electronics

MDPI

Article

On Dynamic Node Cooperation Strategy Design for Energy Efficiency in Hierarchical Federated Learning

Zhuo Li [1,2,*], Sailan Zou [1,2] and Xin Chen [2]

[1] Beijing Key Laboratory of Internet Culture and Digital Dissemination Research, Beijing Information Science and Technology University, Beijing 100101, China; zousailan@163.com
[2] School of Computer Science, Beijing Information Science and Technology University, Beijing 100101, China; chenxin@bistu.edu.cn
* Correspondence: lizhuo@bistu.edu.cn

Abstract: In Hierarchical Federated Learning (HFL), opportunistic communication provides opportunities for node cooperation. In this work, we optimize the node cooperation strategy using opportunistic communization with the objective to minimize energy cost under the delay constraint. We design an online node cooperation strategy (OSRN) based on the optimal stopping theory. Through theoretical analysis, we prove the NP-hardness of the problem investigated and the competition ratio that can be achieved by OSRN. We conduct thorough simulation experiments and find that the proposed algorithm outperforms the random selection algorithm SNNR with 22.04% reduction in energy cost. It is also observed that the energy cost can be reduced by 20.20% and 13.54%, respectively, compared with the existing methods CFL and THF.

Keywords: Hierarchical Federated Learning (HFL); opportunistic communication; node cooperation

check for updates

Citation: Li, Z.; Zou, S.; Chen, X. On Dynamic Node Cooperation Strategy Design for Energy Efficiency in Hierarchical Federated Learning. *Electronics* **2023**, *12*, 2362. https://doi.org/10.3390/electronics12112362

Academic Editors: Fernando De la Prieta Pintado and Antonio Brogi

Received: 25 March 2023
Revised: 19 May 2023
Accepted: 22 May 2023
Published: 23 May 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the popularization of the Internet of Things (IoT) and mobile computing, billions of mobile and IoT devices are connected to the Internet, generating trillions of gigabytes of data at the network edge [1]. According to IDC's report, 59 zettabytes of data were created in 2020, and the global data volume will reach 175 zettabytes in 2025 [2]. The data provides opportunities for the popularization of intelligent applications. In traditional machine learning, mobile devices must upload local data to cloud servers for centralized model training [3]. The uploading process of these data causes a lot of energy cost and latency. At the same time, privacy-sensitive mobile devices face the risk of privacy data leakage. Given this threat, many mobile device users are reluctant to upload private data to cloud servers [4].

For this problem, Federated Learning (FL) was proposed in 2016 [5], a distributed machine learning approach for training. The traditional federated learning model is based on the Client-Cloud star topology [6]. After obtaining the initial global model from the cloud server, each participating device uses the local dataset for local model training and uploads the model parameters to the cloud server. The cloud server performs a global model update based on the local model parameters of each device. It effectively resolves the conflict between data privacy protection and data sharing requirements of privacy-sensitive devices. However, many communication loops are required between the cloud server and the participating devices to exchange model parameters. In particular, traditional federated learning is not applicable when training and managing data-intensive, latency-sensitive machine learning tasks on large-scale networks. However, edge computing has become a key technology to solve a series of problems caused by the increasing number of interconnected devices and large-scale data transmission [7,8]. Therefore, researchers have combined federated learning with edge computing in the last two years to address

this problem and proposed Hierarchical Federation Learning (HFL) based on Client-Edge-Cloud architecture [9–11].

The HFL framework uses edge servers as mediators for communication between terminal devices and cloud servers. After each terminal device obtains the initial global model from the cloud and edge servers, it repeats the following three processes. First, each terminal device generates local model parameters based on gradient descent and then uploads them to the edge server. Second, each edge server aggregates local model parameters and then uploads edge models to the cloud server. Third, the cloud server performs a weighted average of edge models from each edge server and synchronizes the updated global models to the edge servers and terminal devices. In HFL, the uploading of parameters to edge servers from mobile devices may cause high energy cost, as shown in Figure 1.

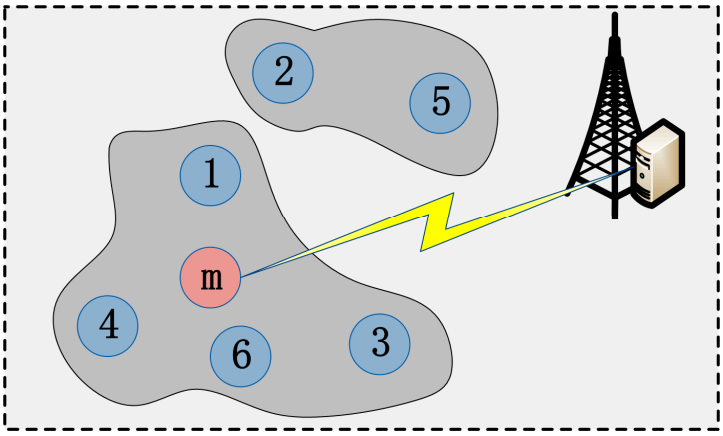

Figure 1. Without cooperation. Nodes upload parameters directly to the edge server.

For this reason, we consider that nodes cooperate to complete the uploading of parameters, as shown in Figure 2. Some nodes need more energy for parameter uploading, while some nodes need less energy for parameter uploading because they may move close to the edge server in some time slot. In this paper, we consider mobile device cooperation through this type of opportunistic communication in HFL. When they meet each other, an online node cooperation algorithm can be used to reduce energy cost for uploading parameters under the latency constraint. For a node and a neighboring node that it meets with, a decision can be made on whether to transmit local model parameters to this neighboring node or not. Once the relay node is selected as a relay, it will be used for cooperative transmission.

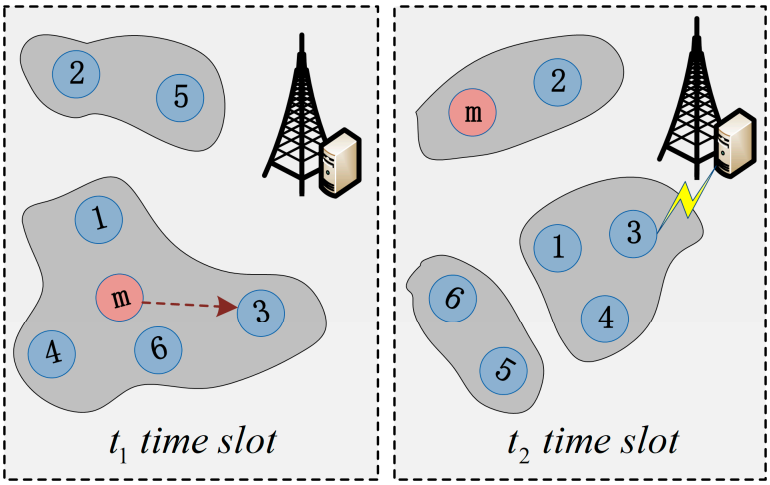

Figure 2. With cooperation. Nodes upload parameters to the edge server with the help of relay node.

The main contributions of this paper can be summarized as follows.

(1) We propose node cooperation for energy cost minimization with the delay constraint in HFL. We formulate the optimization problem and prove its decision problem is NP-hard.
(2) We design an online optimized node cooperation strategy, in which each node training local model can select optimal relay node dynamically. Relay nodes can help transmit the model parameters while minimizing energy cost.
(3) We conduct thorough experiments to evaluate the performance of our proposed strategy. It is found that energy cost can be reduced by 24.49% and 22.04% compared with HierFAVG and SNNR, respectively. Compared with existing CFL and THF, it is also observed that the energy cost is reduced by 20.20% and 13.54%, individually.

The rest of the paper is organized as follows. Section 2 briefly reviews the related work. We introduce the HFL system model and define the energy cost minimization problem with the delay constraint in Section 3. We introduce the design of the online node cooperation strategy in Section 4. We analyze and evaluate the performance of our proposed strategy in Section 5. Finally, we conclude in Section 6.

2. Related Work

With the general concern about data privacy and security, federated learning has become a prominent research topic. The work in [12] shows that cloud-based FL can access millions of mobile devices but causes high communication cost and inefficient model training. Edge-based FL can effectively reduce the communication and computation cost [13], which attracts research interest of both academia and industry.

In the literature on HFL, some studies proposed to increase the local computation of nodes as much as possible to reduce the communication rounds required for model training [14,15]. Reisizadeh et al. [16] considered mobile devices to perform multiple local iterations before uploading parameters. Wang et al. [17] sampled the devices contributing to model training, reducing the communication rounds. Existing work in [18–20] adopted methods such as sparsity to compress model updates of the client, reducing the amount of data uploaded to the edge server in a single communication round. Zhang et al. [21] proposed a distributed stochastic gradient edge learning method based on SDM-DSGD. The model converged with a slight gradient sparsity transmission probability in this method. Zheng et al. [22] proposed a distributed hierarchical depth computation model to compresses model parameters in a high-dimensional space into a set of low-dimensional subspaces. Liu et al. [23] proposed a communication efficient model training algorithm (Local-QSGD) by taking full advantage of cloud and edge servers. This algorithm quantified the weight of model updates in the upload process, which reduced the amount of data. Ren et al. [24] proposed a new data-driven method, namely, cloud-edge based lightweight temporal convolutional networks to reduce the computational time. The research targets of the above works are either to reduce communication rounds for parameter uploading or to reduce the amount of data transmitted in a single communication round by model compression. However, our research focuses on reducing energy cost for uploading parameters under a delay constraint.

