Network Resource Allocation Method based on Blockchain and Federated Learning in IoT

Hui Zhi and Yaning Wang

Abstract**——Virtual network embedding (VNE) is an effective approach to solve the resource allocation problem in IoT networks. But most existing VNE methods are centralized methods, they not only impose an excessive burden on the central server but also result in significant communication overhead. Therefore, this paper proposes a distributed resource allocation method based on federated learning (DRAM-FL) to alleviate the computing and communication overhead, and improve network resource utilization. When utilizing DRAM-FL, it is essential to address the security challenges arising from the unreliable nature of IoT devices. So, we introduce blockchain into DRAM-FL, and propose a distributed resource allocation method based on blockchain and federated learning (DRAM-BFL). In DRAM-BFL, a dual-chain structure is designed to facilitate reliable information exchange among nodes, a node reliability assessment method and EPBFT-NRA consensus algorithm are proposed to improve the security of VNE. Simulation results demonstrate that, compared with other methods, DRAM-BFL can increase the VN acceptance rate and long-term average revenue-to-expenditure ratio while improving system security. In addition, DRAM-BFL exhibits good scalability, and has superior throughput and delay performance in IoT with malicious nodes.**

Index Terms**—Blockchain, federated learning, Internet of things, resource allocation, virtual network embedding.**

I. INTRODUCTION

IN the Internet of things (IoT), terminal devices can form networks through self-organization, thus achieving interconnectivity and communication. Today, IoT can be applied in many fields, such as healthcare, retail, industry, transportation, utility, and communication [1]. By 2030, the number of IoT devices in the world will reach 29.4 billion [2], which will be large scale IoTs. With the increase of the number of devices, the application data generated will also increase exponentially [3]. However, the larger scale the IoT, the greater pressure the data communication and resource requirement at terminals [4]. Unfortunately, terminal devices, such as smart sensors, smartphones, and other intelligent electronic devices [5]–[7], often have limited network resources, which poses a huge challenge to IoT's network resource allocation. Moreover, most existing network resource allocation methods are fixed and insufficient to support diverse application requirements. Therefore, how to improve the resource utilization of terminals while meeting diverse application requirements is a key problem in IoT's network resource allocation. Virtual network embedding (VNE) is an effective way to solve this problem [13]. Under the constraints of resource and QoS requirements, VNE can abstract different IoT application requirements into virtual networks (VNs), and then map them to the physical network (PN) according to network resources. This approach can eliminate the resource allocation limitations of the underlying physical network, solving the problems of limited terminal resources and diverse application requirements [8]. So, in this paper, we also use VNE to realize the network resource allocation for diverse IoT application requirements.

Existing VNE resource allocation methods are all centralized methods. That's to say, network resource information and IoT application requirements are uploaded to the central server, and then the central server makes resource allocation decisions, such as heuristics algorithm [9], ant colony system (ACS) [10], Q-learning [11], reinforcement learning (RL) [12], deep reinforcement learning (DRL) [13], Graph convolutional networks (GCN) [14], etc. These centralized methods [9]–[14] are suitable for small networks with a small number of terminals and simple topology. For large-scale IoT networks, these centralized methods [9]–[14] will lead to high communication and computing costs for the system. That's to say, the large-scale IoT VNE problem needs a distributed approach to solve. Distributed federated learning (FL) can be used to solve this problem. So, in this paper, we propose a distributed resource allocation method based on federated learning (DRAM-FL) to reduce the computing and communication cost, simplify the network complexity, and improve the network resource utilization.

Since IoT terminals are mostly low-cost devices with poor reliability, it is essential to consider the reliability of IoT terminals when using DRAM-FL, and avoid information security issues such as data tampering and malicious node attacks. In this regard, blockchain can establish reliable information interaction between untrusted nodes, and solve the information security problem in distributed system [15]. Therefore, we introduce blockchain into DRAM-FL and propose a distributed resource allocation method based on blockchain and federated learning (DRAM-BFL) to improve information security. As far as we know, it is the first attempt to combine blockchain and FL with VNE to solve the resource allocation problem in IoT.

Manuscript received July 21, 2023; approved for publication November 11, 2023; approved for publication by Choi, Jin-Ghoo Division 3 Editor, Febuary 18, 2024.

This research is supported by the National Natural Science Foundation of China (62001001), Key Research and Development Plan in High tech Field of Anhui Province (202304a05020011), Natural Science Foundation of the Anhui Higher Education Institutions of China(2022AH050109).

The authors are with Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, Anhui University, emails: {zhihui_0902 wyn_1068}@163.com.

H. Zhi is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2024.000007

The main contributions of this paper are as follows:

1) A DRAM-BFL is proposed. Firstly, we transform the resource allocation problem into a VNE problem, and proposes a DRAM-FL to solve it. This method enables local networks to collect information, learn the dynamics of the network and make optimal VNE decisions through their respective local training nodes. It can improve VN acceptance rate and long-term average revenue-to-expenditure ratio, reduce the complexity of IoT network topology, and reduce the computational and communication costs.
2) We introduce blockchain into DRAM-FL and propose DRAM-BFL. A blockchain structure with dual chains is designed, which contains topology chain and model chain. The topology chain is used for secure sharing of network topology information, and the model chain is used for secure sharing of model information in FL. The dual chains can realize protection and access control of on-chain data, reduce the hidden risk of data leakage, and realize decentralized FL.
3) In DRAM-BFL, a node reliability assessment method is proposed and the reliability of physical nodes is measured through evaluating their performance. During the node mapping stage, VNE decisions can be made based on the results of node reliability assessment, this can effectively avoid attacks from malicious nodes.
4) In order to further improve the security of DRAM-BFL, we design the enhanced practical Byzantine fault tolerant consensus algorithm based on node reliability assessment (EPBFT-NRA). And the consensus nodes are dynamically updated to ensure the fairness of nodes and improve the security of consensus process.

II. Related Work

A. Virtual Network Embedding (VNE) Algorithms

The VNE problem has been proven to be NP-hard [16], so most of the traditional VNE methods are heuristics algorithms [17]–[20] that rely on manual rules to make embedding decisions. Initially, these heuristics algorithms were used in single-domain scenarios. However, with the diverse application of the IoT, such as medical services, retail, industry, transportation, etc., there may be multiple network service requirements for the same physical network. Each requirement has different transmission performance requirements. So, researchers began to study multi-domain heuristic algorithms [21]. These multi-domain heuristic algorithms are centralized methods and make decisions by a third party [22]. Furthermore, heuristic algorithms need follow fixed rules to get embedding decisions. They cannot adaptive optimize according to the changes of the physical network, they can easily fall into local optimum and often get suboptimal solutions.

B. Machine Learning based Virtual Network Embedding Algorithms

Large-scale IoT network and fragmented resources bring great challenges to the traditional VNE algorithms. Many researchers turn their attention to the machine learning (ML) methods. Unlike heuristic algorithms, ML can effectively avoid falling into local optimization. For example, [23] models the problem of resource allocation under DDOS as Bayesian Q learning game, and a greedy Q learning algorithm is proposed to solve it. [24] combines deep Q-network and Monte Carlo method to achieve fast convergence of VNE decision. [25] combines Monte Carlo tree searching and RL neural network to solve the VNE problem. [26] uses RL to solve VNE problem and abstracts node mapping into the Markov decision-making process (MDP), which requires large computing costs and has low time efficiency. In [27], RL is used in VNE, where policy gradients are used to train policy networks. In [28], RL and policy networks are used in the node mapping phase, and bandwidth resource sequencing is used in link mapping phase. With the objective of maximizing the number of embedded VNs, deep reinforcement learning (DRL) is applied to VNE in [29], which combines deep learning (DL) and RL to enhance the success rate. In [30], a VNE algorithm based on RL security perception is proposed. In [31], a multi-stage VNE prediction model based on RL is proposed for cloud data centers to determine suitable physical resources and optimize the resource utilization.

In summary, all these heuristic algorithms [17]–[22] and ML algorithms [23]–[31] are all centralized algorithms. The local training data needs to be uploaded to a central server to make resource allocation decisions, resulting in large computing costs and communication costs in large-scale IoT. Therefore, in this paper, we adopt distributed FL, which is different from traditional centralized training methods. FL can reduce the network topology complexity in IoT, greatly minimize communication and computing costs while improving the scalability of VNE.

C. Security-aware Virtual Network Embedding Algorithms

Considering the complex network environment of IoT, the security of the VNE process is also crucial. Distributed FL algorithms face security challenges such as malicious node attacks and data tampering. Some approaches are proposed to address the security of VNE. For example, [31] establishes security levels for VNs, but does not consider the security of physical nodes and the underlying physical network. References [29] and [32] incorporate constraints on reputation or security level of physical nodes into the node importance ranking process. All the methods in [29], [31], [32] are designed for centralized VNE algorithms and rely on manually formulated node reliability assessment, which may not accurately reflect the actual network node situation and cannot ensure the security of VNE decisions.

As a result, some researchers begin to study improvements or new solutions to solve the security problems of today's IoT. Blockchain technology is a relatively new solution. Some researches integrate block chain into the frame of FL in IoT scenarios, build trust between devices, realize privacy preserving and model sharing in FL, effectively prevent the FL single point failure and malicious attacks. For example, [33] jointly considers channel assignment, block size adjustment

TABLE I
SYMBOLS AND VARIABLES.

