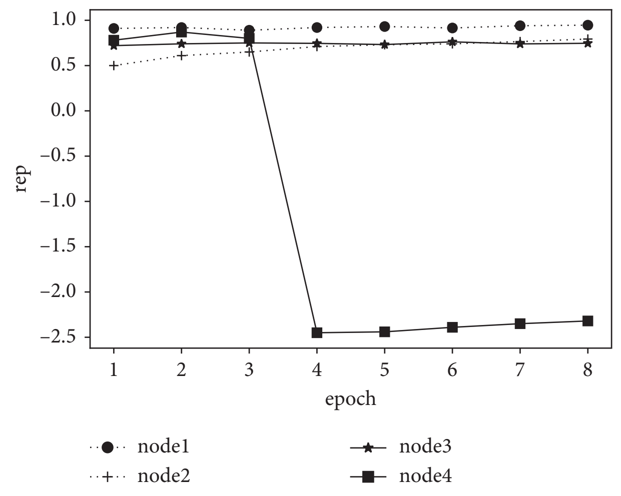

Figure 12: Change curve of node reputation.

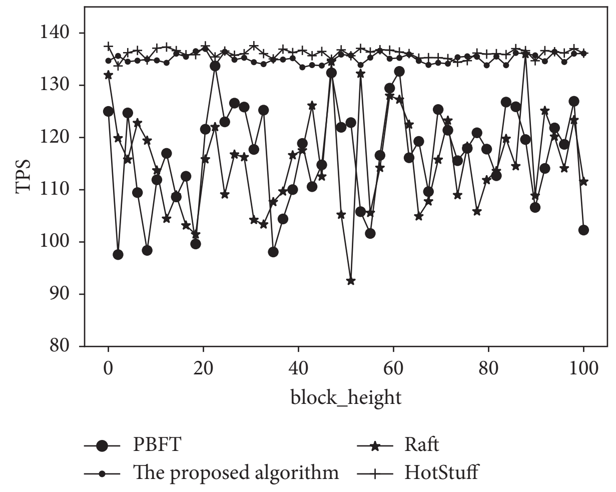

Figure 13: Continuous leader switching.

4.3.2. Continuous Switching of Malicious Leader Nodes. In the experiment, 25 nodes were set up to test the stability of the throughput of each consensus algorithm under the condition of continuous switching of the leader node.

As shown in Figure 13, the throughput of the PBFT and Raft fluctuates drastically, and the lowest is only 91TPS. In contrast, the throughput of HotStuff and the proposed algorithm has good stability and has been maintained at around 130TPS. This is because the traditional PBFT algorithm switches the view after the voting is completed. Due to the multistage point-to-point communication, the complexity is O(n2). In particular, if the leader switches continuously, the communication complexity will be as high as O(n3). As Raft, it switches the leader node based on the heartbeat packet, frequent switching will lead to a continuous election process. However, HotStuff integrates view switch and transaction broadcast communication and uses the pipeline block topology, so it has a better stability. In this paper, the direct switching mechanism between the backup node and the organizing committee node is used to reduce the risk of malicious downtime and switching of the leader node, and the node with a high reputation value is selected as the organizing committee node to further ensure stability.

4.3.3. Expulsion of Malicious Nodes. In the experiment, 100 nodes were set up to test the rate of the proposed consensus algorithm to eliminate malicious nodes, and the running time of the system was set to ten minutes. Malicious nodes are randomly distributed in the network, and the ratio is set to 10%, 20%, 30%, 40%, and 50% of the network nodes. Moreover, they sent double-spending and error messages with a probability of 25%, 50%, 75%, and 100%, respectively. The experimental results are shown in Figure 14. The results indicate that no matter what the ratio of malicious nodes is, the eviction ratio will rise sharply from 0.2 to 0.98 before the ratio of malicious messages reaches 0.75, until 100% eviction, which has nothing to do with the ratio of malicious nodes. The reason can be further described in Figure 15. With the proportion of malicious message sending being 0.25, the proportion of transaction messages is only about 0.42 if the proportion of valid transactions with double-spending is more than 0.5. This is because, in this paper, we consider that as long as the percentage of valid transactions is more than 0.5, even if a double spend transaction is sent, it is considered normal, so the eviction rate is low. Once a malicious transaction is detected, the node will be directly eliminated, and the eviction rate will increase as the proportion of malicious messages increases, so the final approach is approximately 100%.