There are also studies in HFL for reducing energy cost. Lou et al. [25] proposed a novel hierarchical federated edge learning (HFEL) framework to reduce the energy overhead and training time through edge model aggregation. Wang et al. [26] proposed an efficient edge computing and optimization model, which used a tensor-based multi-attribute matching method to reduce economic cost and energy cost. Yi et al. [27] proposed a novel control strategy for information diffusion, which combined graph neural networks with ordinary differential equation systems to achieve low latency and high energy efficiency. Some studies [28,29] proposed a hierarchical federated learning framework to support Client-Edge-Cloud. It allowed multiple edge servers to reduce energy cost by partial model aggregation. Liu et al. [30] proposed a node cooperation scheme for data exchange based on Device-to-Device (D2D) communication. Jin et al. [31] proposed an iterative optimization

algorithm based on fractional matrix programming and then confirmed the advantages of the proposed D2D cooperation scheme in reducing communication costs. Mustafa et al. [32] used the shortest distance selection criterion in D2D communication. In addition, Wang et al. [28] divided edge nodes into different clusters. All nodes in each cluster only uploaded their parameters to the randomly selected cluster headers. Asad et al. [33] made an efficient 3-way hierarchical framework (THF) to promote communication efficiency. Only the cluster head communicated with the edge server, which reduced the energy cost of parameter uploading due to the short distance from source to destination. However, the above-related work to reduce energy cost rarely studies node selection and node cooperative transmission or considers the type of cooperation between nodes as D2D communication. They do not consider the opportunistic communication between nodes.

Existing work [34,35] optimized the data caching and sharing policies, reducing the cost of memory access in terms of energy cost and delay by assigning data to different storage bodies. Zhao et al. [36] proposed a caching algorithm based on multi-layer federated reinforcement learning (CoCaRL) in vehicular networks, which solved the nuisance of transmission delay and energy overhead. Cheng et al. [37,38] proposed a privacy-preserved caching scheme by a double-layer blockchain architecture, which showed the gain of communication delay and energy cost. Qiao et al. [39] proposed a distributed resources-efficient caching policy to improve the content caching efficiency and reduce the energy cost. This caching policy adopted an adaptive content caching algorithm combined deep reinforcement learning. Khanal et al. [40] investigated in detail the role of proactive caching methods in self-driving cars for improving the caching cost. This method used a self-attention technique with an LSTM-based prediction mechanism. Liu et al. [41] designed a dynamic caching replacement mechanism to enhance the personalized utilization of the cache resources and reduce the system cost. This mechanism addressed the challenges in extending the centralized deep deterministic policy gradient to the distributed manner. The above works are related to reducing energy costs but it is achieved by optimizing the cache management strategy, which is different from our approach to solve energy cost problem. We analyze and compare the existing work, as shown in Table 1.

Table 1. Comparison of the related work.

Literature	Optimization Target	Enhance Local Iteration	Compression Model	Multiple Edge Aggregation	Node Cooperation	Cache Management Optimization	Features
[14–17]	Reduce communication rounds	√					Nodes communicate directly with edge servers
[18–24]	Reduce the amount of data in the communication round		√				Nodes communicate directly with edge servers
[25–29]	Reduce the energy cost of model uploading			√			Nodes communicate directly with edge servers
[30–33]	Reduce the energy cost of model uploading				√		Only consider communication between nodes as D2D
[34–41]	Reduce the energy cost of model uploading					√	Only optimize caching strategy to reduce energy cost
This paper	Reduce the energy cost of model uploading				√		Use opportunistic communication and optimize the process of uploading model to reduce energy cost

"√" indicates the optimization method used in related literature.

In this paper, we propose to select relay nodes with low energy cost to cooperate in parameter uploading based on the opportunistic communication for HFL. When nodes meet each other, they decide whether to select the relay node to upload parameters cooperatively or not.

3. System Model and Problem Definition

In this section, we introduce the system model and formulate the problem.

3.1. System Model of HFL

In HFL framework, there is a set of edge servers $\mathcal{K} = \{k : k = 1, \ldots, K\}$, a set of edge nodes $\mathcal{M} = \{m : m = 1, \ldots, M\}$ and a cloud server $\mathcal{S}$, as shown in Figure 3. There is a set of neighboring nodes $\mathcal{N} = \{N_m : N_m = N_1, \ldots, N_M\}$, where N_m is a set of neighboring nodes of node m, consisting of other nodes in the edge network with which it can meet. A set of edge nodes $\mathcal{C} = \{C_k : C_k = C_1, \ldots, C_K\}$, where C_k is a set of nodes associated with edge server k. We denote $N_m^0 = N_m \cup k$ as the set consisting of neighboring nodes of node m and the edge server to which it belongs. We denote $\mathcal{D}_m = \{(x_j, y_j)\}_{j=1}^{|D_m|}$ as the training dataset, and $|D_m|$ as the total number of training samples, where x_j is the j-th input sample and y_j is the corresponding labeled output of x_j in the federated learning task of node m. w_m denotes parameters related to the training model of x_j and y_j.

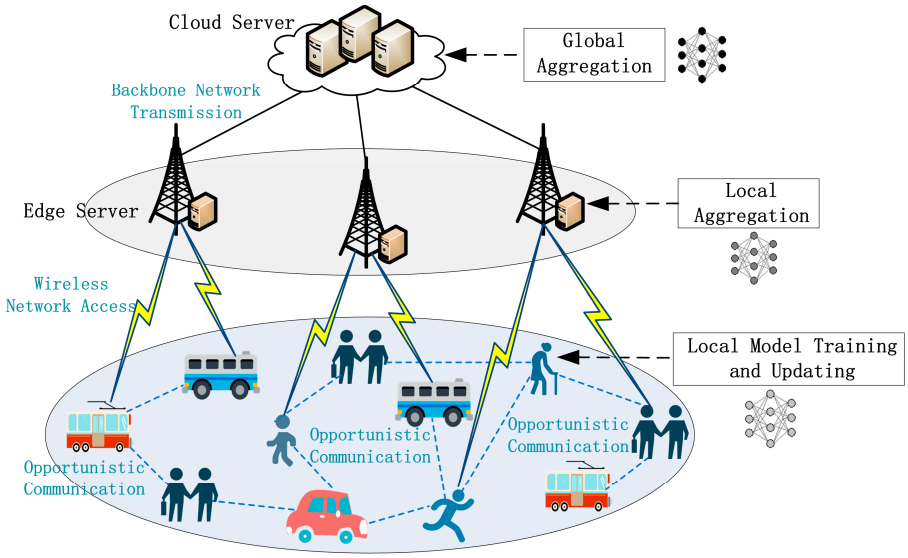

Figure 3. Infrastructure of three-tiered hierarchical federated learning.

3.2. Opportunistic Communication between Nodes

We consider that the communication mode between nodes is opportunistic communication. It is not necessary to have a complete end-to-end communication link between nodes. When two nodes enter each other's wireless communication range and establish a communication connection, we define this moment as the node encounter time. The communication connection is broken when the distance between two nodes exceeds the communication radius. After that, it takes some time for them to meet again. Therefore, we define the encounter interval between nodes as the time interval between this encounter and the subsequent one [42].

The chance of encounter between nodes is generated by node movement. In this study, we assume the encounter interval between nodes $T \sim E(\lambda)$. The probability density function of the exponential distribution is:

$$f(t) = \begin{cases} \lambda e^{-\lambda t}, & t > 0, \\ 0, & t \leq 0, \end{cases}$$

where $\lambda = \frac{1}{E(T)}$ and $E(T)$ is the mathematical expectation of random variable T.

Given node m and neighbor node n, the encounter interval between them is denoted by $ICT_{m,n}$, where $E(ICT_{m,n}) = E(T) = \frac{1}{\lambda_{m,n}}$.

3.3. Computation and Communication Model of HFL

We introduce the computation and communication models in different phrases of HFL.

(1) Edge aggregation. This phase includes local model computation, local model transmission, and edge model aggregation.

Step 1. Local model computation: node m performs local model training on the local dataset D_m. c_m denotes the number of CPU cycles required for m to process a data sample. We assume that each data point has the same size, and the number of CPU cycles for node m to perform one local iteration is $c_m|D_m|$. f_m denotes the CPU frequency assigned by node m to model training, where $f_m \in [0, f_m^{max}]$. m performs local model training to achieve local model accuracy $\alpha \in (0,1)$, which requires $L(\alpha) = \log \frac{1}{\alpha}$ local iterations [43]. Therefore, the computation delay for node m is:

$$t_m^{cmp} = L(\alpha)\frac{c_m|D_m|}{f_m}, \tag{1}$$

and the energy cost is expressed as follows,

$$e_m^{cmp} = L(\alpha)\frac{\alpha_m}{2} f_m^2 c_m \left|D_m\right| \tag{2}$$

where $\frac{\alpha_m}{2}$ is the effective capacitance factor of the calculated chipset of node m.

Step 2. Local model transmission: $I_{mn} \in \{0,1\}$ is used as a binary variable to indicate whether node m and n perform cooperative transmission. When $I_{mn} = 1$, node m selects neighboring node n to participate in the cooperative transmission. When $I_{mn} = 0$, the reverse is true. In the process of node cooperative transmission, we only consider nodes to select at most one relay node. Z denotes the size of traffic caused by model parameters updating. We assume that each model parameter has the same size Z and transmit power is p^s. N_0 denotes the channel noise power at each node. d_{mn}^1 denotes the distance between two nodes. d_{nk}^2 denotes the minimum distance between neighboring node n and edge server k within the remaining delay constraint. λ^* is obtained from the ratio of wave speed and frequency. The estimated minimum transmission delay and energy cost of node m directly transmitting the local model are respectively expressed as

$$t_{mk}^{com} = \frac{Z}{B\log_2(1+\frac{p_k^r}{N_0})}, \tag{3}$$

$$e_{mk}^{com} = \frac{Zp^s}{B\log_2(1+\frac{p_k^r}{N_0})}, \tag{4}$$

where the received power p_k^r is related to the distance from the node to the edge server and $p_k^r = p^s(\frac{\lambda^*}{4\pi d_{mk}^1})^2$.

