Reinforcement Learning for Edge Device Selection using Social Attribute Perception in Industry 4.0

Peiying Zhang, Peng Gan, Gagangeet Singh Aujla, and Ranbir Singh Batth

***Abstract*—In the 5G era, the problem of data islands in various industries restricts the development of artificial intelligence technology, so data sharing is proposed. High-quality data sharing directly affects the effectiveness of machine learning models, but data leakage and abuse will inevitably occur in the process. As a consequence, in order to solve this problem, federated learning is proposed. This method uses the personalized data of multiple edge devices to train the model. The central server collects the training results of the edge devices and updates the global model, and then iteratively tests and updates the model through the edge devices. However, edge devices may have problems such as unbalanced load and exit from the training process, which makes the training time of the model long and the effect is poor. Therefore, in the process of federated learning, the selection of reliable and high-quality edge devices becomes crucial. On this basis, in this paper, we introduces reinforcement learning (RL) to pre-select edge devices and obtain a set of candidate devices, then determines reliable edge devices through social attribute perception. Simulation experiment data analysis demonstrate that this scheme can improve the reliability of federated learning and complete the training process in a shorter time, the efficiency of federated learning increased by approximately 10.3%.**

***Index Terms*—Data Sharing, Federated Learning, Edge Computing, Reinforcement Learning, Social Attribute Perception**

I. Introduction

In the era of Industry 4.0, smart technologies are diffusely used in the Internet of Things [1], [2], such as smart wearable devices [3], [4], smart homes, digital healthcare [5], smart transportation, and vehicular networks [6]. The application of these smart technologies in Industry 4.0 has improved the efficiency and fault tolerance of industrial production, enabling the existing industrial system to face the challenges of complex environments. These applications generate a large amount of personalized data. Recent years, in order to make applications more intelligent and gradually apply to Industry 4.0, machine learning has been used to train models on these data. However, traditional machine learning concentrates large amounts of user data in a single data center for training, which causes the inevitable leakage of user privacy data during the training process. For example, in an intelligent medical system, patient information will be processed centrally on a central server, which will inevitably cause the leakage of patient information, because some patients do not want others to know this information. To avoid data leakage, federated learning is proposed [7]–[10]. Federated learning as a new distributed privacy protection machine learning training [11]–[15] framework has received more and more attention, which prevents users from exposing their data to enterprises or other participants, thereby protecting their privacy and data security to a certain extent. It is a new type of decentralized machine learning method under Industry 4.0. Fig.1 shows the application of federated learning, it distributes existing training models to multiple edge devices (personal devices), and uses the local data of the edge device to train the model. The central server accepts and integrates the training results of each edge device to update the global model. Through continuous iteration of the above steps, the improved global model will be continuously downloaded to the local for verification, and finally a smarter, low-latency, low-overhead, privacy-protecting, and user-friendly model will be obtained.

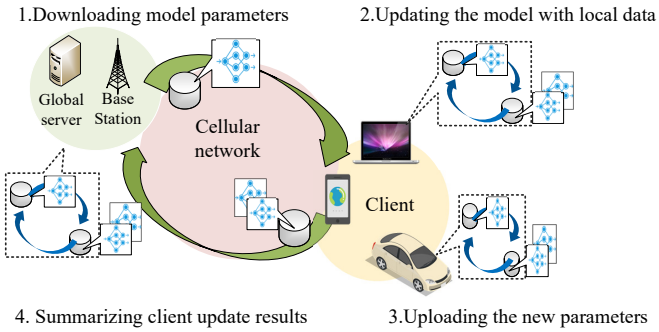

Fig. 1: An application of federated learning.

TABLE I: Three types of federated learning characteristics.

Category	Data feature overlap	User overlap	Training method
Horizontal federated learning	Small quantity	Large amount	Divided by user dimension.
Longitudinal federated learning	Large amount	Small quantity	Divided by data feature dimensions.
Federated transfer learning	Small quantity	Small quantity	Transfer learning.

This work is partially supported by the Start-Up Fund from the Durham University, and partially supported by the Major Scientific and Technological Projects of CNPC under Grant ZD2019-183-006, partially supported by Shandong Provincial Natural Science Foundation, China under Grant ZR2020MF006, and partially supported by "the Fundamental Research Funds for the Central Universities" of China University of Petroleum (East China) under Grant 20CX05017A. (Corresponding authors: Gagangeet Singh Aujla and Peiying Zhang.)

Peiying Zhang and Peng Gan are with the College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China. (E-mail: zhangpeiying@upc.edu.cn, ganpeng.upc@qq.com)

G.S. Aujla is with the School of Computing Science, Durham University, United Kingdom. E-mail: gagi_aujla82@yahoo.com

R.S. Batth is with the School of Computer Science Engineering, Lovely Professional University, India. E-mail: ranbir.21123@lpu.co.in

As shown in TABLE I, federated learning can be classified into three categories based on the characteristics of federated learning data and the amount of user overlap, namely horizontal federated learning, vertical federated learning, and federated transfer learning. Each participant in the three types of federated learning only needs to maintain the user's local data, which is the main reason for the protection of user privacy. In addition to privacy protection, the advantages of federated learning are also reflected in the following two points:

(1) Federated learning can complete model training on large-scale data.

(2) The data required for training has the characteristics of flexibility and personalization.

Although federated learning provides a good solution for solving user privacy problems, it still has the following challenges:

(1) Poisoning attack: As revealed in Fig.2, during the training process, malicious edge devices will attack the training data set, thereby affecting the prediction of the model. Therefore, it is particularly important to choose trusted and safe edge devices to participate in the training process during the federated learning process, which directly affects the accuracy and safety of the training model.

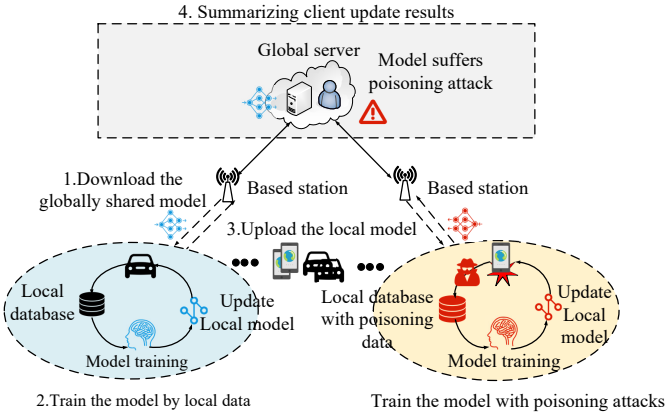

Fig. 2: Data poisoning and model poisoning.