The results of the expulsion velocity experiment are shown in Figure 16. As can be seen that when the malicious message ratio is 0.25, the node expulsion rate is about 13 s. It should be noted that the larger the scale of the malicious node is, the longer the time it takes. When the malicious message ratio is 0.5, the expulsion rate is about 30 s, and when the ratio of malicious transactions is 0.75 and 1, the time consumption will decrease. The main reason is that when the proportion of malicious messages is 0.25, 42% of nodes will send double-spending messages. However, the inspection mechanism needs to wait for the effective ratio to be lower than 0.5 before removing them, so waiting time is required. When the proportion of malicious messages is 0.5, the nodes have some randomness, and the possibility of malicious nodes sending double-spending transactions is stronger, which results in higher time-consuming. In particular, when the proportion of malicious messages is 0.75 and 1, malicious nodes are more likely to send malicious

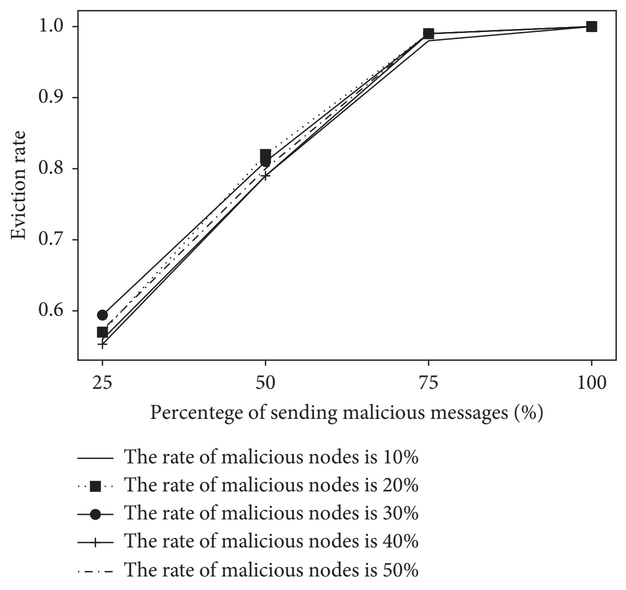

Figure 14: Rate of exclusion of malicious nodes.

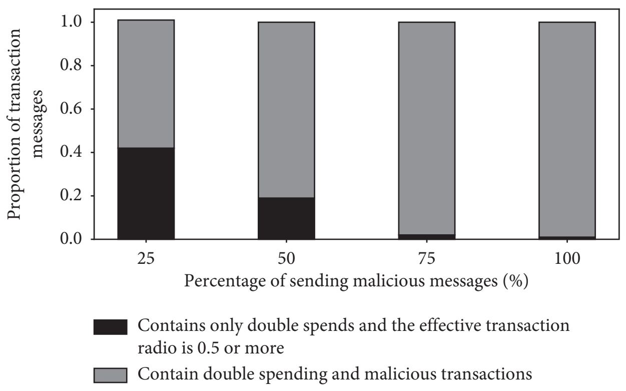

Figure 15: Malicious message ratio.

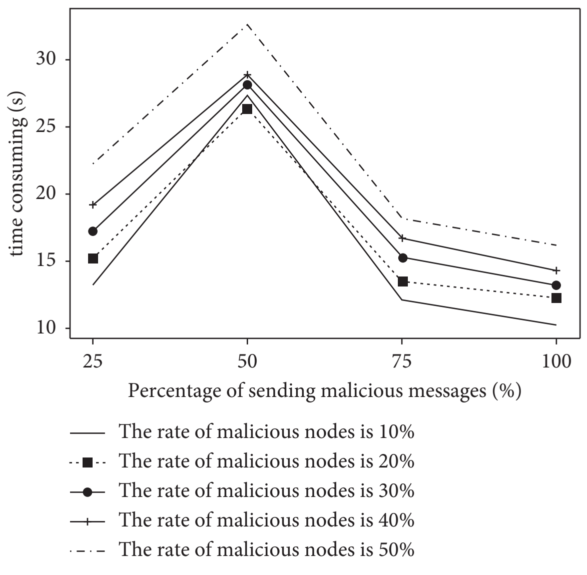

Figure 16: Average velocity of eviction of malicious nodes.

Table 2: Comparison results with existing literature.