The estimated minimum transmission delay and energy cost of node m cooperating with the neighboring node n to transmit the local model are respectively denoted as

$$t_{mn}^{com} = \frac{1}{\lambda_{mn}} + \frac{Z}{B\log_2(1+\frac{p_n^r}{N_0})}, \tag{5}$$

$$e_{mn}^{com} = \frac{Zp^s}{B\log_2(1+\frac{p_n^r}{N_0})}, \tag{6}$$

where p_n^r is the received power of node n and $p_n^r = p^s(\frac{\lambda^*}{4\pi d_{mn}^1})^2$.

$$t_{nk}^{com} = \frac{Z}{B\log_2(1+\frac{p_k^r}{N_0})}, \tag{7}$$

$$e_{nk}^{com} = \frac{Zp^s}{B\log_2(1+\frac{p_k^r}{N_0})}, \tag{8}$$

where p_k^r is the received power of edge server k and $p_k^r = p^s(\frac{\lambda^*}{4\pi d_{nk}^2})^2$.

Therefore, the estimated minimum transmission delay and energy cost of node m and n uploading parameters cooperatively are expressed as

$$T_{mk}^{com} = t_{mn}^{com} + t_{nk}^{com}, \tag{9}$$

$$E_{mk}^{com} = e_{mn}^{com} + e_{nk}^{com}, \tag{10}$$

Therefore, the estimated minimum transmission delay and energy cost of node m uploading parameters are denoted as

$$T_{mk}^{tran} = \begin{cases} t_{mk}^{com}, & I_{mn} = 0, \\ T_{mk}^{com}, & I_{mn} = 1,' \end{cases} \tag{11}$$

$$E_{mk}^{tran} = \begin{cases} e_{mk}^{com}, & I_{mn} = 0, \\ E_{mk}^{com}, & I_{mn} = 1,' \end{cases} \tag{12}$$

Step 3. Edge model aggregation: we denote T_k and E_k as the total delay and total energy cost for edge server k to complete a round of edge model aggregation, which can be expressed as

$$T_k = \max_{m \in C_k} \left\{ t_m^{cmp} + T_{mk}^{tran} \right\}, \tag{13}$$

$$E_k = \sum_{m \in C_k} (e_m^{cmp} + E_{mk}^{tran}). \tag{14}$$

(2) Cloud aggregation. After all edge servers complete edge model aggregation, they upload edge model to the cloud server in a synchronous manner for global model aggregation. The delay of completing edge aggregation and uploading edge model is different for different edge servers. We assume that t_{kc} and e_{kc} denote delay and energy cost of edge server k uploading the edge model to the cloud server. Therefore, for the cloud server, the delay T^c of completing a round of global iteration is determined by the edge server with the largest edge aggregation delay T_k and model upload delay t_{kc}. In addition, the energy cost E^c for completing a round of global iteration is the sum of the energy cost E_k for completing the edge aggregation and the energy cost e_{kc} for uploading the edge model among all edge servers. Thus, the system-wide delay and energy cost under one round of global iteration are respectively expressed as

$$T^c = \max_{k \in \mathcal{K}} \{T_k + t_{kc}\}, \tag{15}$$

$$E^c = \sum_{k \in \mathcal{K}} (E_k + e_{kc}). \tag{16}$$

3.4. Problem Definition

According to the above system model, if we want to minimize total energy cost for model training in a round of global iteration, the sum of energy cost of all nodes uploading local model parameters to the edge server should be minimized. We formulate the ECM (Energy Cost Minimization) problem for cooperation in HFL as follows.

$$\min \sum_{k \in \mathcal{K}} \sum_{m \in C_k} E_{mk}^{tran}, \tag{17}$$

$$s.t. \sum_{n \in N_m^0} I_{mn} = 1, \ \forall m \in C_k, \tag{18}$$

$$\sum_{k=1}^{K} C_k = M, \ \forall k \in \mathcal{K}, \tag{19}$$

$$T_{mk}^{tran} \leq T^{max}, \ \forall m \in C_k, \tag{20}$$

$$I_{mn} \in \{0,1\}, \ \forall m \in C_k, \ \forall n \in N_m^0. \tag{21}$$

where constraint (18) indicates that node m selects at most one neighboring node to cooperate in uploading parameters. Constraint (19) describes the association between the mobile node and the edge server, that is, there can be multiple mobile nodes under each edge server, but each node belongs to only one edge server. Constraint (20) denotes that the transmission delay of each node m to upload parameters to the edge server cannot exceed the maximum tolerance time. Constraint (21) describes whether node m cooperates with node n to upload parameters or not.

For the hardness of the ECM problem, we have the following results as shown in Theorem 1, i.e., its corresponding decision problem can be proved NP-hard.

Theorem 1. *The decision problem of ECM in this paper is NP-hard.*

Proof of Theorem 1. The decision problem of ECM is defined as follows. In polynomial time, if each node can find a neighboring node with low energy cost to upload parameters cooperatively, the delay of completing parameters uploading cannot exceed the maximum transmission delay and each node can select at most one neighbor node to cooperate.

The decision problem of the Quadratic Multiple Knapsack Problem (QMKP) has been proven to be NP-hard [44]. An instance of the decision problem for the QMKP is as follows. A set of items is given as $\mathcal{M} = \{1, 2, \ldots, m, \ldots, M\}$, and a set of knapsacks is given as $\mathcal{N} = \{1, 2, \ldots, i, \ldots, N\}$. Each item m has a price p_m and a weight w_m. When any two items m and n are loaded into the same knapsack, a joint price p_{mn} is generated. For each knapsack i, it has the capacity c_i. The purpose of QMKP is to minimize the total weight of the selected items.

For each instance of the decision problem of QMKP, we can reduce it to an instance of the decision problem of ECM in polynomial time as follows. We assume a set of M nodes and a set of N cooperative combinations. The energy cost for node m to upload parameters is p_m. The delay is w_m. Any two nodes m and n form a cooperative combination to cooperate in uploading parameters, and the energy cost is p_{mn}. Each cooperative combination i has its delay upper limit C_i.

In this way, each instance of the decision problem of QMKP can be reduced to an instance of the decision problem of ECM in polynomial time. That is, the problem of minimizing the total weight of the selected items can be reduced to the problem of minimizing the energy cost when uploading the model parameters. Therefore, the decision problem of ECM is NP-hard. □

4. Design of Online Node Cooperation Strategy

We convert the ECM problem into the optimal relay node selection problem for node cooperation and propose an online node cooperation algorithm. Each node with local model updating continuously detects the energy cost and delay of neighboring nodes, selecting proper ones for the help of uploading parameters and achieving the optimization target of energy cost minimization with delay constraint.

4.1. Optimal Relay Node Selection for Node Cooperation

Each neighboring node calculates the delay and energy cost for uploading parameters cooperatively to determine the optimal value of the combined cost and desired payoff. Each node needs to determine the appropriate time slot to upload the parameters to the optimal relay node based on the expected payoff.

Given the delay constraint T of system model training in one round of global iteration, the delay constraint of nodes cooperating to upload parameters in one round of global iteration can be expressed as

$$T^{max} = T - \max_{k \in \mathcal{K}} t_{kc} - \max_{m \in \mathcal{M}} t_m^{cmp}. \tag{22}$$

The optimal relay node selection must minimize energy cost E_{mk}^{tran} under the constraint of delay T^{max}. Therefore, we define the comprehensive cost Γ_{mk} of node m directly uploading parameters as

$$\Gamma_{mk}^{com} = \sigma e_{mk}^{com} + \beta t_{mk}^{com}, \tag{23}$$

where $\sigma \geq 0$ and $\beta \geq 0$ is the weight coefficient of energy cost and delay.

After completing local training, each node performs the selection of the optimal relay node and stopping time slot. The duration for node m to wait for relay node n is $C\tau$, where C is the number of waiting time slots and $C\tau < T^{max}$. Because the delay for each node to complete parameters uploading cannot exceed the maximum transmission delay. When node m and node n cooperate for uploading parameters, the optimization value X_c of relay node n for comprehensive cost is expressed as

$$\begin{aligned} X_c &= \Gamma_{mk}^{com} - (\Gamma_{mn}^{com} + \Gamma_{nk}^{com}) \\ &= \sigma e_{mk}^{com} + \beta t_{mk}^{com} - (\sigma E_{mk}^{com} + \beta T_{mk}^{com}) \\ &= \sigma(e_{mk}^{com} - e_{mn}^{com} - e_{nk}^{com}) + \beta(t_{mk}^{com} - t_{mn}^{com} - t_{nk}^{com}) \end{aligned}. \tag{24}$$

Node m detects at least a one-time slot. We define the set of time slots for node m to stop detecting as $\mathcal{L} = \{C : C \geq 1\}$, where C denotes the time when node m stops detecting. Node m selects the relay node with the optimized value X_C to upload parameters at the stopping time slot C. The obtained desired payoff Y_C is denoted as

$$Y_C = X_C - C\tau. \tag{25}$$

We select the optimal relay node to maximize the desired payoff Y_C, as follows,

$$\max E[Y_C] = \max E[\sigma(e_{mk}^{com} - e_{mn}^{com} - e_{nk}^{com}) + \beta(t_{mk}^{com} - t_{mn}^{com} - t_{nk}^{com}) - C\tau]. \tag{26}$$

Therefore, the goal of maximizing the desired payoff is to find the stopping time C^* and the corresponding optimal relay node n that minimizes energy cost while satisfying transmission delay constraint. Since the positions of nodes change dynamically and the neighboring nodes cannot be determined in advance, the optimal stopping theory can be used to solve the problem. Based on this theory, we can seek superiority after the first 37% of neighboring nodes and find the optimal decision [45].

To maximize the desired payoff (i.e., Equation (26)), we prove the existence of the optimal stopping rule and the optimal solution in Theorem 2 and Theorem 3, respectively.