(2) Load balancing of edge devices participating in federated learning: The problem of load imbalance caused by limited computing resources of edge devices is still a bottleneck restricting the efficiency of federated learning.

(3) Reliability of edge devices participating in federated learning: federated learning is a continuous iterative process. Throughout the training process, edge devices continue to communicate with the central server. However, in fact, not all equipment can fully participate in the entire training process, which seriously affects the quality of the final training results.

To settle the above problems, we introduced an edge device selection strategy before federated learning with reinforcement learning (RL) and social attribute perception (RL-SAP). As indicated in Fig.3, we first initialize the environment and obtain resource information of optional edge devices. Then we use reinforcement learning and social attribute perception to select edge devices that participate in federated learning, and on this basis, we carry out model distribution, training, and aggregation of training results. Through the above steps, the problem of unreliable equipment participating in model training can be effectively avoided. The main contributions of this paper are as follows:

(1) Firstly, in order to achieve load balancing, it is essential to ensure that the computing power and communication efficiency of the equipment participating in the training meet the requirements. Therefore, we use reinforcement learning (RL) methods to pre-select optional edge devices. Specifically, we construct a four-layer policy network as the RL agent, and then extract the three attributes of the edge device CPU, memory, and bandwidth to form the feature matrix as the agent's training environment. By convolution and softmax operations on the feature matrix, we can obtain the probability that the edge device is selected, thereby obtaining a pre-selected set of edge devices.

(2) Secondly, on the basis of the set of candidate edge devices, we determine the final edge devices participating in training in line with the perception of social attributes, which ensures the reliability of the devices.

(3) Finally, we design and implement the experiment of the scheme proposed in this paper, and verify the feasibility of the scheme from the efficiency of training.

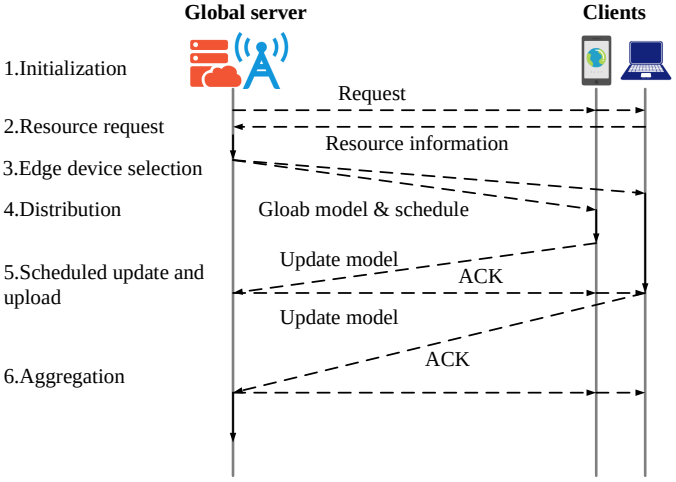

Fig. 3: The RL-SAP flow chart.

The rest of this paper is organized as follows. Section II analyzes the research status of Federated Learning. Section III describes the pre-selection process of edge devices based on reinforcement learning. Section IV introduces the process of determining edge devices based on social attribute perception. Section V compares and analyzes the effectiveness of the proposed scheme from four angles. Section VI summarizes this article and introduces future work.

II. Related Work

Federated learning is an emerging machine learning technology. Literature [16] introduces federated learning to the public for the first time, which is a decentralized machine learning for model training through mobile devices. Following the literature [16], scholars have mainly conducted research on improving the security of the federated learning protocol

[17], [18]. However, the methods in these literatures mainly aim at minimizing the computational overhead or time of general computing tasks, and do not fundamentally improve the efficiency of model training. Literature [18]–[22] have carried out research on privacy protection in response to the problem of information leakage in the process of federated learning. Literature [19] proposes a dual shielding protocol for the privacy tracking problem that may be caused in the process of federated learning, and the central server needs to provide proof of the correctness of the aggregation results to the edge devices participating in the training. The literature [21] combines secure multi-party computing with gradient selection, where secure multi-party computing is used to protect user information, and gradient selection is used to reduce communication overhead. In the literature [21], secure multi-party computing and gradient selection are balanced. In addition, a method based on secret sharing and gradient selection is proposed. But this scheme will cause additional communication overhead and computational overhead. The literature [22] proposes the PEFL scheme for privacy leakage caused by shared parameters, which enhances privacy protection in a non-interactive way and improves the efficiency of federated learning. Literature [23]–[26] mainly focus on improving the efficiency of federated learning. Literature [23] introduces a new federated learning algorithm called CatFedAvg, which not only improves the communication efficiency, but also improves the learning quality by using the strategy of maximizing category coverage. Literature [24] puts forward the STC compression method with the goal of reducing communication overhead. Through this data compression technology, the communication cost is reduced by three orders of magnitude, and it is significantly better than the joint average. In the literature [26], aiming at the problem of excessive consumption of communication resources in the process of federated learning, a multi-objective evolutionary algorithm is used to optimize the neural network model in federated learning. This solution can reduce the overall test errors while minimizing overhead.

Literature [27], [28] focus on the application of federated learning. Literature [28] applies federated learning to the smart keyboard to predict the next word. The literature [27] utilizes federated learning and convolutional neural networks to intrusion detection for the intrusion detection problems in the deep learning process. As a consequence, the efficiency of intrusion detection has improved. In the process of federated learning, the reliability of equipment directly affects the accuracy and efficiency of model training. However, none of the above-mentioned literatures has studied the reliability of mobile devices. Therefore, literature [29]–[31] carry out research on equipment reliability, and literature [29] uses reputation as a measure of equipment to ensure the reliability of federated learning. In order to ensure the accuracy of prediction, the literature [31] combines differential privacy and SMC to ensure the reliability of participating training equipment without affecting privacy. At the same time, the scheme avoids the threat of reasoning to a certain extent.