Noatation	Description
$CPU(n_l^V)$	The computing resource demand of n_l^V
$BW(l_{lk}^V)$	The bandwidth resource demand of l_{lk}^V
Vac	VN acceptance rate
$R(G^V,t)$	The long-term revenue of embedding G^V
$R(G^V)$	The long-term average revenue
$C(G^V,t)$	The long-term expenditure of embedding G^V
$C(G^V)$	The long-term average expenditure
RC	The long-term average revenue-to-expenditure ratio
ω_t^i	The global model parameter of *t*th round aggregation
Δ_t	The local model parameter of *t*th round training
S_t	The set of local training nodes in global update
k	The processing speed of consensus nodes
T_{arrive}	The delay for VN to reach the processing stage
$T_{con}(t)$	The delay of consensus process
$T_{up}(t)$	The delay of the node update process
$T_{bc}(t)$	The average delay of the blockchain system
$\mathbb{S}^{N_i^S}$	The reliability assessment value of node N_i^S

and block producer selection to reduce the energy consumption and improve the security of FL. [34] proposes an IoT resource management framework combining blockchain and FL, which uses support vector machine (SVM) classifier to improve the accuracy of detecting malicious nodes. [35] addresses the privacy risks of gradient inversion attacks by building a self-aggregating privacy-protecting FL model at the top of the blockchain. [36] proposes a Byzantine robust federated learning framework to realize privacy preserving. However, all the studies in [33]–[36] focus on the effective and secure resource management of FL, uses blockchain to realize privacy preserving, model sharing or preventing malicious attacks in FL. All the studies in [33]–[36] did not consider how to effectively utilize IoT's network resources to meet different application requirements, not only the FL resource requirement. VNE model is an effective means to realize IoT resource management to meet different application requirements. So, in this paper, we first propose a VNE model in IoT, and then use FL and blockchain to improve the efficiency and safety of VNE, the proposed VNE model can meet more application needs, including FL.

At present, there is a lack of researches on the security of distributed VNE algorithms. Therefore, in this paper, we intend to incorporate blockchain into distributed VNE resource allocation and propose a dual-chain structure (topology chain and model chain) to improve the security of the VNE process. Simulation results also show that the proposed DRAM-BFL can achieve more secure and efficient IoT resource allocation when compared with the traditional VNE method.

In addition, the symbols and variables mentioned in the paper are given in Table 1.

III. System Model for Network Resource Allocation

A. VNE Model and VNE Problem

For the convenience of analysis, we divide IoT nodes into two types: Base stations (BSs) and general devices. BSs are the nodes that usually have high computing power, large storage space, and fixed location. General devices are terminal devices that usually have low computing power, small storage space, and mobile location. When a virtual network (VN) application requirement arrives, IoT needs to adopt appropriate resource allocation method based on the current network status to meet the application requirement of VN. Once the VN requirement is successfully embedded (mapped), the resources within IoT will be consumed until the end of VN application. In this context, we define the underlying physical network as PNs, refer to application requirements as VNs. As a result, we can transform the network resource allocation problem into a multi-domain VNE problem.

1) Physical network: The physical network is represented as an undirected weighted graph: $G^S = (Node^S, Link^S, Path^S, Func^S)$. There are Q physical nodes in the physical network, and these physical nodes are recorded as $n_1^S, \cdots, n_Q^S$. $Node^S$ represents the set of all physical nodes in the physical network, $Node^S = \{n_1^S, \cdots, n_Q^S\}$. The physical link from n_i^S to n_j^S is denoted as l_{ij}^S, the set of all physical links is $Link^S = \{l_{ij}^S, i,j \in \{1,2,\cdots,Q\}, i \neq j\}$. $Path^S$ represents the set of all physical paths in this physical network. A physical path contains one or more physical links, the latter situation belongs to path splitting. In the VNE, the resource attributes of the physical node n_i^S include CPU resources $CPU(n_i^S)$ and node reliability assessment $NES(n_i^S)$. The resource attribute of the physical link l_{ij}^S is the available bandwidth resource $Bw(l_{ij}^S)$. $Func^S$ epresents the set of functional attributes for the physical network, which includes node location $Loc(n_i^S)$ of all physical nodes.

2) Virtual network: Each application requirement is defined as a VN. Then VNE is used to allocate network resources. A virtual network is described as an undirected weighted graph: $G^V = (Node^V, Link^V, Func^V)$. There are R virtual nodes in the virtual network denoted as n_1^V,...,n_R^V. $Node^V$ is the set of all virtual nodes, that is $Node^V = \{n_1^V, \cdots, n_R^V\}$. The virtual link from n_l^V to n_k^V is denoted as l_{lk}^V. $Link^V$ represents the set of all virtual links in the virtual network, that is $Link^V = \{l_{lk}^V, l,k \in \{1,2,\cdots,R\}, l \neq k\}$. In the VNE, the resource attribute of a virtual node n_l^V is the node CPU resource $CPU(n_l^V)$. The resource attribute of the virtual link l_{lk}^V is available bandwidth $Bw(l_{lk}^V)$. $Func^V$ represents the set of functional attributes for the virtual network, which includes node location $Loc(n_l^V)$ and maximum location deviation $MaxLocDev(n_l^V)$ of all virtual nodes.

3) Multi-domain VNE model: Fig. 1 shows the multi-domain VNE model in IoT, the process of VNE can be represented as $G^V \rightarrow G^S$. As shown in Fig. 1, a virtual node can only be mapped to one physical node, different virtual nodes of the same VN must be mapped to different physical nodes, virtual nodes of different VNs can be mapped to the

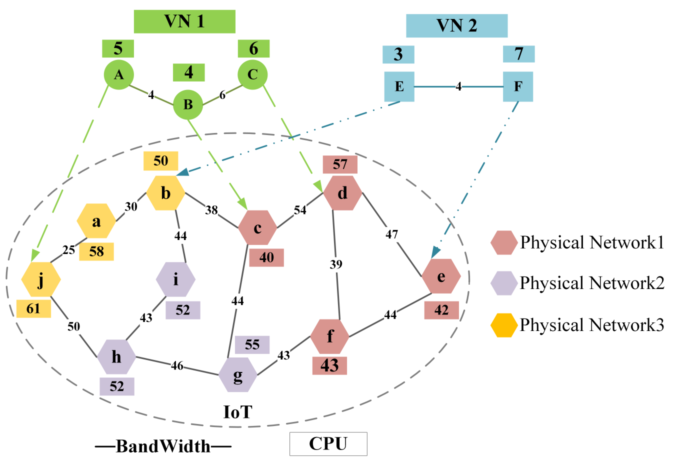

Fig. 1. Multi-domain VNE model in IoT.

same physical node. Furthermore, a virtual link can be mapped into a physical path, which contains one or more physical links.

B. VNE Evaluation Index

1) VN acceptance rate: VN acceptance rate, also known as VN success rate, is one of the important indexes to evaluate the performance of VNE algorithms. VN acceptance rate means the ratio of the number of VNs successful embedding to the total number of VNs in the time T.

$$Vac = \frac{Accepted_{VN}^{T}}{Total_{VN}^{T}}, \tag{1}$$

where $Accepted_{VN}^{T}$ indicates the number of VNs successfully accepted by the physical network from time 0 to T. $Total_{VN}^{T}$ indicates the total number of VNs received by the physical network from time 0 to T. The VN acceptance rate reflects resource utilization, and the higher the VN acceptance rate, the more the resources utilization of the physical network.

2) Long-term average revenue: The revenue of embedding G^V means the revenue obtained by the underlying physical network through renting out resources after G^V is successfully embedded. It is calculated as follows.

$$R(G^V,t) = \sum_{n_l^V \in Node^V} CPU(n_l^V) + \sum_{l_{lk}^V \in Link^V} BW(l_{lk}^V), \tag{2}$$

where $CPU(n_l^V)$ represents the computing resource demand of the virtual node n_l^V, $BW(l_{lk}^V)$ represents the bandwidth resource demand of the virtual link l_{lk}^V. Their revenue weights are determined by the network service provider, and are uniformly set to 1 in this paper to simplify analysis.

That's to say, the revenue of underlying physical network for embedding G^V is determined by the sum of the CPU resources consumed and the bandwidth resources consumed of embedding G^V during the given duration time t.

Then, the long-term average revenue can be defined as:

$$R(G^V) = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} R(G^V,t)}{T}. \tag{3}$$

$R(G^V)$ measures the average value of the revenue obtained by the underlying physical network in time T, which represents the ability of the network service provider to obtain revenue.

3) Long-term average revenue-to-expenditure ratio: The long-term revenue-to-expenditure ratio reflects the utilization of underlying resources. If the ratio is high, it indicates that the utilization rate of the underlying resources is high and the number of VNs successfully accepted is large.