Theorem 2. *The problem of maximizing desired payoff has an optimal stopping rule.*

Proof of Theorem 2. First, we introduce the optimal stopping rule, which is defined as follows.

For a sequence of random variables $X_1, X_2 \ldots$, assume that the return function of the sequence is: $y_0, y_1(x_1), y_2(x_1, x_2), \ldots, y_\infty(x_1, x_2, \ldots)$. After observing that $X_1 = x_1, X_2 = x_2, \ldots, X_c = x_c$, we can choose to stop the observation and obtain gain $y_c(x_1, x_2, \ldots, x_c)$ or continue to observe x_{c+1}. The stopping rule is to choose a stopping time C^* such that the expected return $E[Y_{C^*}]$ is maximized.

According to the literature [46], if it satisfies:

(1) $E[\sup_c Y_c] < \infty$,
(2) $\limsup_{c\to\infty} Y_c \leq Y_\infty$.

then the optimal stopping rule exists.

For mobile node m, the expected payoff Y_c obtained by stopping the detection at time slot c is

$$Y_c = \sigma(e_{mk}^{com} - e_{mn}^{com} - e_{nk}^{com}) + \beta(t_{mk}^{com} - t_{mn}^{com} - t_{nk}^{com}) - c\tau$$

When node m directly uploads parameters, $t_{mk}^{com} < \infty$, $e_{mk}^{com} < \infty$. When node m cooperates to upload parameters through the neighboring node n, $t_{mn}^{com} < \infty$, $e_{mn}^{com} < \infty$, $t_{nk}^{com} < \infty$,$e_{nk}^{com} < \infty$. Therefore, $Y_c < \infty$. Therefore, $E[\sup_c Y_c] < \infty$ and then condition (1) is satisfied. When $c \to \infty$, there exists $-c\tau \to -\infty$. Therefore, $Y_c \to -\infty$. Obviously, $Y_\infty = -\infty$. Therefore, $\limsup_{c\to\infty} Y_c \leq -\infty = Y_\infty$ and then condition (2) is satisfied. Therefore, the problem of maximizing desired payoff has an optimal stop rule. □

Theorem 3. *The problem of maximizing desired payoff has an optimal solution.*

Proof of Theorem 3. When the node obtains the optimal expected reward $V^* = sup_{C\in\mathcal{L}} E[Y_C]$, the stop time C is the optimal solution of the problem of maximizing desired payoff. Therefore, the optimal stop time slot C^* is expressed as

$$C^* = \min\{C \geq 1 : Y_C \geq V^*\}$$

□

In the HFL framework, each node selects the optimal relay node at the optimal stopping time to cooperate in completing parameter upload. Thus, the energy cost of uploading parameters from the node to the edge server under the delay constraint is minimized.

In the HFL framework, if nodes upload model parameters directly, they face a high energy cost problem. However, each node can reduce energy cost by cooperatively uploading parameters with the help of relay nodes. We propose an energy-efficient online node cooperation algorithm (OSRN), as shown in Algorithm 1. The purpose of this algorithm is that after each node completes local model training, we choose the optimal stopping time slot and the optimal relay node for nodes to cooperate for uploading parameters. Therefore, the energy cost of all nodes for uploading parameters is reduced.

In Algorithm 1, the online node cooperation strategy comprises two parts: the observation phase and the decision phase. The encounter relationship between nodes is obtained based on the distribution followed by the encounter interval between nodes. Moreover, historical movement trajectories and typical deep learning methods are used to predict the shortest distance between the meeting nodes and edge server within the remaining delay constraint. In the observation phase, nodes determine the optimal expected payoff among the expected payoffs of all neighboring nodes. The optimal expected reward is then used as a threshold for accepting or rejecting neighboring nodes in the detection phase.

In the observation phase (Algorithm 1, lines 9–12), we obtain the expected payoff Y_c when the first C_m^* relay nodes of each node m cooperate in uploading model parameters,

respectively. The value with the largest expected payoff is chosen as the optimal expected payoff V_m^*. In the decision phase (Algorithm 1, lines 13–21), we observe the expected payoffs obtained by the relay nodes cooperating on uploading parameters under delay constraints. If the expected payoff obtained by this relay node is greater than the optimal desired payoff V_m^*

Algorithm 1 Online node cooperation algorithm (OSRN)

Input: maximum delay constraint T^{max}, number of nodes M number of edge servers K, initial optimal expected payoff $\{V_{m\in\mathcal{M}}^* = 0\}$ of all nodes
Output: the optimal relay node n and optimal stopping time slot c chosen by each node m, the energy cost e^{total} of all nodes to upload parameters successfully
1 : Compute C_m^* using equation $C_m^* = \frac{N_m}{e}$;
2 : for each time slot $c = 1, \ldots, T^{max}$ do
3 : for each $m = 1, \ldots, M$ do
4 : calculate the probability p_n of its encounter with neighboring node n;
5 : if $p_n \geq \text{rand}(0, 1)$ for neighboring n then
6 : node m meets node n in the current time slot t;
7 : use Equation (5) to calculate the delay t_{mn}^{com} and use the deep learning method to obtain the minimum distance d_{nk}^2 under remaining delay constraint;
8 : calculate the maximum value X_c of all encounter nodes of each node m using Equation (24). Record the node n^* with the largest optimization value;
9 : if the number of relay nodes currently encountered by node m is smaller or equal to C_m^* then
10 : use Equation (25) to calculate the expected payoff Y_c obtained by n^*;
11 : set $V_m^* = Y_c$ and reject the node n^* when $Y_c > V_m^*$;
12 : else
13 : use Equation (9) to calculate total delay T_{mk}^{com} for node m to upload parameters through n^*;
14 : if $T_{mk}^{com} < T^{max}$ then
15 : calculate expected payoff Y_c obtained by relay using Equation (25);
16 : m selects n^* as optimal relay node, and return n^* and current time slot c when $Y_c > V_m^*$, Otherwise, node m waits for the next time slot;
17 : else
18 : use Equations (3), (4), (9), (10), (23) to obtain combined cost Γ_{mk} of uploading parameters directly and combined cost Γ'_{mk} of uploading parameters through n^* in current time slot c;
19 : m selects n^* as optimal relay node, and return n^* and current time slot c when $\Gamma_{mk} > \Gamma'_{mk}$. Otherwise, m upload parameters directly, and return m and current time slot c;
20 : end if
21 : end if
22 : end for
23 : end for
24 : compute energy cost e^{total} of all nodes uploading parameters;
25 : return e^{total}

In addition, we consider that each node encounters at most one neighboring node at the current time slot. If a node encounters more than one neighbor node simultaneously, we observe the node with the optimized value of the maximum integrated cost. In the actual implementation, the minimum distance of each encounter node to the edge server within the remaining time constraint can be predicted based on the historical movement trajectory of the node.

4.2. Cooperative Hierarchical Federated Learning Algorithm

After each node completes local model training, we consider cooperation among nodes for parameter uploading. We provide a round of global aggregation processes for cooperative hierarchical federation learning in Algorithm 2.

All nodes use gradient descent iteration for local model updates. The loss function of node m for local model training on the dataset D_m is expressed as

$$F_m(w) = \frac{1}{|D_m|} \sum_{j=1}^{|D_m|} f_m(x_j, y_j, w), \tag{27}$$

Thus, the local model of node m at the e-th local iteration is expressed as

$$w_m^e = w_m^{e-1} - \eta \bigtriangledown F_m(w_m^{e-1}), \tag{28}$$

until satisfies $\| \bigtriangledown F_m(w_m^e) \| \leqslant \alpha \| \bigtriangledown F_m(w_m^{e-1}) \|$, where η is the learning rate.

After m completes local model training, it transmits parameters to the selected optimal relay node n. Node n is derived from the later online node cooperation algorithm (i.e., Algorithm 1). Node n uploads parameters of node m to the edge server. Edge server k performs a weighted average of the received parameters to obtain the edge model,

$$w_k = \frac{\sum\limits_{m \in C_k} |D_m| w_m}{\sum\limits_{m \in C_k} |D_m|}, \tag{29}$$

$|D|$ denotes the total data, where $|D| = |D_m|_{m=1}^{M}$. The global aggregation is as follows,

$$w = \frac{\sum\limits_{k \in \mathcal{K}} |D_m|_{m=1}^{|C_k|} w_k}{|D|}. \tag{30}$$

Algorithm 2 Cooperative hierarchical federated learning

Input: number of nodes M, number of edge servers K, local accuracy α, initial model parameters $\{w_{m \in \mathcal{M}}^0\}$ of all nodes
Output: the global model w
1 : Edge aggregation:
2 : for $e = 1, 2, \ldots, L(\alpha)$ do
3 : for each $m = 1, \ldots, M$ in parallel do
4 : m obtains local model w_m^e using Equation (28);
5 : end for
6 : if $e \% L(\alpha) = 0$ then
7 : use Algorithm 1 to obtain the optimal relay node n for every node m;
8 : for each $m = 1, \ldots, M$ do
9 : node m sends parameters to optimal relay node n and then n uploads parameters to the edge server;
10 : end for
11 : for each edge server $k = 1, \ldots, K$ do
12 : k receives local model $\{w_{m \in C_k}\}$ and obtains edge model w_k using Equation (29);
13 : end for
14 : end if
15 : end for
16 : Cloud aggregation:
17 : The cloud server receives edge model $\{w_{k \in \mathcal{K}}\}$ and uses Equation (30) to obtain the global model w;

5. Performance Evaluation

In this section, we perform the theoretical analysis of OSRN algorithm, proving its time complexity and competition ratio. We also evaluate and analyze the performance of OSRN through extensive experiments.

5.1. Theoretical Analysis

Theorem 4. *The time complexity of the OSRN algorithm is* $O(T^{max} \times M)$.