This paper is similar to the literature [29], [30], focusing on the reliability of the equipment participating in the training process. Fig.4 reveals the process of selecting edge devices in this paper.

III. Edge device pre-selection

A. Algorithm description and model establishment

This section mainly uses RL algorithms to pre-select edge devices participating in federated learning.

This paper constructs a four-layer policy network as a RL agent, which includes an input layer, a convolutional layer, a softmax layer, and an output layer. In order to enable the agent to be trained in as realistic an environment as possible, we extracted the three attributes of each edge device and used them to form a feature matrix as the agent's training environment. Then the convolution layer receives the feature matrix and performs convolution operation on it. Finally, the softmax layer outputs the probability of each edge device being selected.

The key to using RL algorithms to calculate the probability distribution of edge devices being selected lies in the level of understanding of all edge devices. We selected the three attributes of the edge device's CPU, bandwidth, and memory as the input of the agent. But in fact, the attributes of edge devices are far more than these three. The more attributes of the selected edge device, the higher the complexity of the algorithm. Therefore, after comprehensive consideration, we only selected the above three attributes.

This method learns in line with the load balance of edge devices under different available resources. Firstly, it is necessary to determine the matching relationship between the model training task and the edge devices participating in training. Therefore, we solve it by finding the optimal scheduling strategy vector, and realize it by strengthening Q-learning. When the central server delivers the model, it needs to consider the real-time load of the edge devices participating in training, so we weigh the workload status of the edge devices from three aspects: memory utilization, CPU utilization, and bandwidth utilization. This paper defines these three measurement elements as vectors, and the state vector of the edge device can be expressed by the following formula.

$$\bar{\alpha} = (memory, cpu, bandwidth)\,. \tag{1}$$

B. Edge device pre-selection algorithm based on reinforcement learning

We learn in line with the network load balance under different model distribution strategies, collect the load status of different edge devices through local macro base stations, and save the final learning results for the final determination of edge devices. We redefine the state and rewards in RL as follows.

(1) **Status:** The state in the RL [32]–[37] process corresponds to the network load status after the trained model is delivered to a certain edge mobile device. Assuming that the number of optional edge mobile devices is K, so there are K states in the entire system. We evaluate the load balance status of the entire network based on the degree of load balance.

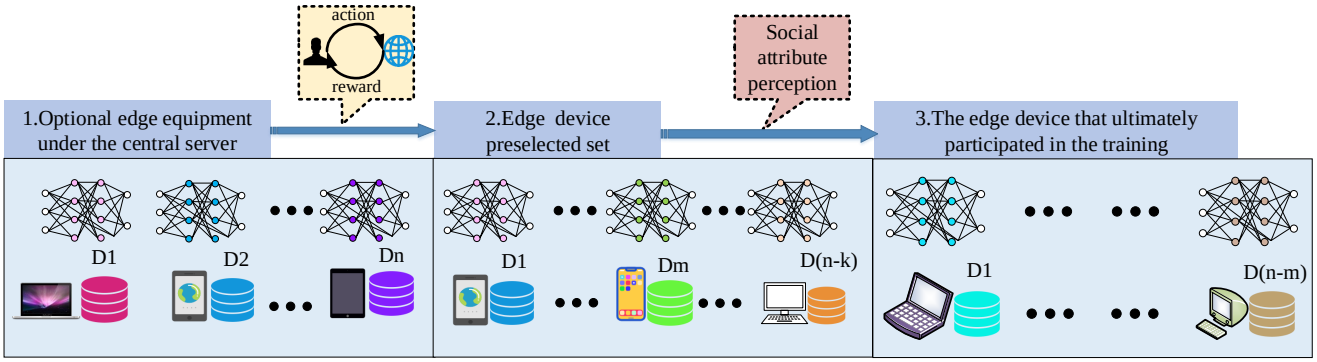

Fig. 4: Pre-selection of edge devices based on reinforcement learning.

Firstly, we calculate the load center value of different resources of the entire network in line with the load status of all optional edge devices participating in federated learning. The calculation method is shown in equation 2, and then the load balance is calculated.

$$l_m = \frac{1}{k}\sum_{i=1}^{k} num\_re_i^m, \tag{2}$$

among them, $num\_re_i^m$ denotes the number of m resources available on the edge device i, and k indicates the number of optional edge devices. Based on equation 2, we can calculate the degree of load balance, the calculation method is shown in equation 3.

$$lb = \frac{1}{k}\sum_{m=1}^{\xi}\sum_{i=1}^{k}\left(l_m^i - l_m\right)^2. \tag{3}$$

meanwhile, we measure the real-time load of the edge device after the model is delivered to the edge device i in line with equation 4.

$$\varepsilon_{ji} = \sum_{m=1}^{\xi} w_{ji}^m\left(1-\mu_{ji}\right), \tag{4}$$

where $m=1$ denotes memory resources, $m=2$ represents CPU resources, and $m=3$ means bandwidth resources. w_{ji}^m is the weight of each resource after model j is delivered to edge device i. μ_{ji}^m indicates the utilization rate of m resources, as revealed in equation 5.

$$\mu_{ji}^m = \frac{amo\_re^m}{num\_re_i^m}. \tag{5}$$

The notation amo_re^m denotes the number of m resources required during model training. Equation 6 is the calculation method of $num\_re_i^m$.

$$num\_re_i^m = tot\_re_i^m - amo\_re^m, \tag{6}$$

where $tot\_re_i^m$ represents the total amount of m resources on the edge device i.

(2) **Reward:** We define the reward function of reinforcement learning as $-lb(s,a)$, where s and a represent the current state and action of the edge device, and $lb(s,a)$ denotes the degree of load balance. If an edge device is overloaded or fully loaded during model training, set the reward value to -1.

This paper uses Q-learning algorithm for RL. It records the Q value of each state of the edge device and stores the Q value of each step in the Q table, which can be regarded as a long-term reward. The Q value of each state can be expressed by the following formula.

$$Q(s,a) = \mathrm{Re}(s,a) + \delta.maxQ\left(s^{'},a^{'}\right), \tag{7}$$

among them, $\mathrm{Re}(s,a)$ represents the reward in the initial state, $s^{'}$ and $a^{'}$ represent the next state and action, respectively. δ is the learning parameter, $\delta \in [0,1]$. If the value of δ approaches 0, direct rewards are mainly considered. If it approaches 1, we need to pay special attention to future rewards. Each step of $Q(s,a)$ will be iterated until the optimal result is obtained.