The long-term expenditure is defined as

$$\begin{aligned} C(G^V,t) = & \sum_{n_l^V \in Node^V} CPU(n_l^V) \\ & + \sum_{l_{lk}^V \in Link^V, p_{ij}^S \in Path^S} Num(p_{ij}^S) BW(l_{lk}^V), \end{aligned} \tag{4}$$

where $CPU(n_l^V)$ represents the amount of physical node computing resource occupied by the virtual node n_l^V, $BW(l_{lk}^V)$ represents the amount of physical path bandwidth resource occupied by the virtual link l_{lk}^V.

In the link embedding in (4), a virtual link l_{lk}^V is mapped into a physical path p_{ij}^S, which consists of one or more physical links. $Num(p_{ij}^S)$ denotes the number of physical links on the physical path p_{ij}^S.

The long-term average expenditure is calculated as

$$C(G^V) = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} C(G^V,t)}{T}. \tag{5}$$

Therefore, the long-term average revenue-to-expenditure ratio of VNE is defined as

$$RC = \frac{R(G^V)}{C(G^V)}. \tag{6}$$

RC measures the ratio of the average revenue and the average resource consumed expenditure of the physical network for embedding G^V during time T. RC is an index used to measure the resource utilization efficiency of the underlying physical network. The higher the RC, the higher the resource utilization efficiency of the network.

IV. DRAM-FL

A. Network Resource Allocation Problem

The goal of the network resource allocation is to find optimal mapping decision $F(n_l^V \to n_i^S, L_{lk}^V \to P_{ij}^S)$ (i.e., embedding decision) to maximize the sum of VN acceptance rate and revenue-to-expenditure ratio under the constraints of resources, function and reliability. Therefore, the network

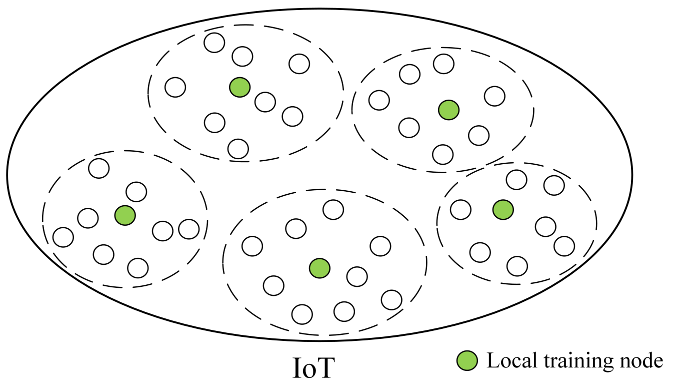

Fig. 2. System model of DRAM-FL.

Algorithm 1: Workflow of DRAM-FL

```
Input: Local training nodes $I$, local mini batches $B$,
  epochs $E$,
learning rate $\eta$, $T$, $i = 1, 2, \cdots, I$
Procedure: Local training
1: for $l = 1 \to E$ do
2:    for $b \in B$ do
3:       $\omega_t^i = \omega_{t-1} - \eta \nabla F_i(\omega_{t-1})$
4:       $\Delta_t^i = \omega_t^i - \omega_{t-1}$
5:    end for
6:    Local model parameters $\Delta_t^i$ are uploaded to the
  central sever
7: end for
Procedure: Checking local model
8: evaluate $\omega_t^i$ (Obtain partial local network
  information through the local dataset $\in D_i$)
9: return $S_t$
Procedure: Checking inner product
10: $\nabla F(\omega_t^i) = {}^{-\Delta_t^i}\!/_{\eta}$
11: $\nabla F(\omega_t) = \frac{1}{|S_t|} \sum_{i \in S_t} \nabla F_i(\omega_t)$
12: for $j = 1, \cdots, \backslash|\{\{S\_t\}\}\backslash|$ do
13:    if $j = 1, \cdots, |S_t|$
14:       $\bar{S}_t \leftarrow S_t.pop(S_t[j])$
15: end for
16: return $\bar{S}_t$
Procedure: Global Model Aggregation
17: Initializes $\omega_0$;
18: for $t = 1, 2, \cdots, T$ do
19:    $\Delta_t = \frac{1}{|\bar{S}_t|} \sum_{i \in \bar{S}_t} \Delta_t^i$
20:    $\omega_t = \omega_{t-1} + \Delta_t$
21: end for
22: return $\omega_t$
```

resource allocation problem can be described as the VNE optimization problem $P1$.

$$
\begin{aligned}
P1 : & \mathop{\arg}_{F(n_l^V \to n_i^S, L_{lk}^V \to P_{ij}^S)} \max(Vac + RC) \\
s.t. C1 : & CPU(n_i^S) \geq CPU(n_l^V) \\
C2 : & Bw(l_{ij}^S) \geq Bw(l_{lk}^V), l_{ij}^S \in p_{ij}^S \\
C3 : & Distance(Loc(n_i^S), Loc(n_l^V)) \leq MaxLocDev(n_l^V) \\
C4 : & \varphi(n_l^V \to n_i^S) = 1, \\
& \forall n_l^V \in Node^V, n_i^S \in Node^S, n_i^S \notin FN \\
C5 : & \varphi(l_{lk}^V \to l_{ij}^S) \geq 1, \\
& \forall l_{lk}^V \in Link^V, l_{ij}^S \in Link^S
\end{aligned}
\tag{7}
$$

Specifically, VNs cannot be indefinitely mapped to the physical network due to the limited resources. Conditions $C1$ and $C2$ are resource constraints, which ensure that the available resources of the underlying network nodes and links meet the resource requirements of the virtual nodes and virtual links. $C3$ means the range of VNE cannot exceed the maximum position deviation, it is a functional constraint to avoid long-distance transmission. $C4$ means that a virtual node of the same VN can only be mapped in one physical node, where the set of malicious physical nodes FN is excluded. $C5$ indicates that a virtual link can be mapped into one or more physical links, the latter case refers to path splitting.

B. System Model of DRAM-FL

As shown in Fig. 2, we integrate FL into VNE problem, and propose the DRAM-FL to solve the VNE problem. We divide IoT into multiple local networks based to geographical location, select a capable node as the local training node in each local network for FL. The local training node collect real-time topology information, such as physical location, CPU resources, link strength, node connectivity, and node reliability assessment, etc.

C. Workflow of DRAM-FL

Assume IoT network is divided into I local networks, which are denoted as $G_1^S, \cdots, G_i^S, \cdots, G_I^S$ respectively, and the dataset size of local network G_i^S is denoted as D_i. When the local training node receives the VN request, it downloads the pre-trained model and starts online learning. The local training node takes the collected local network topology information as the model input and makes VNE decisions as the model output. Local training nodes perform mixed pattern training, that is to say, each local training node will wait for the completion of other training nodes. However, this algorithm has a timeout setting, if it times out, it discards incomplete training node.

For the local training model of VNE, we refer to the neural network model in [37]. Its input is the feature matrix composed of CPU, bandwidth, and node reliability assessment value. The neural network model structure can be divided into four layers:

1) Input layer: it is used to input the feature matrix that contains local network topology information.
2) Convolution layer: it evaluates the resource characteristics of each feature vector in the feature matrix by convolution operation on the extracted feature matrix, and obtains the residual resource vector of each node.
3) Probability layer: the softmax function is used to normalize the gradient logarithm of the discrete probability distribution of the residual resource vector to obtain the corresponding embedding probability of each physical node.

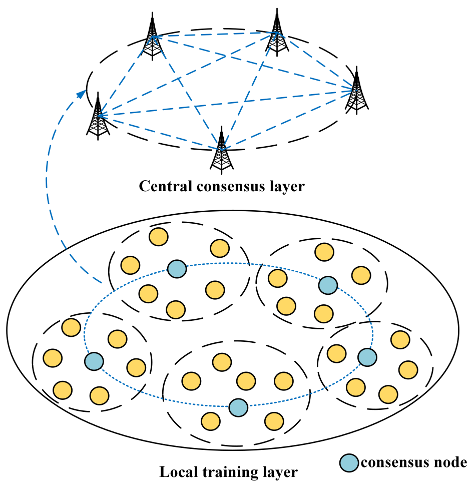

Fig. 3. System model of DRAM-BFL.

4) Filter layer: it filters out physical nodes with insufficient resources and low reliability, and outputs a set of possible embedding physical nodes for each virtual node and their corresponding embedding probabilities.

The workflow of DRAM-FL is described in **Algorithm 1**.

In DRAM-FL, FedAvg is used for global model aggregation, its objective function is

$$\min \frac{D_i}{D} \sum\nolimits_{i=1}^{I} \Delta_t^i. \tag{8}$$

But FedAvg here uses a different local training node selection method when compared to traditional methods. According to [38], we choose the node probability selection scheme, so that the central server can dynamically adjust the probability of each node being selected in each round of aggregation according to the most aggregation results, and accelerate the convergence of the model. We need to check the inner product of local gradient and global gradient for all local training nodes to ensure model convergence. If $\langle \nabla F(\omega_t), \nabla F_i(\omega_t) \rangle < 0$, the convergence speed of the global model will slow down. Therefore, in this paper, we exclude local training nodes that are not conducive to global update, and the remaining set of local training nodes participating in global update is S_t.

V. DRAM-BFL

Since IoT terminals have poor reliability, it is essential to consider the reliability of IoT terminals when using DRAM-FL. In order to promote the security sharing of network information between nodes, and prevent data tampering and malicious node attacking, we introduce blockchain into DRAM-FL, and propose DRAM-BFL. Additionally, we design a node reliability assessment method to mitigate the risk of malicious node attacking.