Proof of Theorem 4. In OSRN, the process of selecting the optimal stopping time slot and optimal relay node for each node is constrained by the maximum delay. Nested loops achieve this process. The outer loop is looped at most T^{max} times and the inner loop is looped at most M times. Therefore, the time complexity of OSRN is $O(T^{max} \times M)$. □

Theorem 5. *For relay node selection, the competition ratio of OSRN is* $\frac{1}{T^{max}-C_m^*}$.

Proof of Theorem 5. We denote the optimal solution as OPT and the solution obtained through OSRN as ALG. For node m, the worst case occurs when the optimal relay node with optimal expected payoff appears before the index C_m^*. In this case, m continuously examines the encounter nodes within the remaining time slot $C_m^* + 1$ to T^{max}. The probability of a relay node being selected within this index is $[1/(T^{max} - C_m^*)]$. Therefore, the competition ratio of the OSRN algorithm is $Com_r = \frac{ALG}{OPT} = \frac{1}{T^{max}-C_m^*}$. □

Theorem 6. *For the energy cost minimization problem, the competitive ratio of OSRN is* $\frac{\sum_{k\in\mathcal{K}}\sum_{m\in C_k}[(e_{mn^*}^{com}+e_{n^*k}^{com})\frac{1}{T^{max}-C_m^*}]}{\sum_{k\in\mathcal{K}}\sum_{m\in C_k}(e_{mn}^{com}+e_{nk}^{com})}$.

Proof of Theorem 6. In the worst case, the optimal relay node n of each node m is located in the first C_m^* neighboring nodes. When the nodes use the optimal algorithm, the sum of the energy cost to complete parameter upload through the optimal relay node n is $\sum_{k\in\mathcal{K}}\sum_{m\in C_k}(e_{mn}^{com} + e_{nk}^{com})$. From Theorem 5, the probability of node m choosing a relay node n^* in the worst case is $[1/(T^{max} - C_m^*)]$. Then, the sum of energy cost to complete parameter uploading through n^* is $\sum_{k\in\mathcal{K}}\sum_{m\in C_k}[(e_{mn^*}^{com} + e_{n^*k}^{com})\frac{1}{T^{max}-C_m^*}]$ when OSRN is used. Therefore, for the optimization problem, the competitive ratio of OSRN is $Com_r = \frac{ALG}{OPT} = \frac{\sum_{k\in\mathcal{K}}\sum_{m\in C_k}[(e_{mn^*}^{com}+e_{n^*k}^{com})\frac{1}{T^{max}-C_m^*}]}{\sum_{k\in\mathcal{K}}\sum_{m\in C_k}(e_{mn}^{com}+e_{nk}^{com})}$. □

Theorem 7. *For the energy cost minimization problem, when the best solution achieves the best trade-off between energy cost and delay, the ratio between OSRN and the best solution is* $\frac{\sum_{m\in\mathcal{M}} E_m^{fin}}{\sum_{m\in\mathcal{M}} E_m^*}$.

Proof of Theorem 7. In the worst case, node m has not found the optimal relay node within the maximum delay constraint. Then, node m either directly uploads parameters in the last slot, or uploads parameters through neighboring node n. We assume that E_m^1 and E_m^2 are the energy costs of node m to upload parameters under these two conditions. When $E_m^1 < E_m^2$, the final energy cost E_m^{fin} for uploading parameters is E_m^1; otherwise, $E_m^{fin} = E_m^2$. In the best solution where the energy cost and delay are optimally balanced, the first neighboring node that node m encounters after completing the detection phase is the optimal relay node. We assume that the energy cost of node m to upload parameters through optimal relay

node n^* is E_m^*. Therefore, $E_m^* < E_m^1$, $E_m^* < E_m^2$, $E_m^* < E_m^{fin}$. Therefore, for the optimization problem, the ratio between OSRN and the best solution is $\frac{\sum_{k\in\mathcal{K}}\sum_{m\in C_k} E_m^{fin}}{\sum_{k\in\mathcal{K}}\sum_{m\in C_k} E_m^*} = \frac{\sum_{m\in\mathcal{M}} E_m^{fin}}{\sum_{m\in\mathcal{M}} E_m^*}$. □

5.2. Simulation Results and Analysis

We evaluate the performance of OSRN through simulations. For comparison, we also implement the existing algorithms CFL [34] and THF [35] in HFL, as well as uploading parameters using HierFAVG and uploading parameters using SNNR.

5.2.1. Simulation Environment

We implement the simulations based on Python. We consider a scenario with 50 nodes and 5 edge servers deployed. They are distributed randomly in a 100 m $\times$ 100 m area. The channel bandwidth $B = 10$ MHz, and the noise power $N_0 = 10^{-10}$ W. The transmit power $p^s = 0.1$ W, frequency is set as 2.4 GHz, data size $Z = 5$ MB, and opportunistic communication is used for data transmission between nodes. The key parameter settings for communication in the simulation experiment are shown in Table 2.

Table 2. Key parameter settings for communication.

Parameter	Description	Values
K	The number of edge servers	5
p^s	Transmit power	0.1 W
N_0	Noise power	10^{-10} W
B	Channel bandwidth	10 MHz
Z	Model parameters size	5 MB
f	Frequency size	2.4 GHz
$\mathcal{G}$	Network range	100 m $\times$ 100 m
r	Communication radius	40 m

In addition, nodes are moved using a random movement model. In the random walk model, the mobile node moves in the simulation area of 100 m $\times$ 100 m. The mobile node randomly selects a direction and speed to move from the current position to a new position. The new forward direction, forward distance, and speed are selected from the predefined ranges [–1, 1], [0, 3], and [0, 0.3], respectively. Each move of the mobile node will be carried out at the selected forward distance, and after completion, the new forward direction, forward distance, and speed will be calculated. If the mobile node of this model reaches the simulation boundary, it will bounce back from the simulation boundary.

5.2.2. Experimental Results Analysis

In the experiments, average energy cost was used as an indicator to measure the performance of our proposed online node cooperation strategy. Average energy cost refers to the average amount of the total energy consumed by all nodes when completing the model parameters upload. Moreover, it can be calculated based on the ratio of the total energy cost e^{total} obtained from algorithm 1 (i.e., OSRN algorithm) on page 12 to the total number of nodes.

In OSRN, the values of energy cost weight coefficient σ and delay weight coefficient β have an impact on the energy cost of uploading parameters. When the delay weight coefficient β is 1, the impact of energy cost weight coefficient σ on average energy cost is shown in Figure 4. When the value of σ is small, the average energy cost is high, and when the value of σ is large, the average energy cost is low. This is because, in the comprehensive cost caused by the node for uploading parameters, the quantity level of energy cost value is far less than the quantity level of delay value. The greater the value of energy cost weight coefficient σ, the greater the proportion of energy cost in comprehensive cost. That is, it is

more important to reduce the energy cost of nodes uploading parameters to edge servers. Therefore, the appropriate value of σ is 10^4.

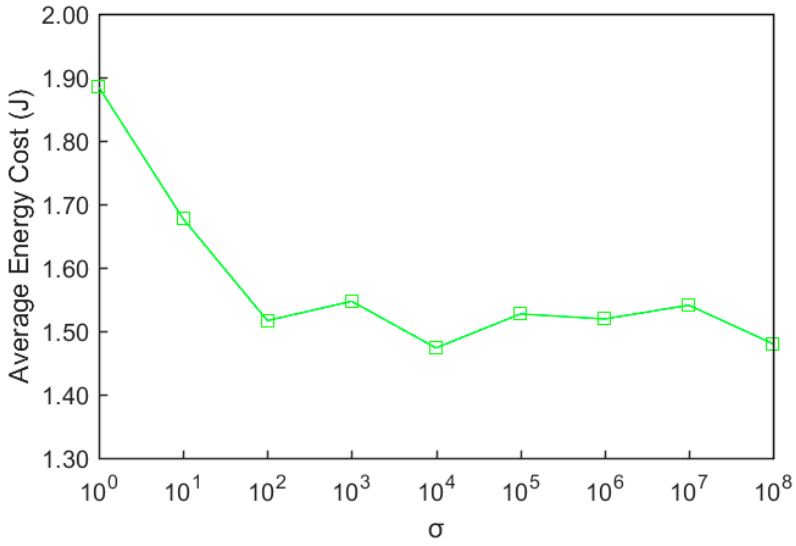

Figure 4. Impact of σ on average energy cost.

We compare the performance of OSRN with HierFAVG, as well as SNNR. We consider different deployment strategies for edge servers, e.g., in four corners and the center, only in the center and randomly deployed. In HierFAVG, all nodes upload local model parameters directly to the edge server. It is found in Figure 5a that with the change of rounds, the energy cost of OSRN is lower than that of HierFAVG and is reduced by 19.03% on average. It is observed in Figure 5b that the energy cost of OSRN is lower than that of HierFAVG and is reduced by 28.28% on average. It is shown in Figure 5c that the energy cost of OSRN is lower than that of HierFAVG and is reduced by 24.49% on average. This is because after waiting for some time, nodes will encounter a relay node with lower transmission energy cost than its own. It is shown in Figure 5a,b that the average energy cost of OSRN is reduced more, and the value is smaller when edge servers are deployed in the center. This is because, in this scenario, some nodes are closer to edge servers than in the scenario where the edge servers are located at four corners and the random position. It is also found in Figure 5b,c that the average energy cost of OSRN is reduced more. Thus, OSRN can perform better when edge servers are deployed in the center.