	Scalability	Fault tolerance (%)	Underlying topology	Active	High concurrency	Resistant double-spending	Communication complexity	Fine-grained
Literature [12]	Low	49	Chain	No	No	High	O(n2)	Block
Literature [9]	High	33	Chain	Yes	Yes	High	O(n)	Block
Literature [26]	High	49	Chain	No	No	High	O(n2)	Block
Literature [24]	Low	33	Chain	Yes	No	High	O(n2)	Block
The proposed algorithm	High	49	DAG	Yes	Yes	High	O(n2)	Transaction

messages directly. As long as the system detects malicious messages, it will directly eliminate them, so the time is shorter.

4.3.4. Comparison with Other Pieces of Literature. Generally, consensus algorithms are usually compared from three aspects, namely, the degree of decentralization, security, and performance. Among them, the degree of decentralization is the scale of nodes participating in the consensus, the security is mainly anti-Byzantine ability and resistant double-spending, and the performance mainly considers factors such as algorithm activity, throughput, communication complexity, and scalability. The proposed algorithm is compared with existing similar literature, and the comparison results are shown in Table 2. Literature [9], literature [12], literature [26], and literature [24] all use the traditional chained bottom topology, and the smallest unit of operation is a block, so their algorithms have good resistant double-spending ability. In terms of scalability, literature [9] adopts a pipelined block topology structure, and there is no mandatory sequence relationship between the generation of blocks, which enhances the scalability. In literature [26], multiple algorithms coexist, making nodes increase and exit free. However, literature [12] and literature [24] are both based on traditional BFT, so there is no improvement in scalability. In terms of fault tolerance, literature [9] and literature [24] are based on the maximum fault tolerance of BFT, that is, 33%. Particularly, the difference is that the threshold of literature [24] is 33% of the total reputation value; literature [12] uses a new examiner mechanism to control the threshold at 49%. Besides, the maximum threshold of literature [26] for a mixture of multiple proof algorithms is 49%. Regarding the complexity of communication, literature [12], literature [26], and literature [24] all use point-to-point propagation, so the communication complexity is O(n2). Literature [9] uses a star topology, so the communication complexity is O(n). In terms of concurrency, literature [12], literature [33], and literature [24] all use the traditional chain structure, so the concurrency is not high. In contrast, literature [9] adopts the pipeline mechanism, and there is no necessary time limit for the generation of the front and back blocks, so the concurrency may be higher. Compared with the above literature, we use DAG as the underlying topology, which naturally supports high concurrency and uses the MapReduce architecture to split tasks into multiple small tasks and send them to other nodes for sorting and deduplication, increasing the availability of ordinary nodes. Therefore, the scalability and resistant double-spending ability of the proposed algorithm is relatively high. Although the communication complexity of this paper is O(n2), the scale is just between the organizing committee nodes, the broadcast adopts the gossip protocol, and the communication complexity is O(n). In addition, this paper utilizes the similar functions of pacemakers to make the algorithm active and support semiasynchronous.

5. Conclusion

In summary, the consortium blockchain architecture has become the first choice for blockchain applications. However, limited by the traditional chain structure, the throughput of the blockchain has been greatly affected. Although the appearance of DAG increases the system throughput in a concurrent manner, it brings new problems of high algorithm complexity and double-spending. For this reason, we propose a high concurrency and scalable consortium blockchain consensus algorithm, which designs a segmented DAG structure to increase system throughput while reducing the time complexity of global retrieval. The resistant double-spending mechanism based on the MapReduce architecture effectively ensures the global uniqueness of the transaction. The consensus algorithm proposed in this paper is suitable for the parallel and collaborative computing of large-scale sensors in the Internet of Things, which improves the security of computing and the scalability of device clusters. In addition, the credible nodes are elected through the reputation model based on the BP neural network, which reduces the risk of malicious nodes doing evil. The simulation experiment results also prove that the algorithm in this paper has better performance. However, detailed theoretical proof was not obtained, and the following three aspects are worthy of in-depth study: (1) The underlying topology: the existing DAG has high concurrency and parallel characteristics better than the chain structure, but the security is poor, and the double-spending problem is difficult to solve. (2) Use the idea of division and autonomy to fragment the blockchain network, thereby reducing the communication scale of the network and increasing the speed of consensus. (3) Most of the existing consensus algorithms tend to adopt hybrid consensus algorithms, which will be a trend. Maybe, the integration of proof-like algorithms and BFT technology is a very meaningful research direction.

