applied sciences

MDPI

Article

Blockchain-Based Decentralized Federated Learning Method in Edge Computing Environment

Song Liu, Xiong Wang, Longshuo Hui and Weiguo Wu *

School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China; liusong@xjtu.edu.cn (S.L.); buer_wang@163.com (X.W.); huils20@stu.xjtu.edu.cn (L.H.)
* Correspondence: wgwu@xjtu.edu.cn; Tel.: +86-187-0683-8060

Abstract: In recent years, federated learning has been able to provide an effective solution for data privacy protection, so it has been widely used in financial, medical, and other fields. However, traditional federated learning still suffers from single-point server failure, which is a frequent issue from the centralized server for global model aggregation. Additionally, it also lacks an incentive mechanism, which leads to the insufficient contribution of local devices to global model training. In this paper, we propose a blockchain-based decentralized federated learning method, named BD-FL, to solve these problems. BD-FL combines blockchain and edge computing techniques to build a decentralized federated learning system. An incentive mechanism is introduced to motivate local devices to actively participate in federated learning model training. In order to minimize the cost of model training, BD-FL designs a preference-based stable matching algorithm to bind local devices with appropriate edge servers, which can reduce communication overhead. In addition, we propose a reputation-based practical Byzantine fault tolerance (R-PBFT) algorithm to optimize the consensus process of global model training in the blockchain. Experiment results show that BD-FL effectively reduces the model training time by up to 34.9% compared with several baseline federated learning methods. The R-PBFT algorithm can improve the training efficiency of BD-FL by 12.2%.

Keywords: decentralized federated learning; blockchain; edge computing; stable matching; consensus algorithm

Citation: Liu, S.; Wang, X.; Hui, L.; Wu, W. Blockchain-Based Decentralized Federated Learning Method in Edge Computing Environment. *Appl. Sci.* **2023**, *13*, 1677. https://doi.org/10.3390/app13031677

Academic Editors: George Drosatos, Avi Arampatzis and Pavlos S. Efraimidis

Received: 15 January 2023
Revised: 24 January 2023
Accepted: 26 January 2023
Published: 28 January 2023

1. Introduction

With the development of new generation information technology such as mobile internet, the number of mobile services and applications is growing exponentially, resulting in the generation of massive amounts of data. It has been shown that there are data security risks in well-known business associations [1]. User data are often stored in the centralized cloud servers of an organization or enterprise and can be accessed without privacy protection, which raises the risk of leakage of user-sensitive data [2]. Google took the lead in proposing a federated learning method [3] to solve the collaborative training problem of privacy protection so that private data can be safely used in a distributed environment. Therefore, federated learning has received extensive attention and research from industry and academia. However, federated learning has its own limitations [4]. It relies on a single centralized server and is vulnerable to a single point of server failure. Additionally, it lacks the incentive mechanism for local devices, which leads to the reduction of the initiative of local devices to participate in the training of federated learning.

Blockchain is a decentralized and auditable ledger technique [5] that has become a solution to replace vulnerable centralized servers in insecure environments. By combining with blockchain, the decentralized federated learning method can be realized. However, in the scenario of combined blockchain and federated learning, distributed servers usually use cloud computing to transmit model data [4]. Since cloud servers are physically far from local devices, which will further increase the delay of network data transmission [6],

traditional cloud computing techniques will weaken the role of servers in supporting the training process of federated learning models. Additionally, edge computing has become an efficient computing paradigm that sinks computing and storage resources to the side close to local devices [7,8]. Compared to cloud-based servers, edge servers are closer to local devices and can respond to requests from local devices faster. Applying edge computing to federated learning can further reduce the latency and energy consumption of model training.

However, when combining blockchain with federated learning in the edge computing environment, it also faces the challenge of the high cost of network communication [9]. As the number of local devices is large and keeps growing dramatically, local devices need to effectively offload data and tasks to the appropriate server close to the edge side of the network, optimizing the utilization of edge server resources and maximizing the system efficiency. This requires efficient and stable matching between local devices and edge servers [10]. The matching problem is affected by many factors, such as distance, network bandwidth, and computing power [11]. These factors involve new techniques such as node localization [12], and they need to be considered to reduce the overall system latency and energy consumption in the edge computing environment. Meanwhile, as the core of blockchain technique, the consensus algorithm plays a decisive role in the security and efficiency of blockchain [13–16]. In highly decentralized federated learning with blockchain, all local devices participating in model training will also participate in the consensus process. The communication cost will increase significantly, which is bound to increase the consensus time of the blockchain, thus reducing the efficiency of model training.

To address the problems of centralization and lack of incentives in traditional federated learning, this paper proposes a blockchain-based decentralized federated learning method for edge computing environments, named BD-FL. By designing a stable matching algorithm between local devices and edge servers and an optimized consensus algorithm, BD-FL can effectively reduce the overall system delay and speed up the model training efficiency. This paper makes the following contributions.

- We propose the BD-FL by combining blockchain with federated learning in the edge computing environment. BD-FL uses the distributed characteristics of blockchain and edge computing to solve the problem of a centralized server in that the local device trains the local model and the edge server aggregates the global model. BD-FT also introduces an incentive mechanism to encourage local devices to actively participate in model training, increasing the number of samples and improving the model accuracy.
- We propose a preference-based stable matching algorithm in BD-FT, which binds local devices to appropriate edge servers, improving the utilization of edge server resources and reducing the delay of data transmission. We propose the R-PBFT algorithm, which optimizes the network topology and the consistency protocol and designs a dynamic reputation mechanism, reducing the communication overhead of the blockchain consensus process and improving the model training efficiency.
- We performed extensive simulation experiments to evaluate the proposed BD-FL. Experimental results show that BD-FL effectively reduces the model training time by up to 19.7% and 34.9%, respectively, compared with several federated learning methods with different matching algorithms and a state-of-the-art blockchain-based federated learning method. The R-PBFT algorithm can reduce the communication overhead of the consensus process and improve the training efficiency of BD-FL by 12.2%.

The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 presents the methodology, including the system architecture, BD-FL, and R-PBFT. Section 4 gives the experimental evaluation. Section 5 concludes this paper.

2. Related Work

A centralized network topology has more serious system security issues and higher communication overhead compared with a decentralized distributed network topology. Traditional federated learning adopts a centralized topology with a single centralized server

responsible for aggregating the global model. As federated learning takes hold in real-life production, the shortcomings of the centralized topology model have gradually become apparent [17]. In practical applications, the centralized server will put huge pressure on the network bandwidth overhead and also reduce the robustness of the system, which will affect the model training process of federation learning once the server is maliciously compromised. Blockchain is a distributed technique with decentralized characteristics and incentive mechanism [5,18,19], which can effectively solve the above problems of traditional federated learning. In addition, due to being tamper-proof and anonymous, blockchain can guarantee data security. Therefore, a lot of work has been done on decentralized federated learning methods using blockchain.