The process of pre-selecting optional edge devices using the above mentioned Q-learning of RL is revealed in Algorithm 1.

Algorithm 1 Training

Input: Parameters of policy network;
Output: The probability of each edge device being selected;
Initialize all agent's parameters;
while $iteration \leq Epoch$ **do**
  for $TASKS \in trainingSet$ **do**
    $getFeatureMatrix()$;
    $probability = agent.getOutput(M_f)$;
    %Get the probability distribution of edge device;
    $Candidate_{Edgedevice} = sample(probability)$;
    $getGradient(Candidate_{Edgedevice})$;
    $Calculate\ reward$;
    $Clear\ gradients$;
  end for
  $iteration++$;
end while

C. Algorithm complexity analysis

The time complexity of our proposed RL-based edge device pre-selection algorithm is $O(C_{Tasks}(Ed_n * d + C_{Task}))$. Among them, C_{Tasks} represents the number of iterations of the training task, Ed_n represents the number of all optional edge devices,

d represents the dimension of the state vector, and C_{Task} is the inherent complexity of each training task. It is worth noting that the time complexity of the RL-based edge device pre-selection algorithm is greatly affected by factors such as the learning rate, the size of the selected convolution kernel, and so on. Therefore, only the general time complexity can be given here.

IV. Social attribute perception

A. Social attribute constraints

Federated learning updates and tests the model continuously iteratively. However, in the process of federated learning, some devices cannot iteratively interact with the central server in real time. Therefore, after pre-selecting the optional edge devices participating in the training in line with the method in section 3, it is necessary to consider the social attributes [38], [39] of mobile edge devices , so as to determine the reliable mobile terminal that ultimately meets the conditions.

Due to the edge mobile devices participating in training are mainly used by users, they have strong social attributes. Therefore, we measure whether the connection is reliable in line with the social attributes of the mobile device. The social attributes considered in this paper mainly include the activity of the device and the intimacy with the central server.

Device activity: Some mobile devices in the network have higher network resource utilization and higher activity. Such devices frequently interact with other devices and central servers. Therefore, how to quantify the activity of the device is of great significance for selecting the edge device that ultimately participates in the training process. This paper assumes that the number oTAf other devices interacting with the device within a certain period of time is $\sum_{y=1}^{n}\tau(x,y)$, and the total number of optional devices is n. Then the ratio act_x of $\sum_{y=1}^{n}\tau(x,y)$ to n represents the activity of the device.

$$act_x = \frac{\sum_{y=1}^{n}\tau(x,y)}{n}. \tag{8}$$

$$\tau(x,y) = \begin{cases} 1, x, y \text{ have interactive behavior,} \\ 0, other. \end{cases} \tag{9}$$

Device and central server intimacy: We measure the difference in social attributes between devices based on the degree of interaction between the device and the central server, that is, the intimacy between the mobile device and the central server. we assume that the interaction duration between the device and the central server in the time period T is T_1, the interaction interval is T_2, and $T = T_1 + T_2$. Therefore, the interaction intimacy can be defined as the ratio of the average interaction interval to the time period T. The following formula represents the average interaction interval.

$$Avg(T_2) = \frac{\sum_{i=1}^{freq}(start_{i+1} - end_i)}{freq}, \tag{10}$$

among them, $freq$ represents the number of interactions between the device and the central server in the time period T, $start_{i+1}$ denotes the time when i+1 interaction starts, and end_i indicates the time when i interaction ends. Therefore, the intimacy C between the device and the central server is shown in equation 11.

$$\delta_{x,j} = \frac{\sum_{i=1}^{freq} start_{i+1} - end_i}{T.freq}. \tag{11}$$

If there is no interaction between the edge mobile device and the central server during the T period, the intimacy between them is denoted by the intimacy in the previous period. Since the reference value of information decreases with time, we update the intimacy according to the principle of exponential function decay.

$$k_{x,j} = \sum_{i=1}^{freq} \exp\left(-\frac{(nt - start_i)}{T}\right).\left(\frac{end_i - start_i}{T}\right), \tag{12}$$

among them, $k_{x,j}$ represents the new intimacy, and nt represents the current moment. According to equation 12, the communication probability $CPro_{x,j}$ between the edge device and the central server can be expressed by equation 13.

$$CPro_{x,j} = \frac{2}{1+\exp(-k_{x,j}/\delta_{x,j})} - 1, \tag{13}$$

among them, $k_{x,j}$ represents the updated intimacy, which can be calculated by formula (12). $\delta_{x,j}$ represents the initial intimacy, which is calculated by formula (11). We comprehensively consider the importance of device activity and its intimacy with the central server, and uses equation 14 to determine the edge devices that will ultimately participate in the training model. Where α is the weighting factor. The weight factor needs to be updated in real time according to the situation to effectively distinguish the importance of different terminals.

$$\omega_k = (1-\alpha)\,act_x + \alpha CPro_{x,j}. \tag{14}$$

B. Physical domain constraints

Due to the rapid increase of IoT data [40], traditional cellular technology consumes a lot of network resources and cannot meet the needs of IoT applications. Therefore, this paper uses the way that two users communicate directly (D2D) technology in the communication between the edge device and the central server. D2D communication enables data to be transmitted directly between devices, which can greatly reduce time delay. Thence, when selecting edge devices, we also need to consider the constraints of the physical domain to ensure that a reliable D2D link is established.

If the device a interacts with the central server b through D2D communication, then the device a will act as a D2D transmitter to construct a communication link to the central server. The signal to interference noise ratio (SINR) received at the central server should be greater than a given threshold. This value ensures reliable multi-hop communication, as revealed in the following equation.

$$\gamma_{de_a,se} = \frac{\left|h_{de_a,se}\right|^2 P_{de_a}}{\left|h_{me_c,se}\right|^2 P_{me_c} + \sigma^2} \geq \gamma_{de}^{th}, \tag{15}$$

where $h_{de_a,se}$ represents the channel gain between the device a and the central server, and $h_{me_c,se}$ indicates the channel gain between the multiplexed cellular user and the central

server. P_{de_a} denotes the transmit power of device a, and P_{me_c} represents the transmit power of cellular user c. $\gamma_{de_a,se}$ is the SINR of the channel between the device a and the central server, and γ_{de}^{th} represents the threshold limit that needs to be met.