A. System Model and Problem Description of DRAM-BFL

In Fig. 3, DRAM-BFL contains a local training layer and a central consensus layer. The local training layer is consistent with the model in Fig. 2. In the central consensus layer, the consensus nodes are selected through the node reliability assessment method (Section V.D). We choose the top n BSs with the highest node reliability as the central consensus layer. This layer implements a distributed blockchain, where consensus nodes can trade and securely share model information and network topology information with other nodes. The central consensus layer verifies the training results of each local training node by collecting the topology information and local model parameters.

Through introducing blockchain, problem is changed into problem $P2$ with adding security constraint $C6$. Detailed analysis of (17) is given in Section V.C.

$$\begin{aligned}
P2:\ & \underset{F(n_l^V \to n_i^S,\, L_{lk}^V \to P_{ij}^S)}{\arg\max} (Vac + RC) \\
s.t.\, C1: & \ CPU(n_i^S) \geq CPU(n_l^V) \\
C2: & \ Bw(l_{ij}^S) \geq Bw(l_{lk}^V),\ l_{ij}^S \in p_{ij}^S \\
C3: & \ Distance(Loc(n_i^S), Loc(n_l^V)) \leq MaxLocDev(n_l^V) \\
C4: & \ \varphi(n_l^V \to n_i^S) = 1, \\
& \ \forall n_l^V \in Node^V, n_i^S \in Node^S, n_i^S \notin FN \\
C5: & \ \varphi(l_{lk}^V \to l_{ij}^S) \geq 1, \\
& \ \forall l_{lk}^V \in Link^V, l_{ij}^S \in Link^S \\
C6: & \ (17)
\end{aligned} \tag{9}$$

B. Workflow of DRAM-BFL

The workflow of DRAM-BFL is shown in Fig. 4. Based on the node assessment method described in Section V.D, we select a base station with high reliability as the local training node to ensure the security of FL. we adopt the Parameter Server architecture and the FL with asynchronous pattern. The detailed descriptions of Fig. 4 are as follows:

<1> Explore environment. Each local training node explores the environment of its own local network and obtains its topology information separately.

<2> Upload information. The topology information obtained by the local training node is uploaded to the central consensus layer.

<3> Chain up. In the central consensus layer, the topology information is packaged into blocks and uploaded to the topology chain.

<4> Local training. When the local training node receives the VN requirements, it downloads the pre-trained model and starts online training, and feeds the training results (local model) back to the central consensus layer.

<5> Verify. The local training node upload trained local model to the central consensus layer. After receiving the local model, the central consensus layer first obtains the topology information from the topology chain to verify the local model, and then adds the verification results to the topology chain. The validation results are also used as part of the node reliability assessment.

<6> Model aggregation. The global model is obtained through the aggregation of local training models.

<7> Chain up. The global model is packaged into blocks and uploaded to the model chain.

<8> Model update. Each local training node obtains the latest global model from the model chain, then it starts a new round of training for its own local model.

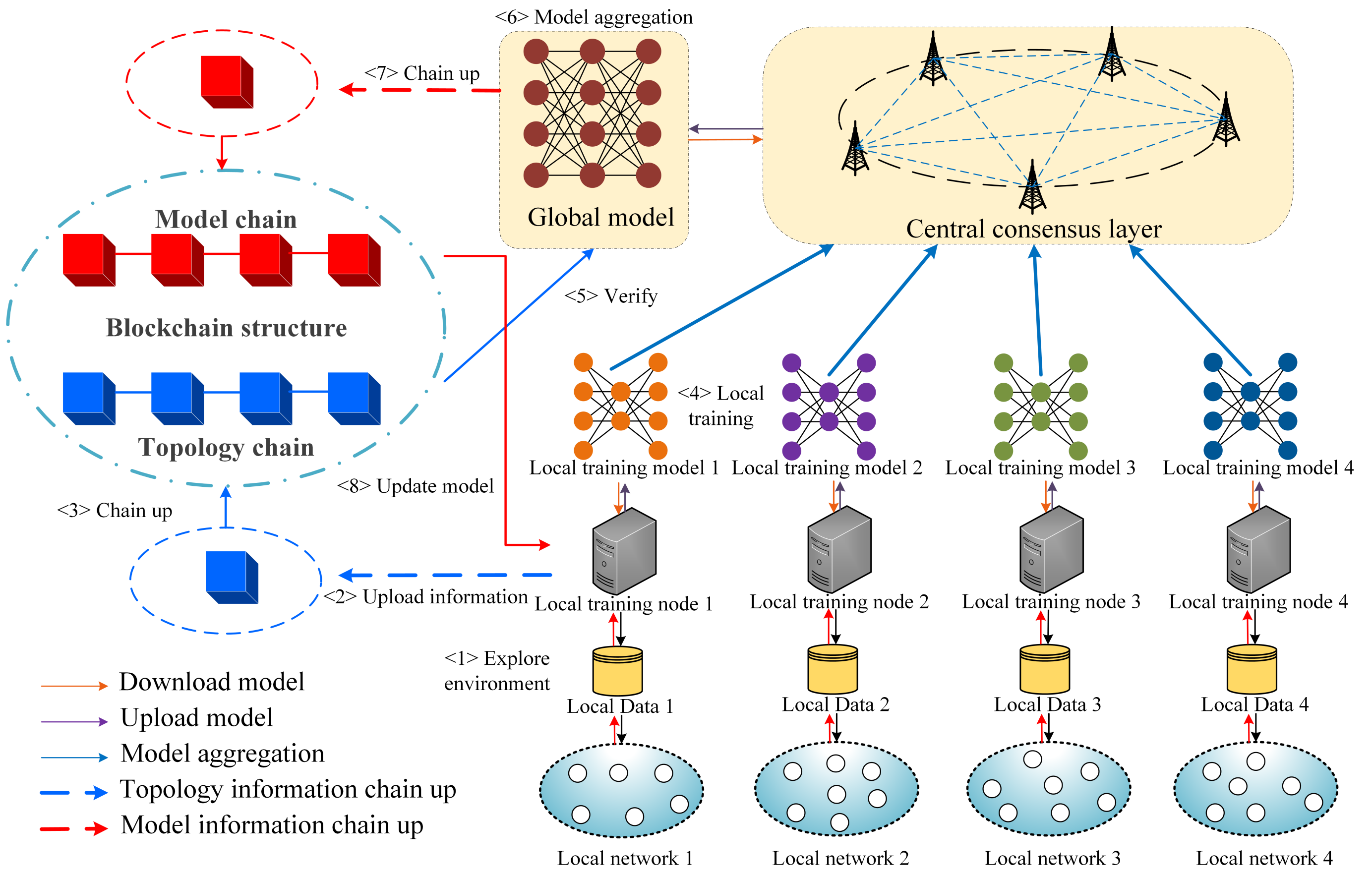

Fig. 4. The workflow of DRAM-BFL.

C. Dual Chain Structure

We introduce blockchain into VNE, design a dual chain structure that contains topology chain and the model chain.

1) Topology chain: The topology chain is a public chain that used to store the local network topology information. Local training nodes are served as light nodes in topology chain, they collect network topology information from neighboring nodes, save it, are then upload it to the central consensus layer. The network topology information includes the physical location of nodes, CPU resources, link strength, node connectivity, and node reliability. After the consensus process, the central consensus layer packages this information into a block, adds it to the topology chain. The topology chain has a high degree of security and transparency, all information including node reliability is publicly recorded and all nodes can access this information. In the VNE process, network topology information stored in the topology chain can effectively improve security and play a key role in subsequent model verification.
2) Model chain: After completing local model training, the local training node uploads the local training model information to the central consensus layer. The central consensus node aggregates these models, packages the latest global aggregated model information into a block, and adds the new block to the model chain through the consensus process. So, the model chain has high privacy and controllability, and can be used to save the sensitive data of model parameters. Then, the local training node can directly obtain the latest global training model information from the model chain, this method can reduce direct information interaction between the local training layer and the central consensus layer and improve the security of the model update process. Additionally, before model aggregation, the central consensus layer verifies local training model, and the validation result is one of the main parameters of reliability evaluating for local training nodes.

The dual-chain structure is a multichain blockchain, two chains store different data content, and the two chains cooperate with each other to complete VNE tasks. Topology chain is a public chain used to store and verify open information, ensuring security and consistency across of the network. Model chain is an alliance chain used to deal with sensitive data and private transactions. Such structure can strike a balance between protecting privacy and maintaining transparency while meeting the needs of different organizations interoperate operations, and providing a secure and reliable way for multiple organizations to share data. What's more, the dual-chain structure facilitates the queries of VNE processes. The workflow of DRAM-BFL can be written into a smart contract that can act automatically.