However, decentralized federated learning with blockchain still faces the challenge of model training efficiency. Kim et al. [20] proposed a blockchain-based federated learning architecture. The local device in this architecture updates the local model based on its available data samples. The architecture uses blockchain to reward updates from local devices, and the reward is proportional to the number of local data samples. This simple reward scheme is not able to accurately reflect the magnitude of the contribution made by the local device to the global model training. Weng et al. [21] designed a federated learning scheme incorporating blockchain incentives. The scheme ensures system reliability by rewarding honest local devices and punishing dishonest ones. They also introduced a consensus protocol based on a committee mechanism, which participates in the consensus process by randomly selecting nodes to form a committee. However, randomly selecting committee nodes is almost negligibly close to a completely random scheme and is probably not optimal. Local devices make significant contributions to the training of the federated learning model, so it is important to reasonably reward local devices. Existing federated learning incentive schemes generally agree that local devices should be fairly rewarded based on the magnitude of their contribution to the model. Jia et al. [22] stated that the most widely used scheme to evaluate the contribution size of local devices is Shapley values (SVs). SVs can fairly distribute rewards for model training, and it is widely used in many fields, such as economics, information theory, and machine learning. However, SV-based reward schemes usually require exponential time to compute, and the computational cost is prohibitive.

In the edge computing environment, the matching problem between local devices and edge servers has an important impact on the data transmission delay and the overall system efficiency. Hu et al. [23] proposed a matching method for mobile edge computing and device-to-device communication environments. This method uses a game model to solve the offloading problem of local devices. Each local device is regarded as a gamer, and the offloading strategy is obtained through a mutual game to make the system reach Nash equilibrium. Wu [24] proposed an intelligent scheduling matching scheme based on a delayed acceptance algorithm. It combines the Hopfield neural network and the decision tree model and quantifies the assignment scheduling problem into a matching optimization problem by defining the cost coefficient between tasks and equipment. Lu et al. [25] proposed an asynchronous greedy matching algorithm, which builds a preference list of both parties based on the utility value between the cooperative node and the requesting node, and uses the greedy strategy for stable matching. The existing related research mainly solves the one-to-one or N-to-N matching problem, and there is less research on stable matching between M devices and N servers in the edge computing environment.

The consensus algorithm, as one of the core ideas of blockchain, can ensure the proper operation of the blockchain, but it has an important impact on communication overhead and model training efficiency. The most commonly used consensus algorithms are proof of stake (PoS) [13], proof of work (PoW) [14], delegated proof of stake (DPoS) [15], and practical Byzantine fault tolerance (PBFT) [16]. The PBFT algorithm is widely used in distributed architectures, but it still suffers from high communication overhead and low reliability of master nodes. Numerous solutions have emerged to address the shortcomings of PBFT. Castro et al. [26] improved the transaction throughput of PBFT by caching blocks, but their

method does not perform well on the delay of the consensus process. Zhang et al. [8] made the certificate and other information of blocks clear in time without communication between nodes according to the timestamp in the blockchain, but this method does not consider the optimization of the PBFT consistency protocol. Zheng et al. [27] combined DPoS and PBFT to make the algorithm with dynamic authorization, but the limited bandwidth still reduces the transaction throughput of the algorithm. On the other hand, some studies are dedicated to reducing the time complexity of PBFT. Ma et al. [28] proposed a scheme to verify the consistency of asynchronous Byzantine nodes through a stochastic prediction model. This scheme randomly selects one of the multi-node proposals in each round of consensus and uses the threshold signature algorithm to reduce the communication cost of each round of consensus to $O(n^2)$. However, the random selection method of this scheme may cause security problems. Gao et al. [29] proposed a scalable Byzantine algorithm (FastBFT) that introduces a tree data structure to achieve the optimal time complexity of $O(nlogn)$. However, in the worst case, the time complexity of FastBFT is still $O(n^2)$. Liu et al. [30] improved the consensus efficiency by caching blocks and smart contract techniques, but without reducing the time complexity of PBFT. Wang et al. [31] proposed a PBFT algorithm based on randomly selected collectors. It can reduce the communication cost to a linear level, but if the selected collector is malicious, the communication cost of the algorithm will rise sharply. However, these consensus algorithms are not applicable to real blockchain and edge computing application scenarios, and still have high computational complexity.

In this work, we present BD-FL to solve the single node failure and network communication overhead problems of centralized federated learning. BD-FL introduces an incentive mechanism to increase the contribution of local devices to global model training. We also propose a stable matching algorithm and the R-PBFT algorithm to reduce the number of nodes participating in communication and consensus, which reduces the system delay and improves the model training efficiency.

3. Methodology

3.1. System Architecture

To implement blockchain-based decentralized federated learning, we first designed the system architecture. Figure 1 shows the architecture of BD-FL, which mainly includes the demand release module, aggregation module, training module, and verification module. Since the nodes participating in the consensus in BD-FL are only a limited number of edge servers, the alliance chain is selected as the implementation platform of the architecture.

The demand release module is mainly composed of model demanders, whose main role is to release the demand task and pay the model training fee and the verification fee. After the model demander pays the fee and provides the initial model data, the system will send the relevant information of the initial model to each edge server for download by local devices. The aggregation module is mainly composed of edge servers that are close to local devices or data sources, such as base stations with certain computing and storage capabilities. It will save the gradient parameters of the local model, and other block data uploaded by local devices, and aggregate the global model on edge servers. It will also verify the accuracy of the uploaded gradient parameters and prevent dishonest local devices from maliciously providing wrong information. In the consensus process of blockchain, the local device will not participate, and the edge servers of the aggregation module will participate in the consensus to reduce the system communication delay. The training module is mainly composed of local devices, and its main role is to train local models using local data samples. In the model training, the system will bind local devices with edge servers according to our proposed matching algorithm. The local device will only upload the local model parameters to its bound edge server and only download the global model from its bound edge server. The verification module is also composed of some local devices. In each round of global model aggregation, the edge server sends the received local model gradient parameters uploaded by the local device to the verification

device, which uses its own dataset to verify the quality of the local model, and returns the results to its bound edge server.