From equation 16, it can be seen that the signal-to-noise ratio threshold is the minimum threshold that meets the task delay.

$$R_{de}^{D2D} = B\log_2\left[1+\min\left(\gamma_{de,se}\right)\right], \tag{16}$$

$$\gamma_{de}^{th} = 2^{\frac{R_{de}^{D2D}}{B}} - 1, \tag{17}$$

where B denotes the bandwidth of the D2D communication link. $\gamma_{de,se}$ represents the interference-to-noise ratio between the equipment and the central server.

The minimum required transmit power constraint for equipment a and the central server to maintain reliable transmission can be expressed by equation 18.

$$P_{de_a}^{\min} = \frac{\gamma_{de}^{th}\left(\left|h_{me_c,se}\right|^2 P_{me_c} + \sigma^2\right)}{\left|h_{de_a,se}\right|^2}, \tag{18}$$

where *se* represents the first hop relay.

The edge device pre-selection algorithm based on reinforcement learning ensures the stability of the entire federated learning process by learning the load balancing of the entire federated learning network. On the other hand, the use of social attribute perception to select the final edge device ensures the reliability of the edge device and effectively avoids the problem of edge device withdrawal from training and data poisoning. After using RL-SAP to select these reliable devices, they will download the latest training model from the central server and use the local private data to train the model. Then the training results are handed over to the central server to aggregate and update the model parameters. In this iteration, a safe and stable model can be obtained.

V. Experiment analysis

This chapter mainly verifies the performance of RL-SAP through simulation experiments. We divide the simulation experiment analysis into two parts.

(1) Firstly, the RL-based edge device pre-selection algorithm proposed in this paper was verified through the Matlab simulation platform. Since selecting the appropriate edge device to participate in the training of the model before federated learning is essentially to find a better scheduling strategy for training tasks, we compared RL-SAP with other task scheduling strategies. They are the WLB-ACO algorithm in the literature [41], the HETS algorithm and the Full-local strategy proposed in the literature [42].

(2) Secondly, the edge device selection scheme based on social attribute perception is verified through the NS2 simulation platform. The algorithms involved in the comparison mainly include the SDFM algorithm proposed in [43], the Random Walk algorithm proposed in [44] and the Bulle Rap algorithm proposed in [45]. This paper designs four experiments to verify the effectiveness of the proposed scheme from load balance, training time, average completion delay, and training accuracy.

We describe and summarize the above algorithms, as shown in TABLE II.

TABLE II: Algorithm Idea Description..

Algorithm	Description
WLB-ACO	In WLB-ACO, the ant colony optimization method is used to assign tasks to worker agents in parallel in order to make correct decisions for scheduling tasks.
HETS	In HETS, the prioritization is done by calculating the edge priority as well as the node priority. HETS algorithm selects the task after all its incoming edges are scheduled.
Full-local	Full-local centralize the training tasks to the local server.
SDFM	SDFM learns the social characteristics of vehicles in a distributed manner, and then transmits messages in a "storage-carry-forward" mode.
Random Walk	The Random Walk model is used to study the task allocation problem in edge computing.
Bulle Rap	Bubble rap is a social-based forwarding algorithm that uses the centrality and community of the real human flow trajectory to enhance delivery performance.

The above simulation experiments are all carried out on the windows 10 operating system, and the final experimental results are displayed by the origin drawing tool. In addition, for the convenience of viewing, we summarize the relevant parameter settings of the simulation experiment in TABLE III.

TABLE III: Simulation experiment parameter setting.

Parameter name	Value
Memory required for training(GB)	[1,4]
CPU cycles required for training(MHZ)	50
Training delay(ms)	[200, 1500]
Edge device CPU frequency(GHZ)	3
Wireless channel bandwidth(MHZ)	5
Number of optional edge devices	5
Learning factor	0.5

A. Edge device pre-selection algorithm based on reinforcement learning

Experiment 1: Fig.5 indicates the change in load balance of edge devices when the available resources are different. As revealed in the Fig.5, with the increase of available resources, except for the Full-Local algorithm, the load balance of the other three algorithms gradually decreases. The main reason for this result is that Full-loacl places the entire training task on the central server. Therefore, as time goes by, the load balance of the entire federated learning network will not decrease with the increase of available resources. In addition, because RL-SAP comprehensively considers the three attributes of the edge device's CPU, bandwidth and memory, and regardless of whether the edge device is overloaded or not, RL-SAP will still select the edge device with the lowest load balance

in the entire training process to participate in the training. Therefore, compared with the WLB-ACO algorithm and the HETS algorithm, the RL-SAP algorithm has always been at a lower degree of load balance. Through the analysis of specific data, the load balance of RL-SAP is reduced by 28.2% compared with WLB-ACO and 28.7% compared with HETS.

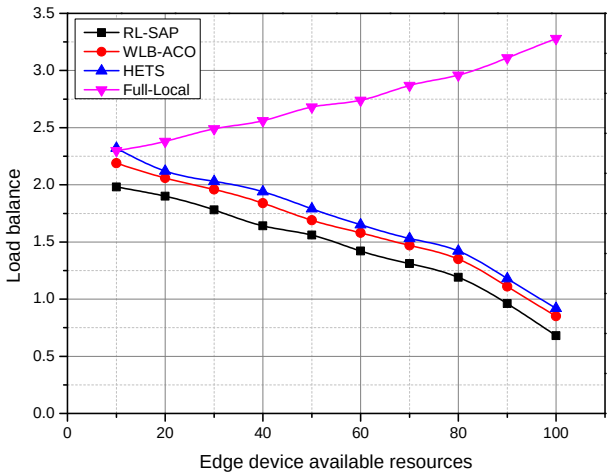

Fig. 5: Load balance under different computing resources.