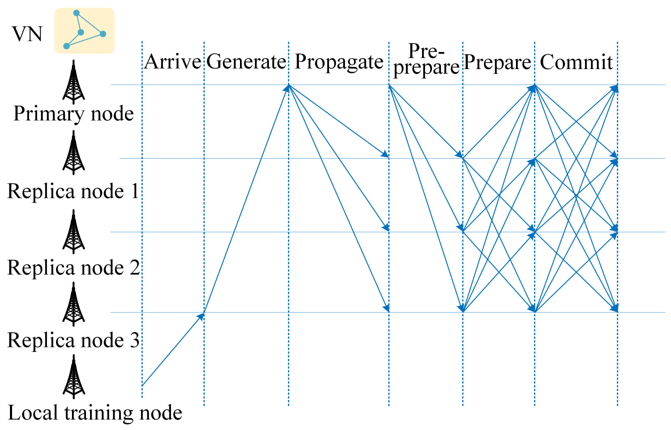

Fig. 5. Consensus process for EPBFT-NRA.

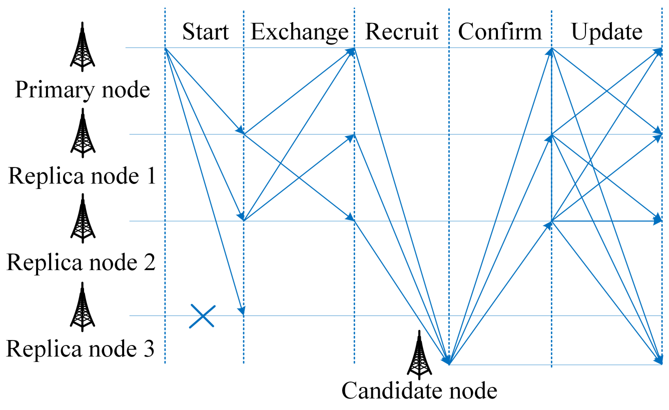

Fig. 6. Consensus node updating process in EPBFT-NRA.

1) EPBFT-NRA: Considering the consensus security, we design the enhanced practical Byzantine fault tolerant (PBFT) consensus based on node reliability assessment. The EBPFT-NRA integrates reliability assessment and consensus node dynamic update, and PBFT. Initially, the node with the highest reliability assessment value is selected as primary node, the top n central consensus nodes with the highest node reliability value are selected as consensus node (replica node). After every M round of consensus process, EPBFT-NRA performs a consensus node updating process to ensure the fairness of nodes and the security of consensus process.

EPBFT-NRA is based on PBFT protocol. If we assume that the total number of consensus nodes in the system is $\mathbb{N}$,and there are f malicious nodes. For PBFT, only when $\mathbb{N} \geq 3f+1$, the security of the system can be guaranteed. The blockchain structure stores the information of the latest VNE process, and begins a new consensus process after completing a new VNE.

The Consensus process of EPBFT-NRA is shown in Fig. 5. The detailed descriptions are as follows.

1. Arrive: When a VNE arrives, the local training node sends its information to the nearest consensus node i.

2. Generate: The node with the highest reliability value is selected as primary node. After receiving the information from the local training node, the consensus node i packages this information and forwards it to the primary node.

3. Propagate: After the primary node receives messages from other consensus nodes, it packages all the messages and generates a block, then sends this new block to other replica nodes. After the primary node verifies the MAC, the three-phase broadcast protocol of PBFT starts, which are pre-prepare, prepare, and commit.

4. Pre-prepare: The primary node sends the pre-prepare message to all replica nodes. Each replica node accepts the pre-prepare message and enters the prepare phase.

5. Prepare: When replica nodes enter the prepare stage, each replica node sends a prepare message to all other replica nodes. After receiving the new block information, each replica node validates the information. For a replica, if it receives information from $2f$ different replicas are consistent with the pre-prepare message, that is, with $2f+1$ confirmations, the replica node is prepared. After all replica nodes are prepared, the consensus enters commit phase.

6. Commit: Each replica node broadcasts a commit message to all other replica nodes. If $2f+1$ commit messages (including the message of itself) are correct, the replica node enters the committed state. After all replica nodes enter the committed state, the consensus process is completed, the new block is added to the blockchain.

The consensus node update process is shown in Fig. 6. The detailed descriptions are as follows.

1. Start: After M rounds of consensus process, the primary node will start the dynamic update process and send the start information to other replica nodes. In order to select new n replica nodes and new primary node, a reliability value threshold is calculated according to new node reliability values, and the start information includes this threshold value.

2. Exchange: After replica nodes receiving the start information from primary node, each replica node determines whether its reliability value is greater than the threshold value, if it is greater than the threshold, the replica node sends the update information to other replica nodes. If a replica node receives the information that from other $2f+1$ replica node, it enters the recruitment phase. The replica nodes with reliability values less than the threshold will be removed from the replica nodes.

3. Recruit: The remaining replica nodes send recruitment information to other consensus nodes (non-replica nodes).

4. Confirm recruitment: Each non-replica node determines whether its reliability value is greater than the threshold value, if its reliability value is greater than the threshold, it becomes a candidate node. All candidate nodes will send recruitment confirmation message to all replica nodes.

5. Update: After receiving recruitment confirming messages, each replica node will send node update information to other replica nodes and candidate nodes. If each replica node and each candidate node receive the update information from more than $2f+1$ nodes, these candidate nodes will become the replica nodes. The process of consensus node updating is end.

2) Blockchain system delay: The delay of block generation can be determined by two parts: consensus delay and node update delay. According to [39], only the computational delay of cryptographic operations is considered, including verifying signatures delay, generating MACs delay, and verifying MACs delay, which are C_s, C_m and C_v respectively, the processing speed of consensus nodes is k GHz.

1) Consensus delay: The blockchain consensus process is shown in Fig. 5. We set the delay T_{arrive} for VN to reach the processing stage to a fixed value. Then, the cost of the primary node and the replica node in the consensus process are respectively.

$$C_{_P}^{con} = 2(\mathbb{N}-1)C_m + (4f+1)(C_s+C_v) \tag{10}$$

$$C_{_R}^{con} = 2(\mathbb{N}-1)C_m + (4f+1)C_s + 2(2f+1)C_v \tag{11}$$

The consensus delay is

$$T_{con}(t) = \frac{\max(C_{_P}^{con}, C_{_R}^{con})}{k} + T_{arrive}. \tag{12}$$

2) Node update delay: The consensus node update process is shown in Fig. 6. According to the process, we can calculate the cost of the primary node and the replica node as follows.

$$C_{_P}^{up} = (\mathbb{N}-1)C_m + 2f \times (C_s+C_v) \tag{13}$$

$$C_{R}^{up} = \mathbb{N} \times C_m + 2f \times C_s + 2(f+1)C_v \tag{14}$$

The delay of the node update process is

$$T_{up}(t) = \frac{\max(C_{_P}^{up}, C_{_R}^{up})}{k}. \tag{15}$$

After every M consensus process, the node update process is developed. Additionally, the average delay calculation of the blockchain system is

$$T_{bc}(t) = \frac{MT_{con}(t) + T_{up}(t)}{M}. \tag{16}$$

To ensure the security of VNE in IoT, it is necessary to limit the consensus delay [40] of the blockchain, so

$$\forall t T_{bc}(t) \le \min(T_{bc}(t)), \tag{17}$$

where $T_{bc}(t)$ is the delay limit to ensure system security.

D. Node Reliability Assessment Method

When there are malicious nodes in IoT, the attack behaviors of malicious nodes, such as malicious packet loss, malicious transmission, can cause VNE failure. This results in resource utilization decreasing and IoT tasks failure. To mitigate the impact of malicious nodes on the system performance, we propose a node reliability assessment method. Firstly, we analyze the behaviors of malicious nodes as follows:

1) Malicious embedding: After receiving the embedding decision, the malicious node may abandon the embedded decision with a certain probability.

2) Malicious transmission: After receiving the packet, the malicious node does not send the packet according to the optimal physical path selected by VNE. For example, arbitrarily transmitting packets to other physical paths that are already embedded, or maliciously deviating the optimal link embedding decision and choosing another long-distance physical path.

3) Malicious model uploading: If the malicious node is a local training node, it may maliciously upload local model information with poor performance to the central consensus layer.

These three malicious behaviors are quantified into corresponding security assessment values respectively and integrated into a node reliability assessment algorithm.

1) Node embedding assessment

$$RS^{N_i^S} = \frac{R^{N_i^S}}{S^{N_i^S}}, \tag{18}$$

where $R^{N_i^S}$ is the embedding completed by the node N_i^S per unit of time; $S^{N_i^S}$ is embedding decisions received by the node N_i^S per unit of time.

2) Node transmission assessment

$$TR^{N_i^S} = 1 - \frac{MD^{N_i^S}}{1 + D^{N_i^S}}, \tag{19}$$

where $MD^{N_i^S}$ is the number of times the node N_i^S is detected as malicious transmission per unit time; $D^{N_i^S}$ is the total number of times the node N_i^S is detected for transmission per unit of time, $1 + D^{N_i^S}$ is set to avoid the situation that denominator is equal to 0.

3) Node model uploading assessment

$$TM^{N_i^S} = 1 - \frac{MZ^{N_i^S}}{1 + Z^{N_i^S}}, \tag{20}$$

where $MZ^{N_i^S}$ is the number of times the node N_i^S is detected as malicious uploads per unit time; $Z^{N_i^S}$ is the total number of times the node N_i^S is detected for model uploading per unit of time.