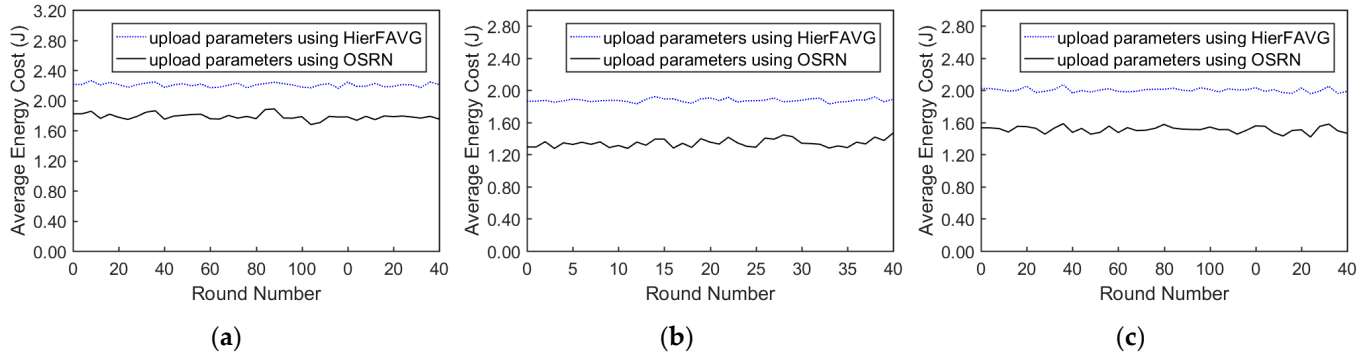

Figure 5. Energy cost comparison with HierFAVG under different deployment locations of edge servers. (**a**) Four Corners and the Center; (**b**) Center; (**c**) Random Deployment.

We compare the performance of OSRN with SNNR. In SNNR, all nodes randomly select a node from the neighboring nodes as a relay node and collaboratively upload parameters to the edge server with the help of this relay node. It is found in Figure 6a

that the energy cost of OSRN is lower than that of SNNR, with an average reduction of 16.79%. It is shown in Figure 6b that the average energy cost of OSRN is lower than that of SNNR, and it is reduced by 24.34% on average. It is observed in Figure 6c that the energy cost of OSRN is lower than that of SNNR, with an average reduction of 22.04%. This is because randomly selected neighboring nodes in SNNR may be far away from the edge server. Moreover, the OSRN continuously detects all neighboring nodes within the maximum delay. In subsequent time slots, nodes encounter optimal relay nodes with low transmission energy cost. Moreover, it is also found in Figure 6a–c that OSRN can perform better when the edge servers' locations are deployed in the center.

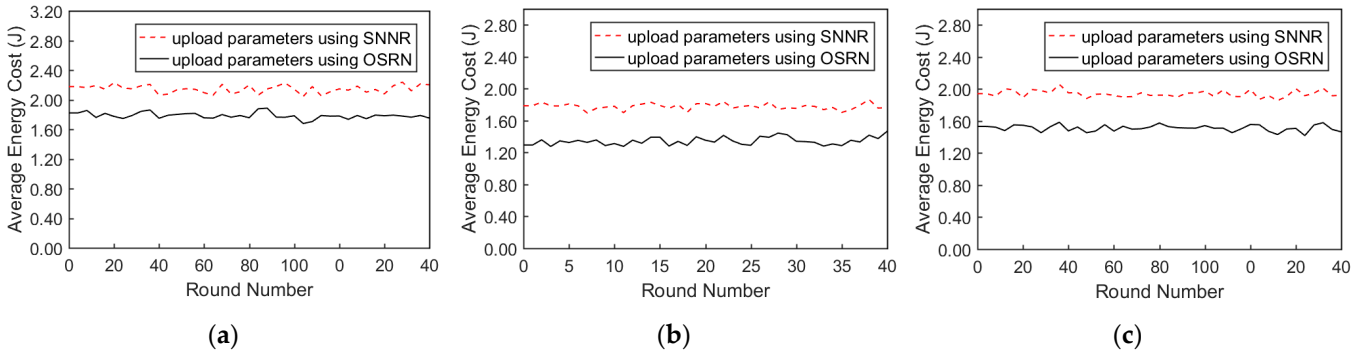

Figure 6. Energy cost comparison with SNNR under different deployment locations of edge servers. (**a**) Four Corners and the Center; (**b**) Center; (**c**) Random Deployment.

We also implement the existing method CFL and THF. As shown in Figure 7a–c, the average energy cost of OSRN is lower than that of CFL, with an average reduction of 15.59%, 26.52%, and 20.20%, respectively. The core idea of CFL is to optimize the process of uploading model parameters through clustering, and the cluster head serves as an intermediary for cooperative communication between nodes and edge servers. In CFL, all nodes are divided into clusters by balanced clustering, and one node is randomly selected as the head node of each cluster. For each cluster, each node first transmits local model parameters to the cluster head, which then uploads the received parameters to its associated edge server. Therefore, each node uploads parameters to the edge server through the head node. However, this randomly selected head node may be far from the edge server, thus resulting in a large amount of transmission energy cost. In contrast, in OSRN proposed in this paper, we consider the shortest distance that all neighboring nodes of each node can reach the edge server in the remaining time, which reduces the energy cost of uploading parameter. Therefore, the energy cost of OSRN is significantly lower than that of CFL.

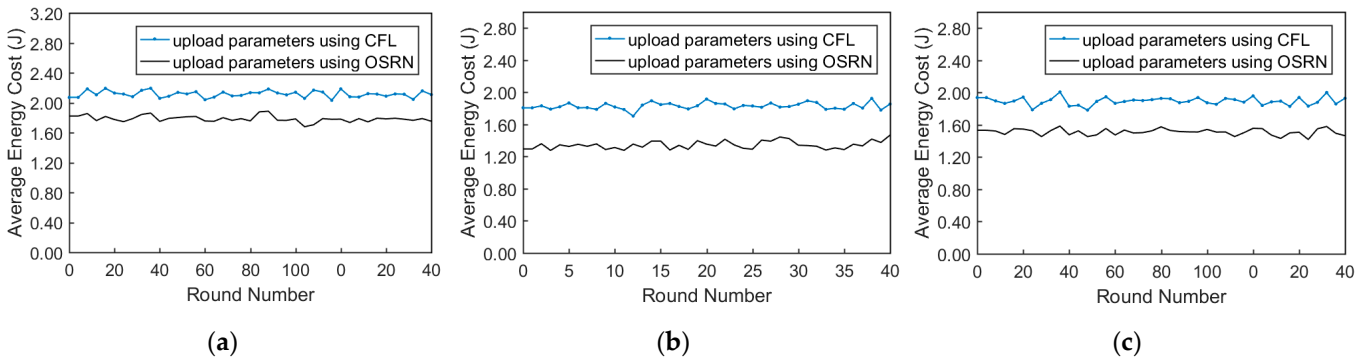

Figure 7. Energy cost comparison with CFL under different deployment locations of edge servers. (**a**) Four Corners and the Center; (**b**) Center; (**c**) Random Deployment.

As shown in Figure 8a–c, the average energy cost of OSRN is lower than that of THF, with an average reduction of 8.74%, 22.54%, and 13.54%, respectively. The core idea of

THF is also to optimize the process of uploading model parameters through clustering, and nodes upload parameters to edge servers through cluster heads. However, cluster heads are not randomly selected. In THF, all nodes in the network are divided into different clusters by agglomerative clustering. The head node of each cluster is selected based on the bandwidth of each node and distance to the edge server. The nodes in the same cluster upload their parameters to the edge server through the same head node. That is, in each cluster, there is only one relay node in the process of uploading parameters to the cloud server. However, in the OSRN, we continuously detect the shortest distance of all neighboring nodes of each node within the remaining delay to ensure that each node can select the optimal relay node with the minimum energy cost to complete parameter uploading. Moreover, the neighboring nodes of each node can be fully utilized. Therefore, the performance of OSRN is better than THF.

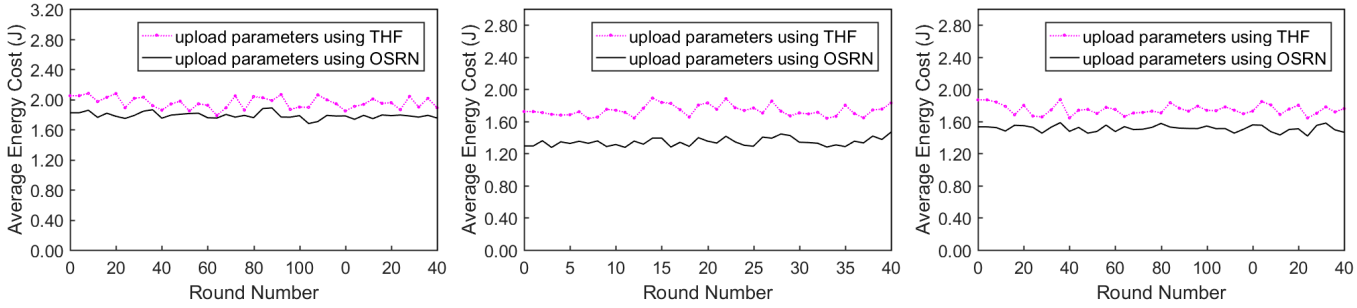

Figure 8. Energy cost comparison with THF under different deployment locations of edge servers. (**a**) Four Corners and the Center; (**b**) Center; (**c**) Random Deployment.

Figure 9 illustrates the average energy cost while directly uploading parameters using HierFAVG, randomly selecting relay nodes using SNNR, as well as using CFL and THF, when the total number of nodes varies. It can be observed that with the increase of the total number of nodes, the fluctuation of average energy cost with using HierFAVG, SNNR, CFL, and THF is not obvious and fluctuates up and down. It is also found that the average energy cost using OSRN is always no higher than the above four comparison algorithm, which demonstrates the better performance of OSRN. When the total number of nodes is in the range of $[50, 150]$, with more cooperative nodes, each node has more neighboring nodes to choose from in the relay node selection process and may choose a more appropriate relay to upload parameters cooperatively. Therefore, the average energy cost of OSRN decreases gradually within this range.