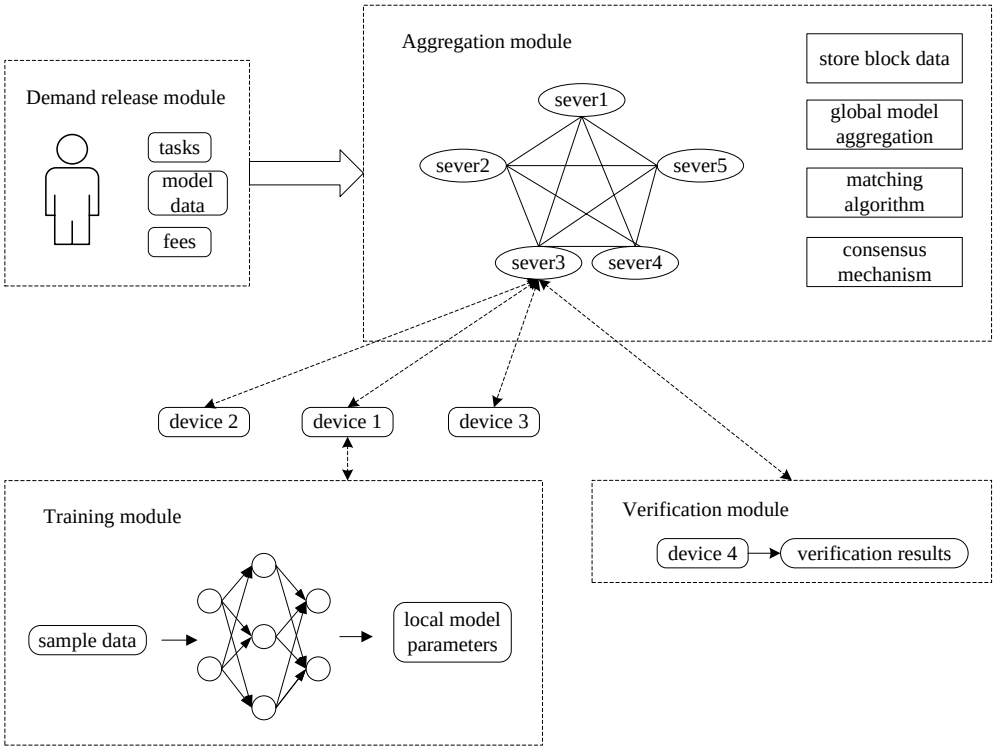

Figure 1. Architecture diagram of the decentralized federated learning.

3.2. Incentive Mechanism

In order to effectively promote local devices to participate in model training and verification, BD-FL has designed an incentive mechanism. During each round of training, the system will store the verification results returned by the verification device, the number of samples uploaded by the training device, and the training time of the device in the block. After a round of global model aggregation, the system will read the data saved on the block, calculate the reward of each training device according to the incentive mechanism, and send it to each local device.

The incentive mechanism gives corresponding rewards or punishments according to the contribution of local devices to model training. During the federated learning process, in order to ensure that the verification devices can give honest reports, their verification results can be re-verified by other devices, and dishonest validation behaviors will be punished. At the same time, in order to improve the fairness of reward distribution, the system will allocate the training fee according to the size of contribution made by each training device. The incentive mechanism introduces two metrics to calculate the final profit of the training device.

The first is the number of data samples owned by the training device. Devices with more data samples contribute more to global training, take longer time to train local models, and cost more. The second one is the accuracy of the model corresponding to the gradient parameters uploaded by the training device. The edge server verifies the model accuracy of the training device using the dataset of the verification device and returns the results to the edge server. An accuracy threshold T is introduced as a standard to measure whether the local model parameters of the training devices are qualified.

The system assigns the training fee S by scoring each training device. The score is related to the training time of the local device (denoted as *trainTime*) and the accu-

racy of the uploaded local model parameters (denoted as $accValue$). It is calculated by Equation (1), where α and β are the score coefficients, and the sum of them is 1.

$$S = \begin{cases} \alpha * trainTime + \beta * accValue \\ \alpha + \beta = 1 \ \ (0 < \alpha, \beta < 1) \end{cases}. \tag{1}$$

Equation (2) gives the calculation of the training reward R^k_{train} that an honest training device (device number is denoted as k) should receive in a training round, where m is the number of training devices, and P_{FLM} is the total cost of the model, which is paid to the public account of the system when the demander releases the task.

$$R^k_{train} = \begin{cases} 0 & (accValue < T) \\ P_{FLM} * \left(S_k / \sum_{i=1}^{m} S_i \right) & (accValue \geq T) \end{cases}. \tag{2}$$

3.3. Preference-Based Stable Matching Algorithm

Since local devices need to transmit a large amount of data to edge servers, the network quality, the transmission distance between nodes, the server throughput, and other factors will largely affect the overall delay and energy consumption of the system. We design a preference-based stable matching algorithm in BD-FL, which considers the above factors that affect data transmission, so as to achieve the optimal matching and binding between local devices and edge servers.

In the network environment, local devices and edge servers are abstracted into two sets, which are, respectively, denoted by the set of local devices $K = \{k_1, k_2, \ldots, k_m\}$ and the set of edge servers $S = \{s_1, s_2, \ldots, s_n\}$. For each $k \in K$, it has a matching request, denoted as Equation (3),

$$Q_k = (D_k, Info_k), \tag{3}$$

where D_k represents the data of uplink communication that local device k uploads to the edge server, mainly including information such as local model parameters, version number, local iteration time, etc.; $Info_k$ represents the state information of local device k, such as bandwidth, physical location, etc.

The uplink communication rate v_{ks} determines the data transmission delay, which is expressed as Equation (4) according to [32],

$$v_{ks} = band_k * \log_2 \left(1 + \frac{p_k * g_{ks}}{N_0} \right), \tag{4}$$

where $band_k$ represents the channel bandwidth allocated to the local device k, p_k represents the transmit power of k, g_{ks} represents the channel gain between k and the edge server s, and N_0 represents the noise power of s.

According to [32], the network distance between nodes is related to the channel bandwidth. It reflects the actual distance and network condition between local devices and edge servers. Assuming that $phys_k$ and $phys_s$ are the physical nodes corresponding to the local device and the edge server, respectively, the network distance $distNode_{ks}$ between them is defined as Equation (5),

$$distNode_{ks} = \frac{band_{aver}}{(band_k + band_s)/2} * distPhys_{ks}, \tag{5}$$

where $distPhys_{ks}$ represents the actual distance between $phys_k$ and $phys_s$, which has a certain influence on the reliability of data transmission and the network latency, $band_{aver}$ represents the average bandwidth between them, and $band_k$ and $band_s$ represent the actual bandwidths of $phys_k$ and $phys_s$.

Equations (6) and (7) express the upload time and the upload energy consumption, respectively, after local device k and edge server s are bound.

$$T_{ks} = \frac{D_k}{v_{ks}}\,, \tag{6}$$

$$E_{ks} = p_k * T_{ks} = p_k * \frac{D_k}{v_{ks}}\,. \tag{7}$$

We use the sum of weighted energy consumption and weighted delay, denoted as w_k, as the cost function of k, which is expressed by Equation (8),

$$\omega_k = \mu_k^e * E_k + \mu_k^t * T_k\,, \tag{8}$$

where $\mu_k^e, \mu_k^t \in [0,1], \mu_k^e + \mu_k^t = 1$, μ_k^e and μ_k^t, represent the energy consumption weight and the delay weight of k, respectively. A larger μ_k^e indicates a higher priority of energy consumption, and a larger μ_k^t indicates a higher priority of latency, and vice versa. The priority of energy consumption and delay of the local device can be adjusted by modifying the value of μ.