4) The assessment of node reliability

$$\mathbb{S}^{N_i^S} = RS^{N_i^S} \cdot TR^{N_i^S} \cdot TM^{N_i^S}. \tag{21}$$

It can be seen from (21) that the larger $\mathbb{S}^{N_i^S}$, the higher the reliability of the node N_i^S.

VI. SIMULATION RESULTS AND ANALYSIS

We use GT-ITM topology generator [41] to generate the topology of the underlying physical network and the virtual network requirements. The number of nodes in the initial underlying physical network is 100, the connection probability between nodes is 0.5, and the CPU and bandwidth resources of each physical link are randomly generated following a uniformly distributed between 50 and 100. The underlying physical network is divided into 8 local networks according to geographical location. For the virtual network requirements, the requirement arrival follows Poisson distribution, the arrival time is 1000 unit times and the expected is 5. The number of virtual nodes follows a uniform distribution between 2 and 8, and the connection probability between nodes is 0.5. The CPU and link bandwidth resources of each virtual node are randomly generated following a uniformly distributed between 1 and 20. We generate 1,000 VN requirements during 200,000 unit times, and we use these VN requirements as a local dataset. The learning rate η is set to 0.001, the blockchain size is 1 MB, k is 2.4 GHz, C_s is 0.034 M (CPU cycles), C_m is 4 M (CPU cycles), and C_v is 4 M (CPU cycles).

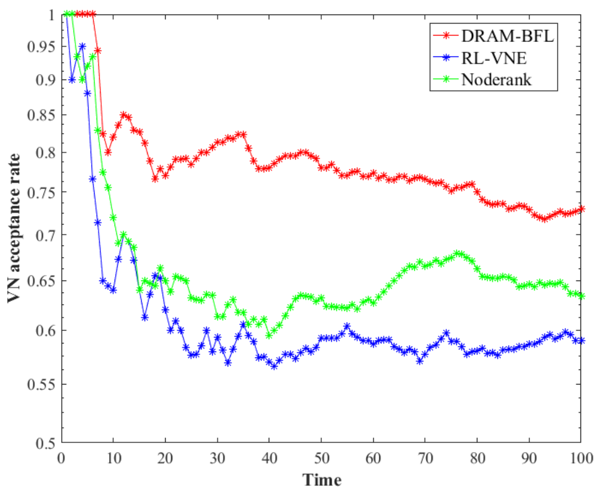

Fig. 7. VN acceptance rate (PN=100).

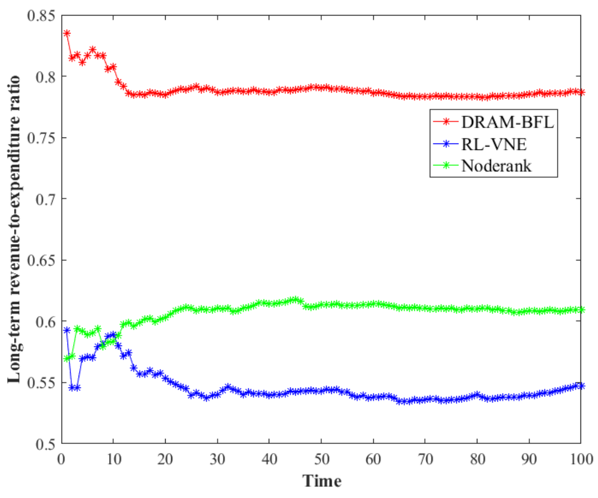

Fig. 8. Long-term revenue-to-expenditure ratio (PN=100).

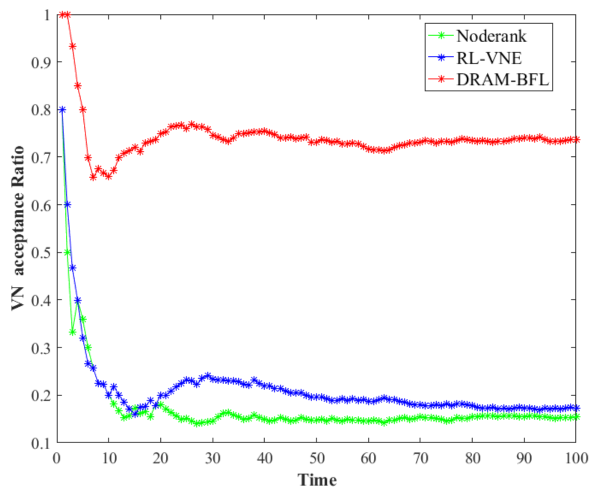

Fig. 9. VN acceptance rate with malicious nodes (PN=100).

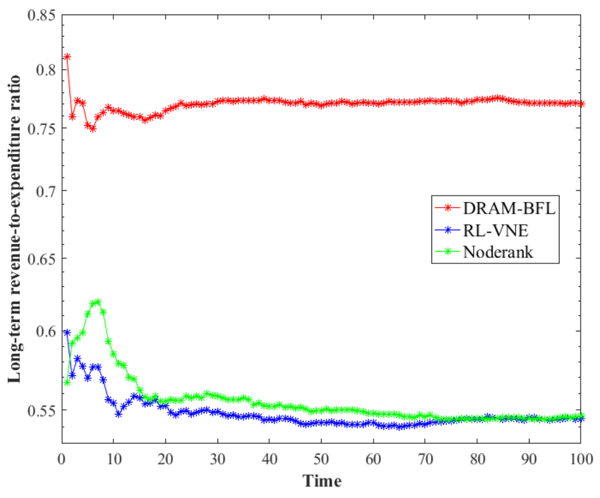

Fig. 10. Long-term revenue-to-expenditure ratio with malicious nodes (PN=100).

To evaluate the performance of DRAM-BFL, we compare DRAM-BFL with two methods, which are Noderank in [20] and RL-VNE in [27], under the network environment with malicious nodes and without malicious node respectively. We compare the performance of VN acceptance rate and long-term average revenue-to-expenditure ratio. We also increase the number of physical nodes and expand the scale of the physical network to test the scalability of DRAM-BFL.

We first evaluate the VN acceptance rate and long-term average revenue-to-expenditure ratio of each method over time. Then we set the physical network environment to 100 physical nodes (PN=100) and 200 nodes (PN=200), and simulate the scenarios with 5 malicious nodes and without malicious node respectively. In order to show the superiority of DRAM-BFL, we simulate its algorithm execution time and its number of communications, and compare DRAM-BFL with centralized VNE algorithms. Then, we conduct the delay and throughput experiments by using DRAM-FL under an ideal environment (no malicious nodes), and compare it with DRAM-FL with malicious nodes and DRAM-BFL with malicious nodes.

Figs. 7 and 8 show the superiority of DRAM-BFL in VN acceptance rate and long-term average revenue-to-expenditure ratio. In the ideal IoT environment without malicious nodes, we can see that VN acceptance rate of all three methods are reduced at first, this is because when more VNs arrive, the underlying network resources will be occupied more, and the available idle resources will be less. However, the long-term average revenue-to-expenditure ratio is relatively stable as it is independent of the available resources. As the underlying network gradually depletes, the performance of the three methods begins to stabilize. Notably, the VN acceptance rate of DRAM-BFL is 15.1% and 24.1% higher than that of Noderank and RL-VNE, and the long-term average revenue-to-expenditure ratio of DRAM-BFL is 25.8% and 41.8% higher than that of Noderank and RL-VNE respectively.

However, when malicious nodes are present in the IoT network, Figs. 9 and 10 show that the VN acceptance rate

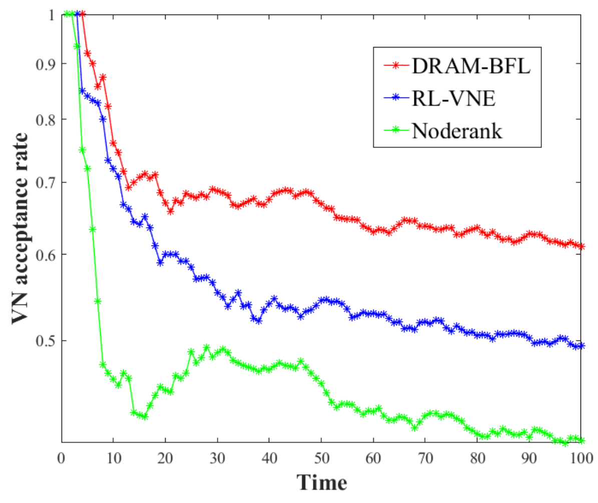

Fig. 11. VN acceptance rate with double physical nodes (PN=200).

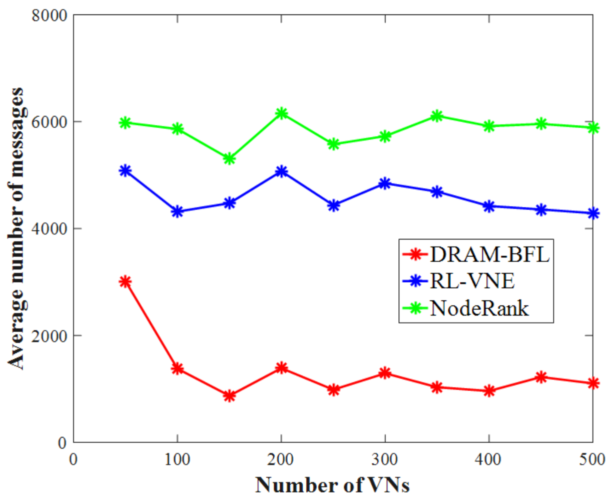

Fig. 13. Communication costs for different methods.