Figure 10 illustrates the impact of maximum delay constraint on average energy cost. It can be observed that with the increase of the maximum delay constraint, the average energy cost of using HierFAVG, SNNR, CFL, and THF fluctuates up and down, and the fluctuation is not obvious. When the maximum delay constraint is in the range of [30,70], the average energy cost using OSRN is always no higher than the above four comparison algorithms and decreases gradually within this range. This is because when the maximum delay constraint becomes larger, nodes have sufficient remaining time to select a better neighboring node as a relay for uploading parameters cooperatively. In addition, each mobile node may move closer to the edge server within the remaining delay constraints. Therefore, the energy cost of OSRN decreases with the increase of maximum delay constraint.

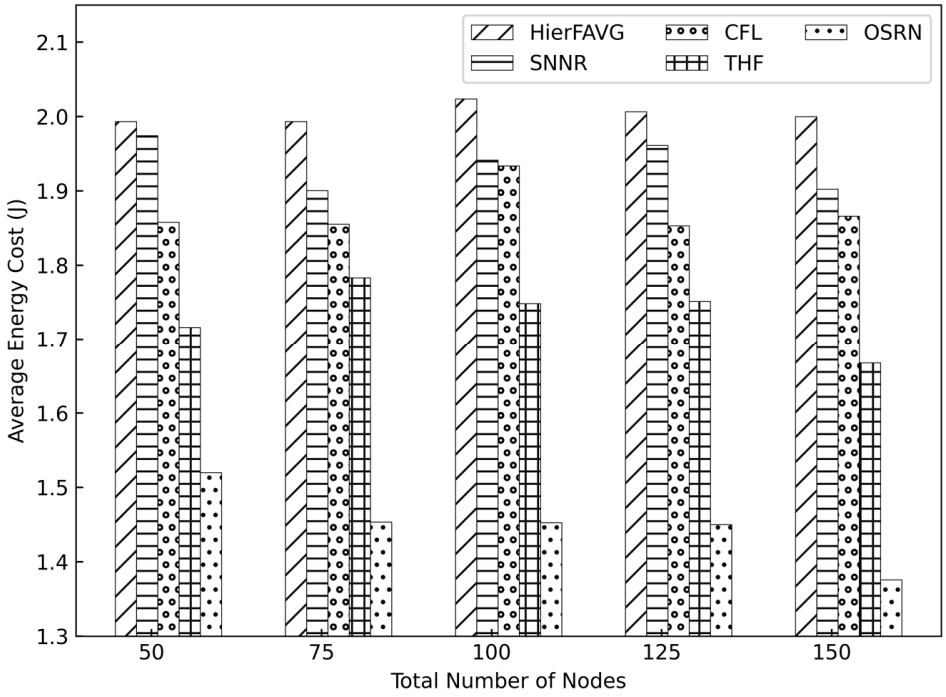

Figure 9. Average energy cost vs. total number of nodes.

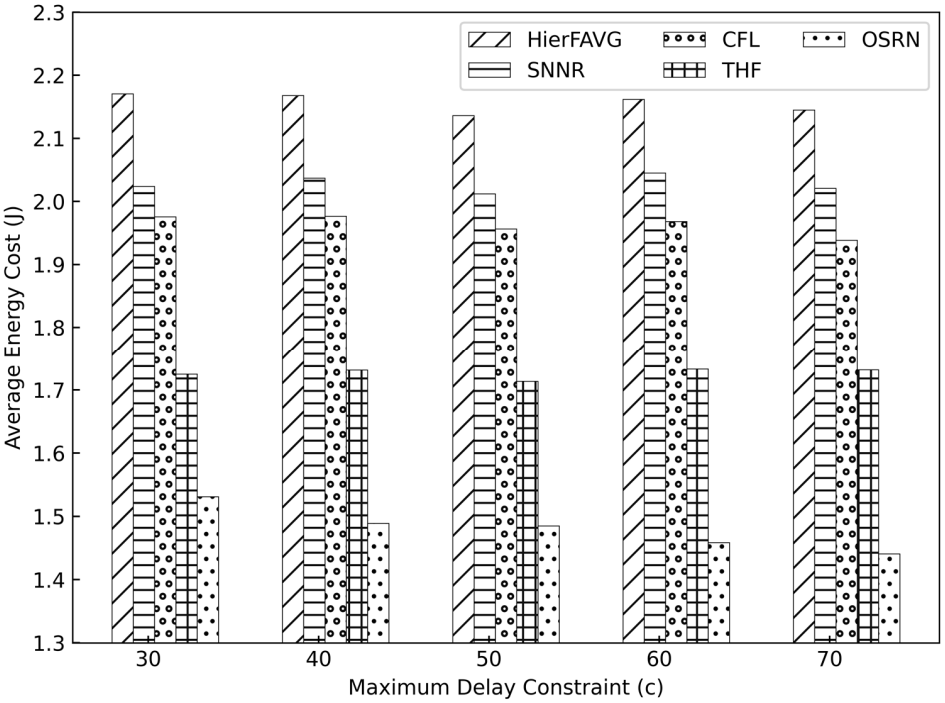

Figure 10. Average energy cost vs. maximum delay constraint.

6. Conclusions

In HFL, high energy cost is incurred by nodes uploading parameters to the edge server. To reduce the energy cost, we investigate the optimization problem for energy with the constraint of maximum transmission delay, which is proven NP-hard. To solve this problem, we convert it into an optimal relay node selection problem for node cooperation and propose an online algorithm OSRN. Opportunistic communication is used for cooperation among nodes. In OSRN, nodes continuously detect neighboring nodes and observe the optimal value of their combined cost, as well as the expected payoff obtained by their

cooperation. Based on this, nodes decide whether to select neighboring nodes as relays. Through theoretical analysis, we prove the existence of the optimal stopping rule and optimal solution for the optimal relay node selection problem used for node cooperation. We also prove the NP-hardness of the problem investigated and the competition ratio that can be achieved by OSRN. Through thorough simulation experiments, it is found that the energy cost in OSRN is reduced by 24.49% and 22.04% compared to that with HierFAVG and SNNR, respectively. Moreover, it is also observed that OSRN can reduce the energy cost by 20.20% and 13.54%, compared with the existing CFL and THF. In addition, in our experimental design, we also examine the impact of the total number of nodes and the maximum delay constraint on the performance of the proposed online node cooperation scheme and derive meaningful experimental results and findings. OSRN performs better when the total number of nodes and the maximum delay constraint become progressively larger.

In future work, mobile nodes will not only cooperate but also compete with each other during model training, and lower energy cost can be achieved through an effective incentive mechanism. Moreover, in the typical three-layer hierarchical federation learning we consider, only one layer of edge servers is introduced as a transmission intermediary, and multi-layer edge model aggregation can be performed by introducing multi-layer edge servers to reduce communication overhead. Therefore, under multi-layer model aggregation, which edge service layers the nodes upload parameters to and which edge layers participate in model aggregation is an issue worthy of further study.

Author Contributions: Conceptualization, Z.L., S.Z. and X.C.; methodology, Z.L., S.Z. and X.C.; software, S.Z.; validation, Z.L. and S.Z.; formal analysis, S.Z.; investigation, Z.L. and S.Z.; resources, X.C.; data curation, S.Z.; writing—original draft preparation, S.Z.; writing—review and editing, Z.L. and X.C.; visualization, S.Z.; supervision, Z.L.; project administration, Z.L. and X.C.; funding acquisition, Z.L. and X.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Beijing Natural Science Foundation (4232024), National Key R&D Program of China (2022YFF0604502), National Natural Science Foundation of China (61872044), and the Beijing Municipal Program for Top Talent.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article. All the data used to support the findings of this study are included within the article as shown in the references.

Acknowledgments: This work is partially supported by the Beijing Natural Science Foundation (4232024), National Key R&D Program of China (2022YFF0604502), National Natural Science Foundation of China (61872044), and the Beijing Municipal Program for Top Talent. The conference paper "Energy-Efficient Online Node Cooperation Strategy for Hierarchical Federated Learning" has been accepted in IEEE UIC 2022—IEEE International Conference on Ubiquitous Intelligence and Computing.

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

References

1. Zhou, Z.; Chen, X.; Li, E.; Zeng, L.; Luo, K.; Zhang, J. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proc. IEEE* **2019**, *107*, 1738–1762. [CrossRef]
2. Put More of Your Business Data to Work—From Edge to Cloud. Available online: https://www.seagate.com/files/www-content/our-story/rethink-data/files/Rethink_Data_Report_2020.pdf (accessed on 26 November 2022).
3. Li, P.; Li, J.; Huang, Z.; Li, T.; Gao, C.-Z.; Yiu, S.-M.; Chen, K. Multi-key privacy-preserving deep learning in cloud computing. *Future Gener. Comput. Syst.* **2017**, *74*, 76–85. [CrossRef]
4. Custers, B.; Sears, A.M.; Dechesne, F.; Georgieva, I.; Tani, T.; van der Hof, S. *EU Personal Data Protection in Policy and Practice*; TMC Asser Press: The Hague, The Netherlands, 2019; Volume 29, pp. 1–249.
5. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*; PMLR: New York, NY, USA, 2017; pp. 1273–1282.
6. Lim, W.Y.B.; Luong, N.C.; Hoang, D.T.; Jiao, Y.; Liang, Y.-C.; Yang, Q.; Niyato, D.; Miao, C. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 2031–2063. [CrossRef]