We define a coefficient matrix W to represent the cost relationship between local devices and edge servers. It is expressed by Equation (9), where w_{ks} denotes the cost when device k is bound to server s, and its value can be calculated by Equation (8).

$$W = (\omega_{ks})_{m*n} = \begin{bmatrix} \omega_{11} & \omega_{12} & \dots & \omega_{1n} \\ \omega_{21} & \omega_{22} & \dots & \omega_{2n} \\ \dots & \dots & \dots & \dots \\ \omega_{m1} & \omega_{m2} & \dots & \omega_{mn} \end{bmatrix}. \tag{9}$$

We also define an assignment matrix X to represent the matching relationship between local devices and edge servers. It is expressed by Equation (10), where x_{ks} denotes the matching relationship between device k and server s, and $x_{ks} = 1$ means k is bound to s, otherwise, k is not bound to s.

$$X = (x_{ks})_{m*n} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix}. \tag{10}$$

The goal of the matching algorithm is to minimize of the total cost of model training, while also satisfying the constraint that the total number of local devices bound to edge server s cannot exceed its maximum number of device bindings. Therefore, we derive the optimization objective function, which is expressed by Equation (11).

$$\min obj = \sum_{k=1}^{m} \sum_{s=1}^{n} \omega_{ks} * x_{ks}\,,\ (x_{ks} = 0\ or\ 1)\,. \tag{11}$$

We propose a preference-based stable matching algorithm to solve the optimization objective function of cost minimization. Each local device k will establish a preference list of matching degrees for all edge servers. We use a binary (k, s) to represent a match between device k and edge server s. Equation (12) gives the preference function of k for s, and $P_{(k,s)}$ represents the matching degree of (k, s). Equation (13) is the preference list of k, i.e., Γ_k. The matches are sorted in descending order of matching degree, that is, the higher the match in the list, the higher the preference.

$$P_{(k,s)} = \eta E_{ks} + (1-\eta) T_{ks}\,, \tag{12}$$

$$\Gamma_k = [P_{(k,s)}, \ \forall s \in S]. \tag{13}$$

Similarly, each edge server s will establish a matching preference list for all local devices, i.e., Γ_s, according to the preference function $P_{(s,k)}$. We only consider the strict partial order, that is, a local device will not have the same matching degree with two edge servers.

The local device tends to match the edge server with the highest matching degree, and the edge server also tends to bind to the local device with the highest matching degree. However, in the matching process, we can only pay more attention to the needs of one side. In order to make the local device have a better experience in the training process, the matching algorithm should meet the needs of the local device as much as possible, that is, the local device has priority to match.

Algorithm 1 describes the preference-based stable matching algorithm. All local devices and edge servers are initialized as unmatched. Then, all devices and servers broadcast their status information to each other, and each device and server establishes its preference lists. Each local device k sends a matching request to the edge server with the highest matching degree according to its Γ_k. If the request is rejected, it sends the matching request to other edge servers again according to their preference lists. Each edge server first places all devices that it receives the matching request into its match list. If the number of devices in the match list is greater than its maximum binding number, the edge server will pre-enroll devices according to its preference list from front to back, and reject the devices with low matching degrees in its Γ_s until the number of devices in the match list equals its maximum binding number. If the number of local devices in the match list is less than the maximum binding number, all devices are reserved. Each edge server repeats updating its match list until all local devices are in the match lists of edge servers. The output matches of the algorithm are Pareto optimal [33]. The time complexity of the algorithm is $O(nm) + O(n + m)$.

Algorithm 1 Preference-based stable matching algorithm.

Input: local device set K, edge server set S, maximum binding number of the server L_s.
Output: match list of edge servers N_s.

All items in K and S are initialized as unmatched;
All devices and servers broadcast their status information to each other;
Each k and s establish the preference lists Γ_k and Γ_s;
$i = 0, N_s = \emptyset, N_k = K$; $//N_k$ is an unmatched device list
while $\mid N_k \mid > 0$ **do**
  for $k \in N_k$ **do**
    $i = i + 1$; $//k$ sends a matching request to the edge server with the highest matching degree in its Γ_k
    k sends Q_k to s_i in its Γ_k;
    $N_s = N_s \bigcup k$;
    $N_k = N_k - k$;
  end for
  for $s \in N_s$ **do**
    while $N_s > L_s$ **do**
      $//k'$ is the device with lowest matching degree in its Γ_s
      $k' = argmin_{k \in N_s} \Gamma_s$;
      $N_s = N_s - k'$;
      $N_k = N_k \bigcup k'$;
    end while
  end for
end while
return N_s;

3.4. R-PBFT Consensus Algorithm

To improve the model training efficiency of BD-FL, we propose a reputation-based practical Byzantine fault tolerance (R-PBFT) consensus algorithm. In the BD-FL environment, due to the large number of nodes, the traditional PBFT algorithm will lead to a sharp increase in communication cost and network bandwidth consumption in the consensus process, which is easy to cause network congestion and increase the delay of model training. In addition, the master node election in PBFT adopts the modulus calculation, which cannot guarantee the optimal master node of the election, thus affecting the consistency and reducing the reliability and security of the system.

The R-PBFT consensus algorithm combines the characteristics of BD-FL and designs the following improvements to solve the shortcomings of traditional PBFT for BD-FL.

- Remove the client node. In the traditional PBFT algorithm, the request phase and the reply phase occur between the client and the master nodes. However, in the blockchain structure, information is broadcast between nodes in the form of P2P, without the participation of the client. Therefore, we remove the request and reply phases of the client node in the consistency protocol, modify the C/S structure of PBFT to a distributed topology, and divide all nodes into master and slave nodes.
- Optimize the consistency protocol. The five phases of consensus in PBFT are changed to three phases, including the pre-preparation phase, preparation phase, and confirmation phase. In the pre-preparation phase, the master node broadcasts blocks to other slave nodes. In the preparation phase, the slave node broadcasts the block verification results to other slave nodes and master nodes. In the confirmation phase, traditional PBFT requires mutual interaction between nodes. We simplify it as all slave nodes send verification results to the master node, and the master node makes a decision on the consensus results, thus reducing the communication overhead of consensus.
- Introduce reputation mechanism. The main purpose of the reputation mechanism is to make the nodes with high reliability easier to be elected as the master node. Each node will be divided into different reputation levels according to the reputation value, and then each node will be rewarded or punished based on its performance in each round of consensus. According to a preset reputation threshold, nodes can be dynamically transformed in different reputation levels.

When the current round of consensus is completed, the system performs the reputation mechanism to assign each node to different reputation levels according to the reputation value and then selects the node with the highest score among the trusted nodes as the master node for the next round of consensus. Figure 2 shows the execution flow of a round of consensus, where P denotes the master node and S denotes the slave node.

According to the reputation value R of each node, the reputation mechanism divides it into three different reputation levels, namely trusted node, normal node, and unreliable node. R is a real number between 0 and 1, and its size reflects the reliability of the node. For the trusted node, the range of R is (0.8,1], and the node of this level generated valid blocks multiple times. For the normal node, the range of R is (0.3,0.8], and the node of this level generated unqualified blocks, but less often. For the unreliable nodes, the range of R is [0,0.3], and the node of this level generated unqualified blocks many times. The unreliable node will not participate in the election of the master node and the consensus process of the blockchain and only saves block data.