In addition, in order to verify the performance of the RL-SAP algorithm when the available resources and model size are constant, we use different training models as experimental factors to observe the load balance of the four algorithms when facing different training tasks. The experimental results are shown in Fig.6. It can be clearly seen from the Fig.6 that the RL-SAP algorithm can still maintain a low load balance when facing different training tasks. This is because RL-SAP takes minimizing the load balance as the training goal, so it can always find the strategy that minimizes the load balance of the entire federated learning network. In addition, the load balance of the Full-Local algorithm does not increase with the change of the model, because the load balance of the Full-Local algorithm depends on the resources required for the training task.

Experiment 2: Fig.7 indicates the time required for the training process under different available resources. In this experiment, the model we trained is the MNIST dataset carried by tensorflow, which contains 50,000 training images. As revealed in the Fig.7, because the RL-SAP algorithm first uses the RL algorithm to consider the load balance of the edge device, and then uses the social attribute perception algorithm to ensure the reliability of the edge device. Therefore, when the RL-SAP algorithm is used to schedule the federated learning task, there will be no edge device overload or withdrawal from training, and the time required for federated learning is therefore reduced. Through the analysis of specific data, the speed of the RL-SAP algorithm proposed in this article is 10.3% higher than that of the WLB-ACO algorithm, and 14.5% higher than that of Full-local, but there is no significant improvement compared to the HETS algorithm.

Similar to the experimental purpose in Fig.6, in order to verify the time performance of the RL-SAP algorithm in

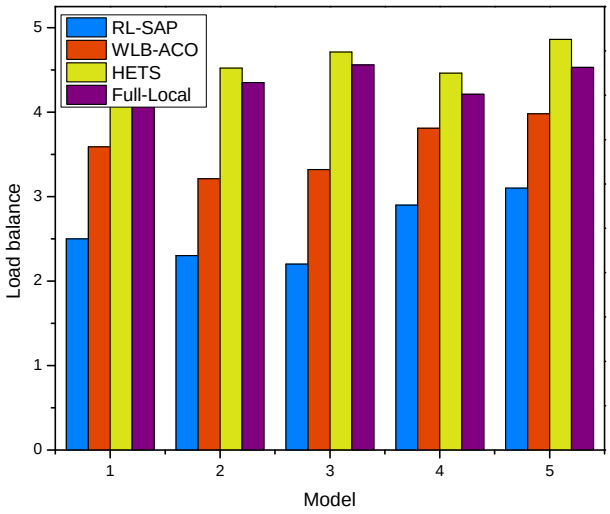

Fig. 6: Load balance in different scenarios.

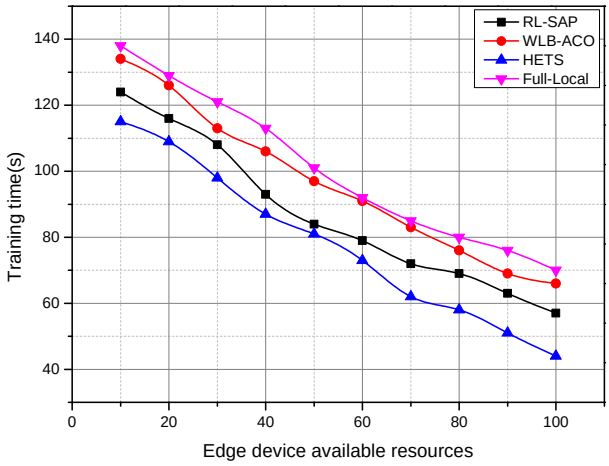

Fig. 7: Training time under different computing resources.

the face of different training tasks, we use different training models as experimental factors. The experimental results are shown in Fig.8. Obviously, RL-SAP can maintain better time performance when facing different training tasks. The reason for this result is the same as that shown in Fig.7. Because the load of the edge device and the frequency of interaction with the central server are considered comprehensively, RL-SAP can complete the training task faster.

B. Edge device pre-selection algorithm based on reinforcement learning

Experiment 3: Fig.9 describes the relationship between the number of different edge devices and the average training completion delay. As displayed in the Fig.9, the average training completion delay of the four algorithms shows a downward trend with the increase of edge devices. Among them, the edge device selection scheme based on social attribute perception proposed in this paper is always at a low level. Compared with the Random Walk algorithm, the solution in this article has an average performance when there are fewer devices involved in training. However, as the number of devices increases, RL-

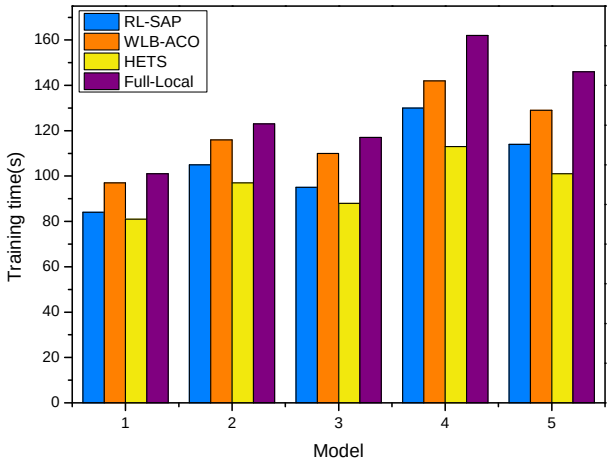

Fig. 8: Training time in different scenarios.

SAP performs better, while Random Walk gradually stabilizes. This is mainly because RL-SAP considers the load balance of the edge device while ensuring the frequency and intimacy between it and the central server. Therefore, the edge device participating in the training process is more reliable and the required communication delay is smaller. The other three algorithms may take a long time to complete the training due to the overload of the edge device. Compared with the SDFM algorithm and BubbleRap algorithm, he average training completion time has been reduced by 18% and 25% respectively.