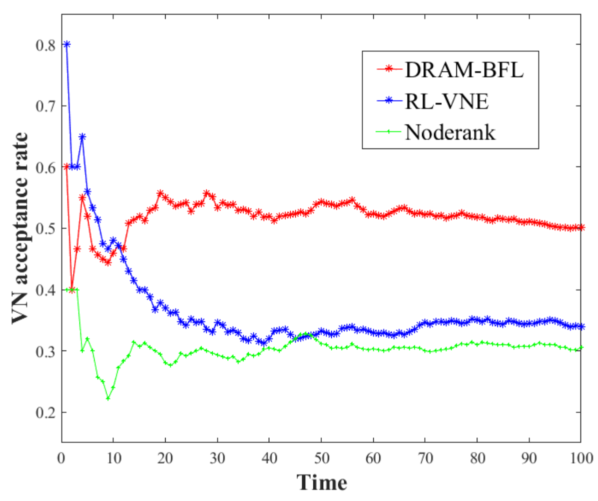

Fig. 12. VN acceptance rate with double physical nodes and malicious nodes (PN=200).

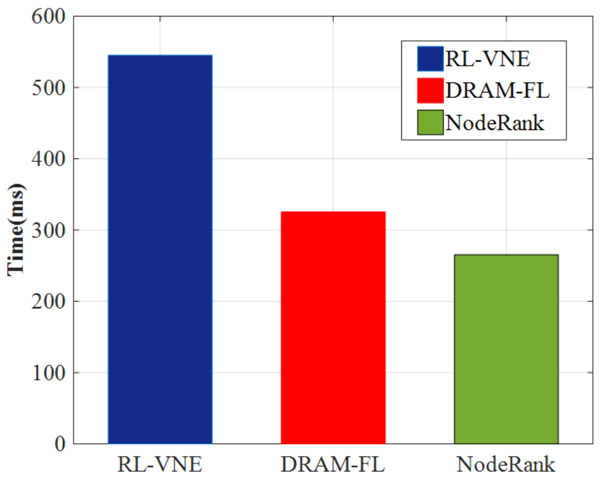

Fig. 14. Algorithm execution time for different methods.

of RL-VNE and Noderank significantly decreases. This is because malicious nodes cause security problems while RL-VNE and Noderank do not have blockchain protection. To address this problem, DRAM-BFL uses a node reliability assessment method, and prefers physical secure nodes for embedding. The method can effectively reduce the impact of malicious nodes on VNE performance.

Figs. 11 and 12 show the VN acceptance rate of three methods in the network with double physical nodes. We can see that, even with the increase of physical nodes and the underlying physical network resources, DRAM-BFL still outperforms the other two methods in terms of VN acceptance rate. The increase of physical nodes leads to a great increase of computation required to find suitable nodes and links during VNE. As the Noderank employs the greedy search strategy, the time required increase exponentially as the number of nodes increases. RL-VNE, relying on the embedding probability calculated by a centralized ML policy, needs to collect the feature information of all local nodes, resulting in double computing and storage resources requirements in double scale network.

Due to the direct correlation between computational cost and algorithm execution time, we evaluated the computational cost of DRAM-FL by algorithm execution time, and we evaluated the communication cost of DRAM-FL by the number of communications. Figs. 13 and 14 show the number of communications and the algorithm execution time of three methods.

Fig. 13 shows the average number of communications required by embedding a VN, and it can be seen that the communication costs of DRAM-BFL is only about 20% of Node Rank and about 25% of RL-VNE. That's because DRAM-BFL obtains local network information through the local training node collecting, while RL-VNE and Node Rank obtain local network information through all nodes uploading to central server. In Fig. 14, we can see that NodeRank takes the least execution time because it uses a simple search strategy, but

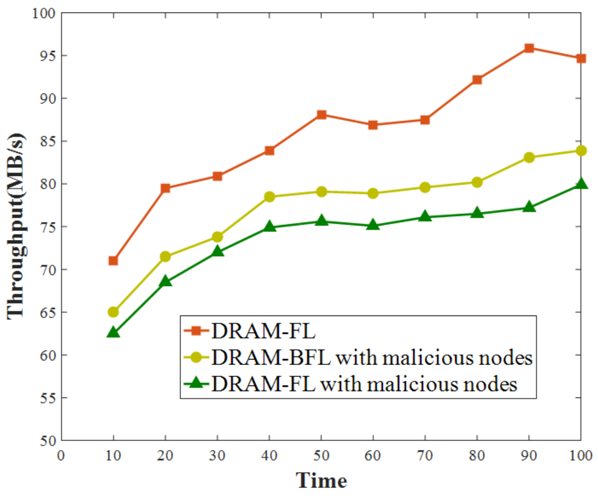

Fig. 15. Network throughput for different methods.

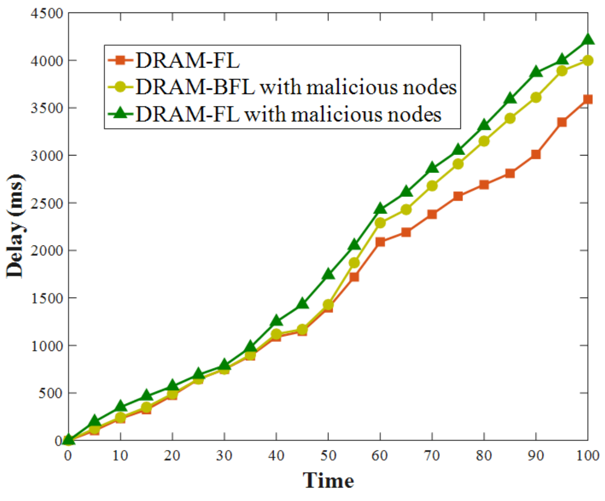

Fig. 17. Network delay for different methods.

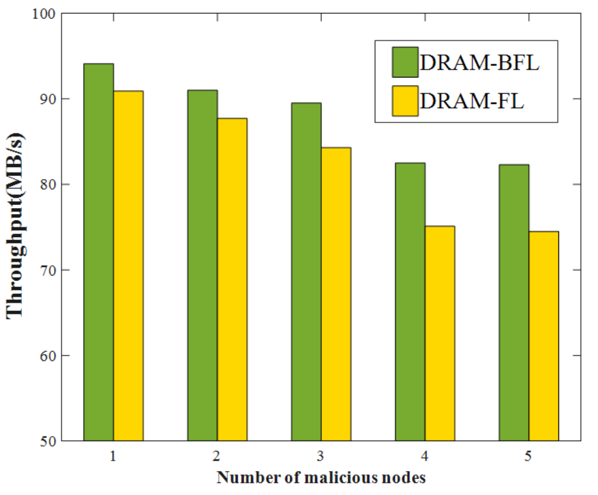

Fig. 16. Throughput with different number of malicious nodes.

its VN acceptance rate is poor. RL-VNE adopts a centralized training strategy, which increases the computational complexity, so it has more time-consuming than DRAM-FL. The time-consuming of DRAM-FL is 40.3% lower than that of centralized RL-VNE method. The nDRAM-BFL umber of communications of DRAM-FL is much lower than that of centralized algorithms RL-VNE and NodeRank. So, DRAM-FL can reduce the computational and communication costs when compared with centralized VNE algorithms.

Figs. 15–17 further illustrate the superiority of the DRAM-BFL blockchain system in the malicious node environment.

Fig. 15 shows that the throughput of DRAM-FL without malicious nodes is better than that of DRAM-BFL with malicious nodes and DRAM-FL with malicious nodes. Furthermore, due to the security protection of blockchain, the throughput of DRAM-BFL with malicious nodes is higher than that of DRAM-FL with malicious nodes.

Fig. 16 shows that, as the number of malicious nodes increases, the performance of DRAM-FL and DRAM-BFL decreases. This is because in a malicious node environment, the greater the amount of data, the more data that the malicious node handles, and the lower the throughput. However, DRAM-BFL can effectively avoid the selection of malicious nodes, so the throughput of DRAM-BFL scheme is always higher than that of DRAM-FL.

Fig. 17 shows the network delay of different schemes. As time increases, the delay of the DRAM-FL without malicious nodes is lower than that of DRAM-BFL with malicious nodes and DRAM-FL with malicious nodes. That's because, in the environment of malicious nodes, malicious nodes will choose non-optimal paths to transmit data, resulting in the increase of network transmission delay. In addition, due to the security protection of blockchain, the delay of DRAM-FL with malicious nodes is always lower than that of DRAM-BFL with malicious nodes.

Based on the above analysis, the proposed DRAM-BFL can not only improve the security of resource allocation in IoT, but also provide excellent performance and practical significance.