Each node updates its R according to the following rules after a consensus, so as to dynamically transform in different reputation levels. (1) The R of the node will be increased by 0.01 for each successful consensus participation. (2) If the master node successfully generates a valid block, its R will be increased by 0.02. However, if the master node fails or is identified as a malicious node, its R will be deducted by 0.2, and it will be immediately removed from the trusted node. (3) The slave node that correctly overthrows the malicious master node will increase its R by 0.02.

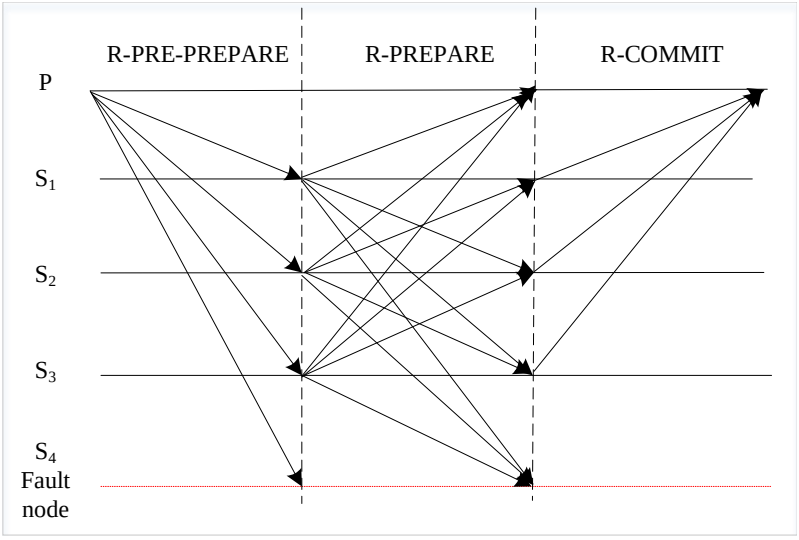

Figure 2. Execution flow of a round of consensus.

The initialization of reputation value is mainly divided into two cases. For the initial nodes of the system, the comprehensive strength of these nodes, such as computing power and network bandwidth, is used as the basis for initializing their reputation values. For the newly added node, other nodes vote to get its initial reputation value according to the comprehensive strength of the new node. Figure 3 shows the dynamic reputation level transformation of nodes.

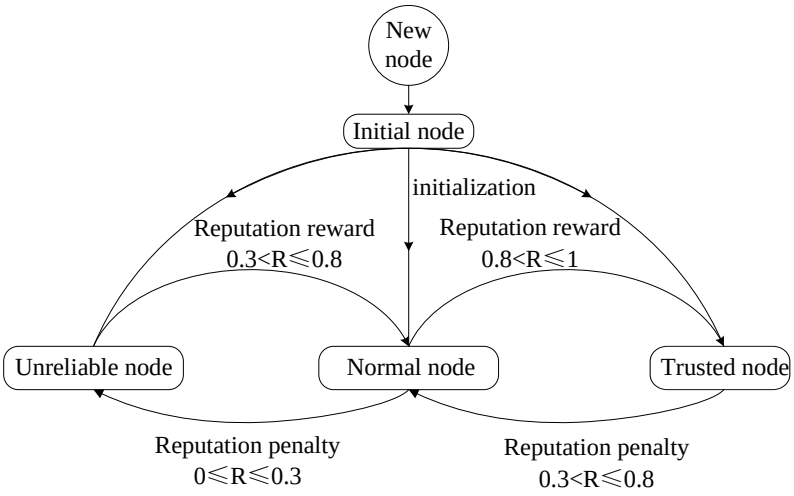

Figure 3. Dynamic reputation level transformation diagram of nodes.

On the one hand, the reputation mechanism can ensure the fairness of R-PBFT. Honest nodes will be rewarded, while malicious nodes will be punished for dishonesty. On the other hand, it can ensure the reliability of R-PBFT and remove malicious nodes in the blockchain network to avoid affecting the security of the system.

3.5. Training of BD-FL

Algorithm 2 describes the training process of the BD-FL. Assuming that the model needs r times of iterative training to reach convergence, the total time complexity of BD-FL is $O(n^2) + O(nm) + O(n+m) + O(1)$.

Algorithm 2 BD-FL training.

Input: local device set K, edge server set S, model demander MQ.
Output: global aggregation model M.

Initialization of K and MQ;
MQ releases task;
for $k \in K$ **do**
  calculate the matching degree with all edge servers;
  bind with a edge server according to the proposed matching algorithm;
end for
while the cost of global model is greater than the set threshold **do**
  for $k \in K$ **do**
    download global model M;
    train M using local data set;
    upload local model parameters, training time, and other information to its bound server after training;
  end for
  for $s \in S$ **do**
    use verification device to verify the authenticity of uploaded data and the accuracy of local model parameters;
    calculate the scores of local devices according to the incentive mechanism;
    if it is a master node **then**
      participate in the consensus process;
      aggregate all local model parameters and update M;
      store the global aggregation information, transaction information, scores, and etc. into blocks and broadcast in the blockchain;
      update the reputation value according to R-PBFT;
    else
      participate in the consensus process or not according to R-PBFT;
      receive M from the master node;
      notify local devices to download M after consensus;
      update the reputation value;
    end if
  end for
end while
return M;

4. Experiments and Results

4.1. Experiment Setting

We built a simulation experiment environment on an Intel(R) Xeon(R) server with two Gold 6248 processors @ 2.50 GHz (Intel Corporation, Santa Clara, CA, USA). We used Java version 1.8 to implement edge servers and the blockchain system, and Python version 3.7 to implement the local device environment. The local device communicates with the edge server through Socket. In the simulation environment, local devices are distributed within the coverage range of a 250 m radius of each edge server. The channel gain between them is modeled as $30.6 + 36.7log_{10}(distNode_{ks})$ dB using the block fading model. Table 1 shows the simulation parameter settings.

Table 1. Simulation parameter settings.

Simulation Parameters	Value
Network Bandwidth	20 MHz
Shooting Power p_k	200 mW
Power Spectral Density	-95 dbm/Hz
Uplink Data Size D_k	[3000, 4000] kb
μ, η	0.5,0.5

We use the ResNet18 implemented by PyTorch as the federated learning network model, and the dataset is CIFAR-10. In experiments, the CIFAR-10 dataset will be randomly divided into multiple copies with different sizes. The local device will randomly obtain one copy as the local data sample.

4.2. Evaluation of BD-FL

Since the proposed preference-based stable matching algorithm plays an important role in BD-FL to minimize the model cost and reduce the system delay, we first conducted a set of experiments to evaluate the stable matching algorithm. We designed three matching algorithms as the baseline, i.e., the local device first greedy algorithm (DFG), the edge server first greedy algorithm (SFG), and the random matching algorithm (RMA), and applied them to BD-FL for comparison experiments.