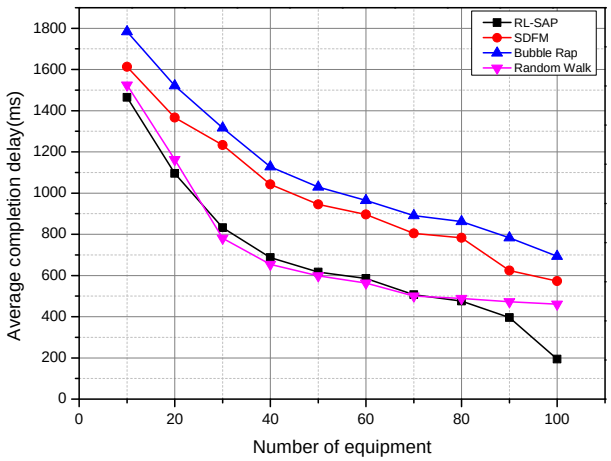

Fig. 9: Average completion delay under different number of devices.

Experiment 4: In addition, we also verify the effectiveness of the algorithm from the accuracy of the training results, and compare the federated learning algorithm that does not consider the reliability of the equipment participating in the training and the Reliable-FL algorithm proposed in [29]. As shown in Fig.10, the scheme proposed in this paper improves the accuracy of training results by 5.7% compared to the FL algorithm without considering the reliability of the equipment, and improves by 0.9% compared with Reliable-FL. However, the overall training accuracy is slightly insufficient compared to the FedCS algorithm. This is mainly due to the consideration of the activity of the edge device and the intimacy with the central server, which ensures the reliability of the equipment participating in the training, and to a large extent avoids the problem that the edge device cannot continue to iteratively update the training model with the central server. Therefore, the accuracy of training is greatly improved. Compared with the Reliable-FL algorithm, because the algorithm only considers the reliability of the edge device training model data and lacks consideration of iterative interaction, the accuracy is relatively poor.

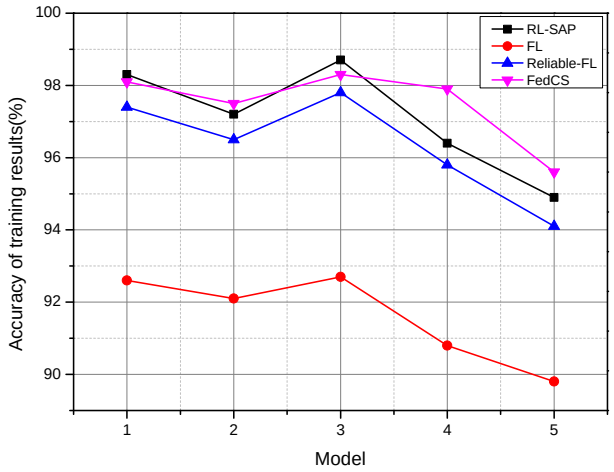

Fig. 10: Training accuracy of different models.

VI. Conclusion

Although federated learning provides a good solution to the problem of user privacy leakage in the process of machine learning. However, in the process of training the model, there is a lack of consideration of the reliability of the edge devices participating in the training, resulting in long training time, low efficiency, and insufficient training accuracy. In response to this problem, this paper proposes an edge device selection scheme based on reinforcement learning and social attribute perception, and the feasibility of the scheme is verified from multiple angles of the network load balance of the training process, the time required for training, the average training completion delay, and the accuracy of the training results. The results demonstrate that this scheme can effectively improve the efficiency of model training and the accuracy of training results. However, this solution is still insufficient in consideration of the training completion delay. Therefore, future work will focus on ensuring the reliability of the edge devices participating in the training, while studying how to reduce the training completion delay and further improve the efficiency of model training.

References

[1] Haipeng Yao, Pengcheng Gao, Jingjing Wang, Peiying Zhang, and Zhu Han. Capsule network assisted iot traffic classification mechanism for smart cities. *IEEE Internet of Things Journal*, 6(5):7515–7525, 2019.

[2] Xiaodong Ren, Gagangeet Singh Aujla, Anish Jindal, Ranbir Singh Batth, and Peiying Zhang. Adaptive recovery mechanism for sdn controllers in edge-cloud supported fintech applications. *IEEE Internet of Things Journal*, PP(99):1–1, 2021.

[3] Sudip Misra and Subhadeep Sarkar. Priority-based time-slot allocation in wireless body area networks during medical emergency situations: An evolutionary game-theoretic perspective. *IEEE J. Biomed. Health Informatics*, 19(2):541–548, 2015.

[4] Sudip Misra, Soumen Moulik, and Han-Chieh Chao. A cooperative bargaining solution for priority-based data-rate tuning in a wireless body area network. *IEEE Trans. Wirel. Commun.*, 14(5):2769–2777, 2015.

[5] Sudip Misra, Vivek Tiwari, and Mohammad S. Obaidat. Lacas: learning automata-based congestion avoidance scheme for healthcare wireless sensor networks. *IEEE J. Sel. Areas Commun.*, 27(4):466–479, 2009.

[6] Bhaskar Das, Sudip Misra, and Utpal Roy. Coalition formation for cooperative service-based message sharing in vehicular ad hoc networks. *IEEE Trans. Parallel Distributed Syst.*, 27(1):144–156, 2016.

[7] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.*, 37(3):50–60, 2020.

[8] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):1–19, 2019.

[9] Advances and open problems in federated learning. 2019.

[10] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Commun. Surv. Tutorials*, 22(3):2031–2063, 2020.

[11] Haipeng Yao, Danyang Fu, Peiying Zhang, Maozhen Li, and Yunjie Liu. Msml: A novel multi-level semi-supervised machine learning framework for intrusion detection system. *IEEE Internet of Things Journal*, 2018.

[12] Neeraj Kumar, Jatinder Manhas, and Vinod Sharma. A comparative analysis to visualize the behavior of different machine learning algorithms for normalized and un-normalized data in predicting alzheimer's disease. *Journal of Computational & Theoretical Nanoence*, 16(9), 2019.

[13] Chaman Verma, Veronika Stoffova, Zoltán Illés, Sudeep Tanwar, and Neeraj Kumar. Machine learning-based student's native place identification for real-time. *IEEE Access*, 8(99):1–15, 2020.

[14] Peiying Zhang, Chunxiao Jiang, Xue Pang, and Yi Qian. Stec-iot: A security tactic by virtualizing edge computing on iot. *IEEE Internet of Things Journal*, 8(4):2459–2467, 2021.