7. Kong, X.; Wu, Y.; Wang, H.; Xia, F. Edge Computing for Internet of Everything: A Survey. *IEEE Internet Things J.* **2022**, *9*, 23472–23485. [CrossRef]
8. Kong, X.; Wang, K.; Hou, M.; Hao, X.; Shen, G.; Chen, X.; Xia, F. A Federated Learning-based License Plate Recognition Scheme for 5G-enabled Internet of Vehicles. *IEEE Trans. Ind. Inform.* **2021**, *17*, 8523–8530. [CrossRef]
9. Liu, L.; Zhang, J.; Song, S.H.; Letaief, K.B. Client-edge-cloud hierarchical federated learning. In Proceedings of the IEEE International Conference on Communications, Dublin, Ireland, 7–11 June 2020; pp. 1–6.
10. Abdellatif, A.A.; Mhaisen, N.; Mohamed, A.; Erbad, A.; Guizani, M.; Dawy, Z.; Nasreddine, W. Communication-efficient hierarchical federated learning for IoT heterogeneous systems with imbalanced data. *Future Gener. Comput. Syst.* **2022**, *128*, 406–419. [CrossRef]
11. Saadat, H.; Aboumadi, A.; Mohamed, A.; Erbad, A.; Guizani, M. Hierarchical federated learning for collaborative IDS in IoT applications. In Proceedings of the 2021 10th Mediterranean Conference on Embedded Computing, Budva, Montenegro, 7–10 June 2021; pp. 1–6.
12. Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konečný, J.; Mazzocchi, S.; McMahan, H.B.; et al. Towards federated learning at scale: System design. *Proc. Mach. Learn. Syst.* **2019**, *1*, 374–388.
13. Wang, S.; Tuor, T.; Salonidis, T.; Leung, K.K.; Makaya, C.; He, T.; Chan, K. Adaptive federated learning in resource constrained edge computing systems. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1205–1221. [CrossRef]
14. Yao, X.; Huang, C.; Sun, L. Two-stream federated learning: Reduce the communication costs. In Proceedings of the 2018 IEEE Visual Communications and Image Processing (VCIP), Taichung, Taiwan, 9–12 December 2018; pp. 1–4.
15. Liu, L.; Zhang, J.; Song, S.H. Edge-Assisted Hierarchical Federated Learning with Non-IID Data. *arXiv* **2019**, arXiv:1905.06641.
16. Reisizadeh, A.; Mokhtari, A.; Hassani, H.; Jadbabaie, A.; Pedarsani, R. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. *Int. Conf. Artif. Intell. Stat.* **2020**, *108*, 2021–2031.
17. Wang, S.; Lee, M.; Hosseinalipour, S.; Morabito, R.; Chiang, M.; Brinton, C.G. Device sampling for heterogeneous federated learning: Theory, algorithms, and implementation. In Proceedings of the IEEE INFOCOM 2021-IEEE Conference on Computer Communications, Vancouver, BC, Canada, 10–13 May 2021; pp. 1–10.
18. Konecny, J.; McMahan, H.B.; Felix, X.Y.; Richtárik, P.; Suresh, A.T.; Bacon, D. Federated Learning: Strategies for Improving Communication Efficiency. *arXiv* **2016**, arXiv:1610.05492.
19. Alistarh, D.; Grubic, D.; Li, J.; Tomioka, R.; Vojnovic, M. QSGD: Communication-efficient SGD via Gradient Quantization and Encoding. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1709–1720.
20. Suresh, A.T.; Felix, X.Y.; Kumar, S.; McMahan, H.B. Distributed mean estimation with limited communication. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Volume 70, pp. 3329–3337.
21. Zhang, X.; Fang, M.; Liu, J.; Zhu, Z. Private and communication-efficient edge learning: A sparse differential gaussian-masking distributed SGD approach. In Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing, Shanghai, China, 26–29 July 2020; pp. 261–270.
22. Zheng, H.; Gao, M.; Chen, Z.; Feng, X. A distributed hierarchical deep computation model for federated learning in edge computing. *IEEE Trans. Ind. Inform.* **2021**, *17*, 7946–7956. [CrossRef]
23. Liu, L.; Zhang, J.; Song, S.; Khaled, B.L. Hierarchical quantized federated learning: Convergence analysis and system design. *arXiv* **2021**, arXiv:2103.14272.
24. Ren, L.; Liu, Y.; Wang, X.; Lv, J.; Deen, M.J. Cloud-Edge based Lightweight Temporal Convolutional Networks for Remaining Useful Life Prediction in IIoT. *IEEE Internet Things J.* **2020**, *8*, 12578–12587. [CrossRef]
25. Lou, S.; Chen, X.; Wu, Q.; Zhou, Z.; Yu, S. HFEL: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 6535–6548.
26. Wang, X.; Yang, L.T.; Ren, L.; Wang, Y.; Deen, M.J. A Tensor-based Computing and Optimization Model for Intelligent Edge Services. *IEEE Netw.* **2022**, *36*, 40–44. [CrossRef]
27. Yi, Y.; Zhang, Z.; Yang, L.T.; Wang, X.; Gan, C. Edge-aided Control Dynamics for Information Diffusion in Social Internet of Things. *Neurocomputing* **2021**, *485*, 274–284. [CrossRef]
28. Wang, Z.; Xu, H.; Liu, J.; Huang, H.; Qiao, C.; Zhao, Y.; Wang, Z.; Xu, H.; Liu, J.; Huang, H.; et al. Resource-Efficient Federated Learning with Hierarchical Aggregation in Edge Computing. In Proceedings of the IEEE INFOCOM 2021-IEEE Conference on Computer Communications, Vancouver, BC, Canada, 10–13 May 2021; pp. 1–10.
29. Hosseinalipour, S.; Brinton, C.G.; Aggarwal, V.; Dai, H.; Chiang, M. From federated to fog learning: Distributed machine learning over heterogeneous wireless networks. *IEEE Commun. Mag.* **2020**, *58*, 41–47. [CrossRef]
30. Liu, Y.; Pan, C.; You, L.; Han, W. D2D-Enabled User Cooperation in Massive MIMO. *IEEE Syst. J.* **2020**, *14*, 4406–4417. [CrossRef]
31. Park, S.H.; Jin, X. Joint Secure Design of Downlink and D2D Cooperation Strategies for Multi-User Systems. *IEEE Signal Process. Lett.* **2021**, *28*, 917–921. [CrossRef]
32. Mustafa, H.A.; Shakir, M.Z.; Imran, M.A.; Tafazolli, R. Distance Based Cooperation Region for D2D Pair. In Proceedings of the 2015 IEEE 81st Vehicular Technology Conference (VTC Spring), Glasgow, UK, 11–14 May 2015; pp. 1–6.
33. Asad, M.; Moustafa, A.; Rabhi, F.A.; Aslam, M. THF: 3-Way Hierarchical Framework for Efficient Client Selection and Resource Management in Federated Learning. *IEEE Internet Things J.* **2021**, *9*, 11085–11097. [CrossRef]

34. Wang, Z.; Wang, Y.; Wang, L.; Wang, T.; Xu, D. A delay-driven early caching and sharing strategy for D2D transmission network. In Proceedings of the 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), Antwerp, Belgium, 25–28 May 2020; pp. 1–5.
35. Qiu, M.; Chen, Z.; Liu, M. Low-power low-latency data allocation for hybrid scratch-pad memory. *IEEE Embed. Syst. Lett.* **2014**, *6*, 69–72. [CrossRef]
36. Zhao, L.; Ran, Y.; Wang, H.; Wang, J.; Luo, J. Towards Cooperative Caching for Vehicular Networks with Multi-level Federated Reinforcement Learning. In Proceedings of the ICC 2021-IEEE International Conference on Communications, Montreal, QC, Canada, 14–23 June 2021; pp. 1–6.
37. Cheng, R.; Sun, Y.; Liu, Y.; Xia, L.; Sun, S.; Imran, M.A. A Privacy-preserved D2D Caching Scheme Underpinned by Blockchain-enabled Federated Learning. In Proceedings of the 2021 IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 7–11 December 2021; pp. 1–6.
38. Cheng, R.; Sun, Y.; Liu, Y.; Xia, L.; Feng, D.; Imran, M.A. Blockchain-Empowered Federated Learning Approach for an Intelligent and Reliable D2D Caching Scheme. *IEEE Internet Things J.* **2022**, *9*, 7879–7890. [CrossRef]
39. Qiao, D.; Guo, S.; Liu, D.; Long, S.; Zhou, P.; Li, Z. Adaptive Federated Deep Reinforcement Learning for Proactive Content Caching in Edge Computing. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 4767–4782. [CrossRef]
40. Khanal, S.; Thar, K.; Huh, E.N. Route-Based Proactive Content Caching Using Self-Attention in Hierarchical Federated Learning. *IEEE Access* **2022**, *10*, 29514–29527. [CrossRef]
41. Liu, S.; Zheng, C.; Huang, Y.; Quek, T.Q.S. Distributed Reinforcement Learning for Privacy-Preserving Dynamic Edge Caching. *IEEE J. Sel. Areas Commun.* **2022**, *40*, 749–760. [CrossRef]
42. Batabyal, S.; Bhaumik, P. Mobility models, traces and impact of mobility on opportunistic routing algorithms: A survey. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1679–1707. [CrossRef]
43. Zhan, Y.; Li, P.; Qu, Z.; Zeng, D.; Guo, S. A learning-based incentive mechanism for federated learning. *IEEE Internet Things J.* **2020**, *7*, 6360–6368. [CrossRef]
44. Amanda, H.; Bryant, A.J. The quadratic multiple knapsack problem and three heuristic approaches to it. In Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, Seattle, WA, USA, 8–12 July 2006; pp. 547–552.
45. Wang, F.; Wang, G. Study on Energy Minimization Data Transmission Strategy in Mobile Cloud Computing. In Proceedings of the IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, Guangzhou, China, 8–12 October 2018; pp. 1211–1218.
46. Zheng, D.; Ge, W.; Zhang, J. Distributed opportunistic scheduling for ad hoc networks with random access: An optimal stopping approach. *IEEE Trans. Inf. Theory* **2008**, *55*, 205–222.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.