For DFG, each local device sends a matching request to the edge server according to its preference list. If the candidate list of the edge server does not reach the maximum number of bindings, the device binds with the server directly, otherwise, it continues to send requests to other servers. For SFG, each edge server sends a matching request to the local device according to its preference list. If the local device is unbound, the server binds with the device directly, otherwise, it continues to send requests to other local devices. For RMA, the local device and the edge server randomly send matching requests to each other. If both parties are unbound, they can bind directly, otherwise, they continue to send requests to other unbound devices or servers.

Figure 4 shows the total system costs of the four matching algorithms with different numbers of nodes. As RMA is a pure random matching algorithm without considering any factors that affect the matching result, it has the highest system cost. The costs of DFG and SFG are also higher than that of the stable matching algorithm. Because they only consider the one-side cost function of the local device or the edge server as the minimization objective, they cannot achieve the overall optimization of the system cost. In contrast, the preference-based stable matching algorithm designs the corresponding preference functions of local devices and edge servers by considering various influencing factors, therefore, both parties are able to match and bind efficiently. The stable matching algorithm achieves the minimum system cost, which reduces the cost by 34.9%, 43.6%, and 69.9% compared to DFG, SFG, and RMA, respectively, on average.

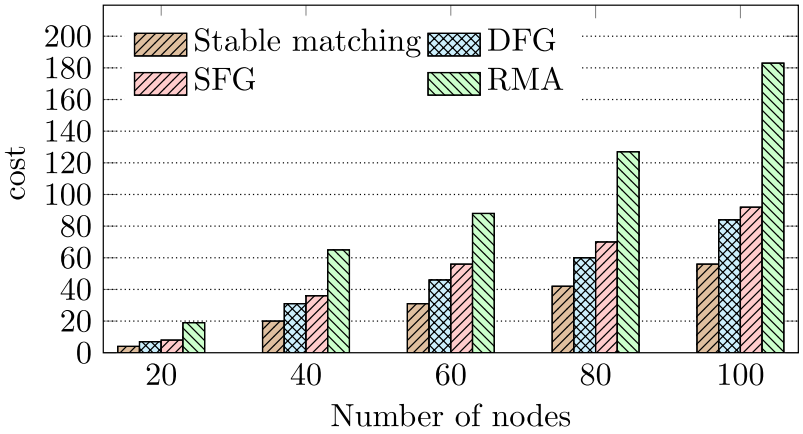

Figure 4. System costs of 4 matching algorithms with different numbers of nodes.

We also counted the number of unstable matches of DFG, SFG, and RMA, which is shown in Figure 5. It can be seen that the number of unstable matches in these algorithms increases with the number of nodes. Due to the randomness of RMA, the number of unstable matches is the largest. DFG and SFG also have multiple unstable matches. The result of unstable matches is consistent with the result of system costs in Figure 4, and the reason is the same. For the stable matching algorithm, the number of unstable matches is always 0, even if the number of nodes in BD-FL increases.

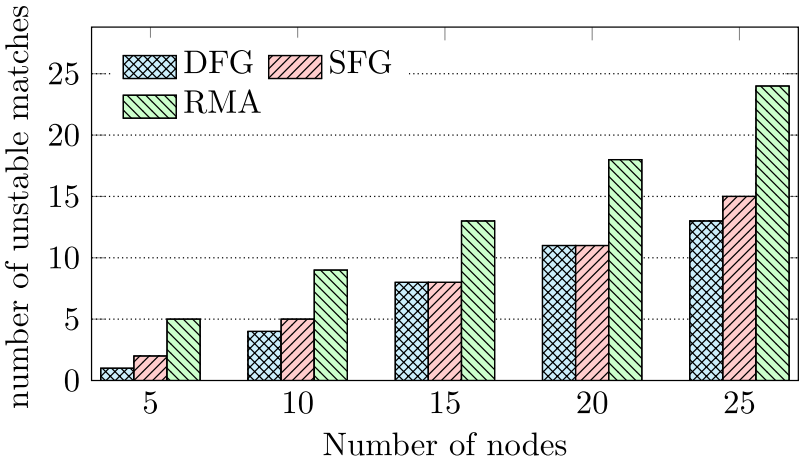

Figure 5. Number of unstable matches of DFG, SFG, and RMA.

To evaluate the model training time of BD-FL with the four matching algorithms, we set the number of edge servers to 4 and the number of local devices to 20 to build the simulation environment. In order to eliminate the influence of the consensus algorithm, all BD-FL models adopt traditional PBFT for blockchain consensus. Figure 6 shows the results of model training time. It can be seen that the training time of BD-FL models with DFG, SFG, and RMA is longer than that of the stable matching algorithm. This is because DFG and SFG only consider the single-side matching on local devices or edge servers, and RMA binds devices to servers in a random way, they cannot achieve the optimal stable matching. According to Figure 5, there are unstable matches in DFG, SFG, and RMA, which will increase the communication delay, and thus lead to a longer total time of model training. On the contrary, BD-FL uses the stable matching algorithm on both sides of the device and the server to utilize the computing resources and minimize the system communication cost. Compared with DFG, SFG, and RMA, the stable matching algorithm reduces the training time of BD-FL by 10.0%, 12.5%, and 19.7%, respectively, for 15 rounds of training. The result of model training is consistent with that of system cost and unstable matches.

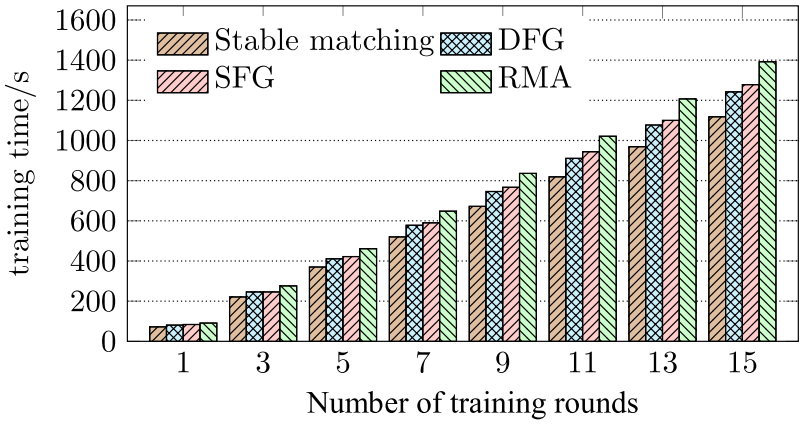

Figure 6. Model training time with different algorithms.

In order to better evaluate the performance of BD-FL, we chose a state-of-the-art blockchain-based federated learning method [17], named FLChain, for comparison. FLChain also applies blockchain techniques to federated learning. However, the nodes in the blockchain are composed of entities registered in FLChain, and all local devices participate in the consensus process of the blockchain.