VII. Conclusion

Based on the characteristics of resource fragmentation in IoT, we use network virtualization technology to model the resource allocation problem as a VNE optimization problem. Considering the restricted terminal resource and security problems, we introduce blockchain and FL into VNE and propose DRAM-BFL to solve this VNE optimization problem. In DRAM-BFL, a dual chain structure is proposed to protect network topology information and model parameter information, a node reliability assessment method and EPBFT-NRA consensus algorithm are proposed to improve security performance. Finally, simulation results show that DRAM-BFL has good performance in VN acceptance rate, revenue-to-expenditure ratio, network throughput and delay while improving system security.

References

[1] Z. Sheng *et al.*, "A survey on the IETF protocol suite for the Internet of things: Standards, challenges, and opportunities," *IEEE Wireless Commun.* vol. 20, no. 6, pp. 91–98, 2013.

[2] L. S. Vailshery, "IoT connected devices worldwide 2030." https://www.statista.com/statistics/802690 /worldwide-connected-devices-by-access-technology/ (accessed Jan. 22, 2021).

[3] N. Cveticanin, "30 big data statistics everybody's talking about." https://dataprot.net/statistics/datastatistics/ (accessed Mar. 8, 2022).

[4] L. Gao, G. Iosifidis, J. Huang, and L. Tassiulas, "Economics of mobile data offloading," in *Proc. IEEE INFOCOM*, 2013.

[5] H. Tahaei, F. Afifi, A. Asemi, F. Zaki, and N. B. Anuar, "The rise of traffic classification in IoT networks: A survey," *J. Netw. Comput. Appl.*, vol. 154, p. 102538, 2020.

[6] H. Elazhary, "Internet of things (IoT), mobile cloud, cloudlet, mobile IoT, IoT cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions," *J. Netw. Comput. Appl.*, vol. 128, pp. 105–140, 2019.

[7] E. A. Kosmatos, N. D. Tselikas, and A. C. Boucouvalas, "Integrating RFIDs and smart objects into a unified Internet of things architecture," *Adv. Internet Things*, vol. 1, no. 2, pp. 5–12, 2011.

[8] A. Belbekkouche, M. Hasan, and A. Karmouth, "Resource discovery and allocation in network virtualization," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 1114–1128, 2012.

[9] Y. Wang, Q. Hu, L. Nguyen, and M. Jalalitabar, "Minimum-cost embedding of virtual networks: An iterative decomposition approach," *Comput. Netw.*, vol. 234, p. 109907, 2023.

[10] Y. Guan *et al.*, "Multidimensional resource fragmentation-aware virtual network embedding for IoT applications in MEC networks," *IEEE Internet Things J.*, vol. 10, no. 24, pp. 22223–22232, 2023.

[11] X. Hesselbach and D. Escobar-Perez, "Machine learning requirements for energy-efficient virtual network embedding," *Energies* vol. 16, no. 11, p. 4439, 2023.

[12] M. Lu and M. Li, "A multiple QoS metrics-aware virtual network embedding algorithm," *Comput. J.*, 2023.

[13] Z. Duan and T. Wang, "Towards learning-based energy-efficient online coordinated virtual network embedding framework," *Comput. Netw.*, vol. 239, p. 110139, 2024.

[14] P. Zhang *et al.*, "CE-VNE: Constraint escalation virtual network embedding algorithm assisted by graph convolutional networks," *J. Netw. Comput. Appl.*, vol. 221, p. 103736, 2024.

[15] B. Cao, L. Zhang, T. Q. S. Quek, and S. Yang, "Guest editorial special issue on blockchain-enabled Internet of things," *IEEE Internet things J.*, vol. 9, no. 11, pp. 7876–7878, 2022.

[16] J. Duan, Z. Guo, and Y. Yang, "Cost efficient and performance guaranteed virtual network embedding in multicast fat-tree DCNs," in *Proc. IEEE INFOCOM*, 2015.

[17] Z. Yan, J. Ge, Y. Wu, L. Li, and T. Li, "Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1040–1057, 2020.

[18] Z. Li, Z. Lu, S. Deng, and X. Gao, "A self-adaptive virtual network embedding algorithm based on software-defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 1, pp. 362–373, 2019.

[19] X. Cheng *et al.*, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, 2011.

[20] W. Wei, H. Gu, K. Wang, X. Yu, and X. Liu, "Improving cloud-based IoT services through virtual network embedding in elastic optical inter-DC networks," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 986–996, 2019.

[21] P. Zhang *et al.*, "RKD-VNE: Virtual network embedding algorithm assisted by resource knowledge description and deep reinforcement learning in IIoT scenario," *Future Generation Comput. Syst.*, vol. 135, pp. 426–437, 2022.

[22] H. Yao, X. Chen, M. Li, P. Zhang, and L. Wang, "A novel reinforcement learning algorithm for virtual network embedding," *Neurocomputing*, vol. 284, pp. 1–9, 2018.

[23] J. Liu *et al.*, "A Bayesian Q-learning game for dependable task offloading against DDoS attacks in sensor edge cloud," *IEEE Internet things J.*, vol. 8, no. 9, pp. 7546–7561, 2021.

[24] G. Dandachi, A. Rkhami, Y. Hadjadj-Aoul, and A. Outtagarts, "A robust Monte-Carlo-based deep learning strategy for virtual network embedding," in *Proc. IEEE LCN*, 2022.

[25] C. Wang, X. Chen, Y. Luo, and G. Zhang, "Solving virtual network mapping fast by combining neural network and MCTS," in *Proc. CBD*, 2022.

[26] S. Haeri and L. Trajković, "Virtual network embedding via Monte Carlo tree search," *IEEE Trans. Cybern.*, vol. 48, no. 2, pp. 510–521, 2018.

[27] P. Zhang *et al.*, "Reinforcement learning assisted bandwidth aware virtual network resource allocation," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 4, pp. 4111–4123, 2022.

[28] Q. Gao, N. Lyu, J. Miao, and W. Pan, "Wireless virtual network embedding algorithm based on deep reinforcement learning," *Electronics*, vol. 11, no. 14, p. 2243, 2022.

[29] P. Zhang, C. Wang, C. Jiang, and A. Benslimane, "Security-aware virtual network embedding algorithm based on reinforcement learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1095–1105, 2021.

[30] H. K. Thakkar, C. K. Dehury, and P. K. Sahoo, "MUVINE: Multi-stage virtual network embedding in cloud data centers using reinforcement learning-based predictions," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1058–1074, 2020.

[31] L. R. Bays, R. R. Oliveira, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "A heuristic-based algorithm for privacy-oriented virtual network embedding," in *Proc. IEEE NOMS*, 2014.

[32] P. Rezaeimoghaddam and I. Al-Anbagi, "Trust-aware virtual network embedding in wireless sensor networks," *IEEE Sensors J.*, vol. 23, no. 6, pp. 6326–6337, 2023.

[33] J. Zhang, Y. Liu, X. Qin, X. Xu, and P. Zhang, "Adaptive resource allocation for blockchain-based federated learning in Internet of things," *IEEE Internet things J.*, vol. 10, no. 12, pp. 10621–10635, 2023.

[34] X. Fu *et al.*, "Federated learning-based resource management with blockchain trust assurance in smart IoT," *Electronics*, vol. 12, no. 4, p. 1034, 2023.

[35] H. Li *et al.*, "Privacy-preserving cross-silo federated learning atop blockchain for IoT," *IEEE Internet things J.*, vol. 10, no. 24, pp. 21176–21186, 2023.

[36] H. Kasyap and S. Tripathy, "Privacy-preserving and Byzantine-robust federated learning framework using permissioned blockchain," *Expert Syst. Appl.*, vol. 238, Part E, p. 122210, 2024.

[37] P. Zhang, C. Wang, C. Jiang, N. Kumar, and Q. Lu, "Resource management and security scheme of ICPSs and IoT based on VNE algorithm," *IEEE Internet things J.*, vol. 9, no. 22, pp. 22071–22080, 2022.

[38] H. Wu and P. Wang, "Node selection toward faster convergence for federated learning on non-IID data," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3099–3111, 2022.

[39] A. Clement, E. L. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, "Making Byzantine fault tolerant systems tolerate Byzantine faults," in *Proc. NSDI*, 2009.

[40] H. Liao *et al.*, "Blockchain and semi-distributed learning-based secure- and low-latency computation offloading in space-air-ground-integrated power IoT," *IEEE J. Sel. Topics Signal Process.*, vol. 16, no. 3, pp. 381–394, 2022.

[41] M. O'Sullivan, L. Aniello, and V. Sassone, "A methodology to select topology generators for ad hoc mesh network simulations," *J. Commun.*, vol. 15, no. 10, pp. 741–746, 2020.

Hui Zhi received the MS degree and PhD degree in communication and information engineering from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2009 and in 2013, respectively.

Since 2013, she has been a teacher at Anhui University, Hefei, China. She is currently a researcher in the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, Anhui University. Her current research interests include block chain, cooperative communications, network coding and wireless sensor networks.

Yaning Wang received a bachelor's degree in communication engineering from Changsha University of Science and Technology in 2019 and is currently studying for a master's degree in electronics and communication engineering at Anhui University, China. Since 2021, she has been engaged in wireless communication research at theKey Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, Anhui University. Her research interests include network, resource, and service management in wireless networks and network resource allocation.

Ms. Wang's awards and honors include the Academic Scholarships of Anhui University and Changsha University of Science and Technology.