Data Availability

The coding data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 61762046 and 62166019) and the Science and technology research project of Education Department of Jiangxi Province (no. GJJ209412).

References

[1] J. Leng, G. Ruan, P. Jiang et al., "Blockchain-empowered sustainable manufacturing and product lifecycle management in industry 4.0: a survey," *Renewable and Sustainable Energy Reviews*, vol. 132, Article ID 110112, 2020.

[2] J. Leng, P. Jiang, K. Xu et al., "Makerchain: a blockchain with chemical signature for self-organizing process in social manufacturing," *Journal of Cleaner Production*, vol. 234, pp. 767–778, 2019.

[3] G. Rekha, K. Gulshan, and A. Mamoun, "A secure localization scheme based on trust assessment for WSNs using blockchain technology," *Future Generation Computer Systems*, vol. 125, no. 11, pp. 221–231, 2021.

[4] E. K. Wang, Z. Liang, C.-M. Chen, S. Kumari, and M. K. Khan, "PoRX: a reputation incentive scheme for blockchain consensus of IIoT," *Future Generation Computer Systems*, vol. 102, no. 1, pp. 140–151, 2020.

[5] F. Zeng, Q. Chen, L. Meng, and J. Wu, "Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3247–3257, 2021.

[6] F. Zeng, Y. Chen, L. Yao, and J. Wu, "A novel reputation incentive mechanism and game theory analysis for service caching in software-defined vehicle edge computing," *Peer-to-Peer Networking and Applications*, vol. 14, no. 2, pp. 467–481, 2021.

[7] 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.

[8] 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.

[9] I. Abraham, D. Malkhi, and K. Nayak, "Sync HotStuff: simple and practical synchronous state machine replication," in *Proceedings of the 41th Symposium on Security and Privacy*, Piscataway, May 2020.

[10] S. Rahul, W. Mohammad, and G. Prosanta, "A blockchain based secure communication framework for community interaction," *Journal of In-formation Security and Applications*, vol. 58, no. 3, pp. 102790–102803, 2021.

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

[12] 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.

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

[14] J. Dean and S. Ghemawat, "MapReduce," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[15] 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.

[16] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2022, https://bitcoin.org/bitcoin.pdf.

[17] NXT, "Nxt Whitepaper," 2022, https://nxtwiki.org/wiki/Whitepaper:Nxt.

[18] BitShares, "Delegated proof of stake," 2022, http://docs.bitshares.org/bitshares/dpo8.html.

[19] E. WhitePaper, "A next-generation smart contract and decentralized application platform," 2022, https://github.com/ethereum/wiki/wiki/WhitePaper.

[20] S. Singh, "Hyperledger Fabric WhilePaper," 2022, https://github.com/hyperledger/fabric.

[21] L. Lamport, "The part-time parliament," *ACM Transactions on Computer Systems*, vol. 16, no. 2, pp. 33–169, 1998.

[22] L. Lamport, "Paxos made simple," *ACM SIGACT News*, vol. 32, no. 1, pp. 18–25, 2001.

[23] L. Lamport and M. Massa, "Cheap Paxos," in *Proceedings of the 2004 International Conference on Dependable Systems & Networks*, Florence, Italy, June 2004.

[24] N. A. Lin, Z. H. Chen, and G. K. Liu, "Mechanism for proof-of-reputation consensus for blockchain validator nodes," *Journal of Xidian UniverSity*, vol. 47, no. 5, pp. 61–66, 2020.

[25] Q. Li, T. Zhou, L. Lü, and D. Chen, "Identifying influential spreaders by weighted LeaderRank," *Physica A: Statistical Mechanics and Its Applications*, vol. 404, no. 2, pp. 47–55, 2014.

[26] 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.

[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[28] 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.

[29] 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.

[30] 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.

[31] W. F. Silvano and R. Marcelino, "Iota Tangle: a cryptocurrency to communicate Internet-of-Things data," *Future*

Generation Computer Systems, vol. 112, no. 12, pp. 307–319, 2020.

[32] Hedera, "The swirlds hashgraph consensus algorithm: fair, fast, byzantine fault tolerance," 2022, https://docs.hedera.com/guides/core-concepts/hashgraph-consensus-algorithms.

[33] 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.

[34] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[35] 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.

[36] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, New York, USA, October 1999.