[15] Peiying Zhang, Xue Pang, Neeraj Kumar, Gagangeet Singh Aujla, and Haotong Cao. A reliable data-transmission mechanism using blockchain in edge computing scenarios. *IEEE Internet of Things Journal*, PP(99):1–1, 2020.

[16] H. Brendan Mcmahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. 2016.

[17] Sheng Shen, Tianqing Zhu, Di Wu, Wei Wang, and Wanlei Zhou. From distributed machine learning to federated learning: In the view of data privacy and security. *CoRR*, abs/2010.09258, 2020.

[18] Alberto Blanco-Justicia, Josep Domingo-Ferrer, Sergio Martínez, David Sánchez, Adrian Flanagan, and Kuan Eeik Tan. Achieving security and privacy in federated learning systems: Survey, research challenges and future directions. *CoRR*, abs/2012.06810, 2020.

[19] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. Verifynet: Secure and verifiable federated learning. *IEEE Trans. Inf. Forensics Secur.*, 15:911–926, 2020.

[20] Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, and Heiko Ludwig. Hybridalpha: An efficient approach for privacy-preserving federated learning. In *the 12th ACM Workshop*, 2019.

[21] Chen Xiaojun Zeng Shuai Dong Ye, Hou Wei. Efficient and secure federated learning based on secret sharing and gradients selection. *Journal of Computer Research and Development*, 57(10):2241, 2020.

[22] Meng Hao, Hongwei Li, Xizhao Luo, Guowen Xu, Haomiao Yang, and Sen Liu. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Trans. Ind. Informatics*, 16(10):6532–6542, 2020.

[23] Dipankar Sarkar, Sumit Rai, and Ankur Narang. Catfedavg: Optimising communication-efficiency and classification accuracy in federated learning. *CoRR*, abs/2011.07229, 2020.

[24] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-i.i.d. data. *IEEE Trans. Neural Networks Learn. Syst.*, 31(9):3400–3413, 2020.

[25] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. *arXiv*, 2019.

[26] Hangyu Zhu and Yaochu Jin. Multi-objective evolutionary federated learning. *IEEE Trans. Neural Networks Learn. Syst.*, 31(4):1310–1322, 2020.

[27] Xiaofei Wang, Yiwen Han, Chenyang Wang, Qiyang Zhao, Xu Chen, and Min Chen. In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. 2018.

[28] Andrew Hard, Kanishka Rao, Rajiv Mathews, Franoise Beaufays, and Daniel Ramage. Federated learning for mobile keyboard prediction. 2018.

[29] Jiawen Kang, Zehui Xiong, Dusit Niyato, Yuze Zou, and Mohsen Guizani. Reliable federated learning for mobile networks. *IEEE Wireless Communications*, 2019.

[30] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. 2018.

[31] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, AISec'19, page 1–11, New York, NY, USA, 2019. Association for Computing Machinery.

[32] Jie Lei, Qiao Luan, Xinhui Song, Xiao Liu, Dapeng Tao, and Mingli Song. Action parsing-driven video summarization based on reinforcement learning. *IEEE Trans. Circuits Syst. Video Technol.*, 29(7):2126–2137, 2019.

[33] Chen Zhong, Mustafa Cenk Gursoy, and Senem Velipasalar. Deep reinforcement learning-based edge caching in wireless networks. *IEEE Trans. Cogn. Commun. Netw.*, 6(1):48–61, 2020.

[34] Vincent Franois-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *Foundations and trends in machine learning*, 11(3-4):1,3–15,17–25,27–55,57–69,71–77,79–99,101–105,107–111,113–145, 2018.

[35] Godfrey Kibalya, Joan Serrat, Juan Luis Gorricho, Dorothy Okello, and Peiying Zhang. A deep reinforcement learning-based algorithm for reliability-aware multi-domain service deployment in smart ecosystems. *Neural Computing and Applications*, pages 1–23, 2020.

[36] Haipeng Yao, Bo Zhang, Peiying Zhang, Sheng Wu, and Song Guo. Rdam: A reinforcement learning based dynamic attribute matrix representation for virtual network embedding. *IEEE Transactions on Emerging Topics in Computing*, PP(99):1–1, 2018.

[37] Peiying Zhang, Chao Wang, Chunxiao Jiang, and Zhu Han. Deep reinforcement learning assisted federated learning algorithm for data management of iiot. *IEEE Transactions on Industrial Informatics*, PP(99):1–1, 2021.

[38] Xiang Wang, Supeng Leng, and Kun Yang. Social-aware edge caching in fog radio access networks. *IEEE Access*, 5:8492–8501, 2017.

[39] Liudong Chen, Nian Liu, Chenchen Li, and Jianhui Wang. Peer-to-peer energy sharing with social attributes: A stochastic leader-follower game approach. *IEEE Trans. Ind. Informatics*, 17(4):2545–2556, 2021.

[40] Xiaolin Fang, Junzhou Luo, Guangchun Luo, Weiwei Wu, Zhipeng Cai, and Yi Pan. Big data transmission in industrial iot systems with small capacitor supplying energy. *IEEE Trans. Ind. Informatics*, 15(4):2360–2371, 2019.

[41] Younes Hajoui, Omar Bouattane, Mohamed Youssfi, and Elhocein Illoussamen. New load balancing framework based on mobile agent and ant-colony optimization technique. In *New load balancing Framework based on mobile AGENT and ant-colony optimization technique*, 2017.

[42] Anum Masood, Ehsan Ullah Munir, M. Mustafa Rafique, and Samee Ullah Khan. Hets: Heterogeneous edge and task scheduling algorithm for heterogeneous computing systems. In *IEEE International Conference on IEEE International Conference on High Performance Computing and Communications, IEEE International Symposium on Cyberspace Safety and Security*, 2015.

[43] Rui Tian, Zhenzhen Jiao, Guiyun Bian, Zhiqing Huang, and Yibin Hou. A social-based data forwarding mechanism for v2v communication in vanets. In *International Conference on Communications and Networking in China*, 2016.

[44] Jie Zhang, Hongzhi Guo, and Jiajia Liu. Adaptive task offloading in vehicular edge computing networks: a reinforcement learning based scheme. *Mob. Networks Appl.*, 25(5):1736–1745, 2020.

[45] Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble rap: Social-based forwarding in delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 10(11):1576–1589, 2011.