In the comparison experiments, we set two configurations for BD-FL. In the first configuration, 20 local devices are bound to 4 edge servers in the BD-FL with the stable matching algorithm, denoted as BD-FL1. Additionally, in the second configuration, 20 local devices are bound to 10 edge servers, denoted as BD-FL2. For the network environment of

FLChain, we set 20 local devices and 4 edge servers. The 10 local devices are not bound to edge servers in FlChain for consensus. Figure 7 shows the comparison results of model training time. It can be seen that as the number of global training rounds increases, the training time of all methods grows, but the training time of FLChain grows significantly and is the longest. This is because the more nodes participating in the blockchain consensus, the longer the communication time, and ultimately the longer the model training time. FLChain does not optimize the blockchain consensus process, and all nodes need to participate in consensus, causing a higher communication cost and thus a longer training time. All 20 local devices in FLChain participate in the consensus process, while the numbers of nodes participating in the consensus in BD-FL1 and BD-FL2 are 4 and 10. Therefore, BD-FL1 is the most efficient in the consensus process among these three methods. Experimental results show that BD-FL1 and BD-FL2 reduce the training time by 34.9% and 27.0%, respectively, over FLChain for 150 rounds of global model training.

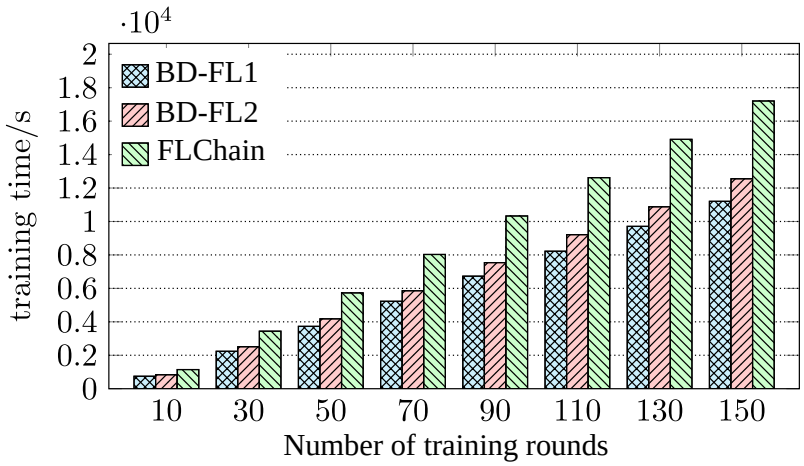

Figure 7. Model training time with different methods.

4.3. Evaluation of R-PBFT

To evaluate the performance of BD-FL with R-PBFT, we compared this method to the BD-FL with PBFT that applies the traditional PBFT algorithm to the consensus process of blockchain. In the experiments, BD-FL uses 20 local devices to bind with 4 edge servers according to the stable matching algorithm for both methods. Figure 8 shows the global model training time of these two methods over different numbers of training rounds. It can be seen that as the number of training rounds increases, the training time of the two methods grows, and the BD-FL with R-PBFT consumes significantly less time than the BD-FL with PBFT. This is due to the fact that R-PBFT streamlines the consistency protocol and eliminates the client nodes to optimize the traditional PBFT for the decentralized federated learning system in the edge computing environment. Thus, R-PBFT can effectively reduce the communication overhead of consensus and improve the global model training efficiency compared to the traditional PBFT. Meanwhile, R-PBFT can better guarantee the security of the system by introducing the reputation mechanism. In the experiments, BD-FL with R-PBFT reduces the training time by 12.2% compared with BD-FL with PBFT for 150 rounds of global model training.

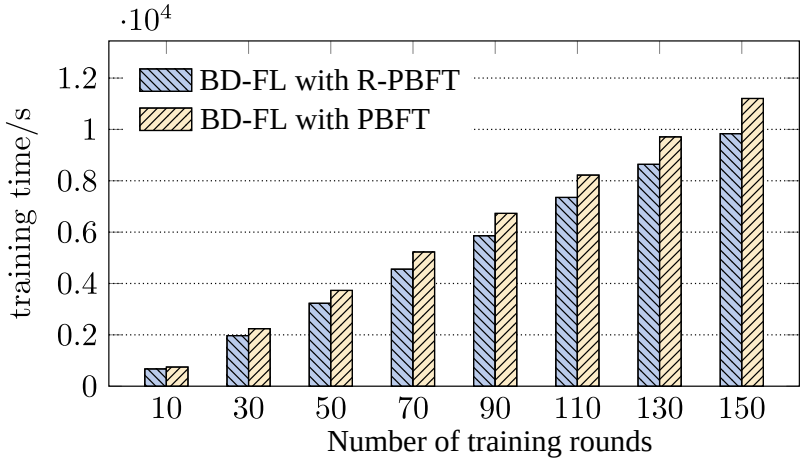

Figure 8. Model training time with different consensus algorithms.

5. Conclusions

In this paper, we mainly proposed a blockchain-based decentralized federated learning method for the edge computing environment. The proposed method joins all edge servers into the blockchain system, and the edge server nodes that obtain bookkeeping rights aggregate the global model to solve the centralization problem of federated learning caused by a single point of failure. In this method, we introduced an incentive mechanism to promote local devices to contribute data samples for model training. To further enhance the system efficiency, we proposed a preference-based stable matching algorithm to bind local devices with appropriate edge servers. For the consensus process of blockchain, we optimized the PBFT algorithm to reduce the communication overhead and enhance the system security, which improves the model training efficiency. Experimental results verified the effectiveness of the proposed method in communication overhead, system delay, and model training efficiency.

In the blockchain consensus process of the proposed method, the information broadcast between edge server nodes is not encrypted, which may lead to the disclosure of local model parameter information of the local models. Avoiding information leakage and improving system security in the consensus process will be one of our future research directions.

Author Contributions: Conceptualization, S.L. and X.W.; formal analysis, S.L.; experiment, S.L. and X.W. and L.H.; writing—original draft preparation, X.W.; writing—review and editing, S.L.; visualization, L.H.; supervision, W.W.; project administration, W.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (No. 62002279 and No. 61972311), Natural Science Basic Research Program of Shaanxi (Program No. 2020JQ-077), and Shandong Provincial Natural Science Foundation (No. ZR2021LZH009).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Vinod, D.; Bharathiraja, N.; Anand, M.; Antonidoss, A. An improved security assurance model for collaborating small material business processes. *Mater. Today Proc.* **2021**, *46*, 4077–4081. [CrossRef]
2. Zhang, Q.; Ding, Q.; Zhu, J.; Li, D. Blockchain empowered reliable federated learning by worker selection: A trustworthy reputation evaluation method. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Nanjing, China, 29 March–1 April 2021; pp. 1–6.

3. Tomovic, S.; Yoshigoe, K.; Maljevic, I.; Radusinovic, I. Software-defined fog network architecture for IoT. *Wirel. Pers. Commun.* **2017**, *92*, 181–196. [CrossRef]
4. Hu, Y.; Niu, D.; Yang, J.; Zhou, S. FDML: A collaborative machine learning framework for distributed features. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 2232–2240.
5. Nakamoto, S.; Bitcoin, A. A peer-to-peer electronic cash system. *Bitcoin* **2008**, *4*, 2. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 1 January 2023)
6. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge computing: Vision and challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
7. Liu, Z.; Cao, Y.; Gao, P.; Hua, X.; Zhang, D.; Jiang, T. Multi-UAV network assisted intelligent edge computing: Challenges and opportunities. *China Commun.* **2022**, *19*, 258–278. [CrossRef]
8. Zhang, X.; Wu, W.; Yang, S.; Wang, X. Falcon: A blockchain-based edge service migration framework in MEC. *Mobile Inf. Syst.* **2020**, *2020*, 8820507. [CrossRef]
9. Guo, F.; Yu, F.R.; Zhang, H.; Ji, H.; Liu, M.; Leung, V.C. Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing. *IEEE Trans. Wirel. Commun.* **2019**, *19*, 1689–1703. [CrossRef]
10. Shahryari, O.K.; Pedram, H.; Khajehvand, V.; TakhtFooladi, M.D. Energy and task completion time trade-off for task offloading in fog-enabled IoT networks. *Pervasive Mob. Comput.* **2021**, *74*, 101395. [CrossRef]
11. Yang, S.; Han, K.; Zheng, Z.; Tang, S.; Wu, F. Towards personalized task matching in mobile crowdsensing via fine-grained user profiling. In Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications, Honolulu, HI, USA, 15–19 April 2018; pp. 2411–2419.
12. Bharathiraja, N.; Padmaja, P.; Rajeshwari, S.; Kallimani, J.S.; Buttar, A.M.; Lingaiah, T.B. Elite Oppositional Farmland Fertility Optimization Based Node Localization Technique for Wireless Networks. In Proceedings of the Wireless Communications and Mobile Computing, Dubrovnik, Croatia, 30 May–3 June 2022; Volume 2022.
13. Vasin, P. Blackcoin's Proof-of-Stake Protocol v2. 2014. Volume 71. Available online: https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf (accessed on 1 January 2023).
14. Liu, D.; Camp, L.J. Proof of Work can Work. In Proceedings of the 5th Annual Workshop on the Economics of Information Security, Robinson College, University of Cambridge, England, UK, 26–28 June 2006. Available online: https://econinfosec.org/archive/weis2006/docs/50.pdf (accessed on 1 January 2023).
15. Yang, F.; Zhou, W.; Wu, Q.; Long, R.; Xiong, N.N.; Zhou, M. Delegated proof of stake with downgrade: A secure and efficient blockchain consensus algorithm with downgrade mechanism. *IEEE Access* **2019**, *7*, 118541–118555. [CrossRef]
16. Castro, M.; Liskov, B. Practical Byzantine Fault Tolerance. *ACM Trans. Comput. Syst. (TOCS)* **2002**, *20*, 398–461 . [CrossRef]
17. Bao, X.; Su, C.; Xiong, Y.; Huang, W.; Hu, Y. FLChain: A blockchain for auditable federated learning with trust and incentive. In Proceedings of the 2019 5th International Conference on Big Data Computing and Communications (BIGCOM), Qingdao, China, 9–11 August 2019; pp. 151–159.
18. Nofer, M.; Gomber, P.; Hinz, O.; Schiereck, D. Blockchain. *Bus. Inf. Syst. Eng.* **2017**, *59*, 183–187. [CrossRef]
19. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X.; Wang, H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352–375. [CrossRef]
20. Kim, H.; Park, J.; Bennis, M.; Kim, S.L. Blockchained on-device federated learning. *IEEE Commun. Lett.* **2019**, *24*, 1279–1283. [CrossRef]
21. Weng, J.; Weng, J.; Zhang, J.; Li, M.; Zhang, Y.; Luo, W. Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. *IEEE Trans. Dependable Secur. Comput.* **2019**, *18*, 2438–2455. [CrossRef]
22. Jia, R.; Dao, D.; Wang, B.; Hubis, F.A.; Hynes, N.; Gürel, N.M.; Li, B.; Zhang, C.; Song, D.; Spanos, C.J. Towards efficient data valuation based on the shapley value. In Proceedings of the The 22nd International Conference on Artificial Intelligence and Statistics, PMLR, Naha, Okinawa, Japan, 16–18 April 2019; pp. 1167–1176.
23. Hu, G.; Jia, Y.; Chen, Z. Multi-user computation offloading with d2d for mobile edge computing. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
24. Wu, D. Research on multi-task multi-device matching algorithm based on machine learning. Master's Thesis, Zhe Jiang University, Hangzhou, China, 2019.
25. Lu, X.; Liao, Y.; Lio, P.; Hui, P. Privacy-preserving asynchronous federated learning mechanism for edge network computing. *IEEE Access* **2020**, *8*, 48970–48981. [CrossRef]
26. Zheng, H.; Guo, W.; Xiong, N. A kernel-based compressive sensing approach for mobile data gathering in wireless sensor network systems. *IEEE Trans. Syst. Man, Cybern. Syst.* **2017**, *48*, 2315–2327. [CrossRef]
27. Ma, S.; Cao, Y.; Xiong, L. Transparent contribution evaluation for secure federated learning on blockchain. In Proceedings of the 2021 IEEE 37th International Conference on Data Engineering Workshops (ICDEW), Chania, Greece, 19–22 April 2021; pp. 88–91.
28. Gao, S.; Yu, T.; Zhu, J.; Cai, W. T-PBFT: An EigenTrust-based practical Byzantine fault tolerance consensus algorithm. *China Commun.* **2019**, *16*, 111–123. [CrossRef]
29. Liu, J.; Li, W.; Karame, G.O.; Asokan, N. Scalable byzantine consensus via hardware-assisted secret sharing. *IEEE Trans. Comput.* **2018**, *68*, 139–151. [CrossRef]

30. Wang, Y.; Song, Z.; Cheng, T. Improvement research of PBFT consensus algorithm based on credit. In *International Conference on Blockchain and Trustworthy Systems*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 47–59.
31. Yu, G.; Wu, B.; Niu, X. Improved blockchain consensus mechanism based on PBFT algorithm. In Proceedings of the 2020 2nd International Conference on Advances in Computer Technology, Information Science and Communications (CTISC), Suzhou, China, 10–12 July 2020; pp. 14–21.
32. Nithya, G.; Engels, R.; Das, H.R.; Jayapratha, G. A Novel-Based Multi-agent Brokering Approach for Job Scheduling in a Cloud Environment. In *Informatics and Communication Technologies for Societal Development*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 71–84.
33. Hochman, H.M.; Rodgers, J.D. Pareto optimal redistribution. *Am. Econ. Rev.* **1969**, *59*, 542–557.