Users Collaborative Computing for Hierarchical Federated Learning Based on Incentive Mechanism

Bei Zhuang[1(✉)], Shangjing Lin[1], Yueying Li[1], Ji Ma[2], Jin Tian[2], Chunhong Zhang[3], and Zheng Hu[3]

[1] School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing, China
{zhuangb,linshangjing,yyingli}@bupt.edu.cn
[2] School of Network and Communication Engineering, Jinling Institute of Technology, Nanjing, China
{Maji,jim.tian}@jit.edu.cn
[3] School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China
zhangch.bupt.001@gmail.com, huzheng@bupt.edu.cn

Abstract. Nowadays, Federated Learning (FL) is widely applied in the Internet of Things (IoT). However, when a large number of devices participate in FL, they still face the challenge of low communication efficiency. In addition, how to reasonably allocate FL trained models to third parties (e.g. task publishers) is also a problem that needs to be solved. In this article, we propose a hierarchical FL (HFL) framework based on incentive mechanisms, where task publishers mobilize users for collaborative computing through edge servers. At the lower layer, evolutionary game is used to model the dynamic decision-making process of users with bounded rationality, and users select user groups (UGs) to participate in training by considering model accuracy and training costs. At the upper layer, an iterative double auction mechanism is adopted to allocate the model reasonably to multiple task publishers, maximizing the total social welfare. Finally, the effectiveness of the proposed scheme is verified through experiments.

Keywords: Federated Learning · Evolutionary Game · Double Auction · Internet of Things · Hierarchical

1 Introduction

In recent years, Artificial Intelligence (AI) based model training methods mainly use the traditional cloud-based centralized learning framework [1–3]. However, this method of transmitting training data to the cloud server for processing has a series of problems, including transmission quality, data privacy and high consumption. Therefore, edge computing [4] is proposed to improve these problems,

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2025
Published by Springer Nature Switzerland AG 2025. All Rights Reserved
X. Chen et al. (Eds.): IoTaaS 2023, LNICST 585, pp. 239–254, 2025.
https://doi.org/10.1007/978-3-031-70507-6_18

and data processing and other tasks are placed on the edge server closer to the device. Federated Learning (FL) [5], as a distributed framework, promotes the development of edge computing. Users are allowed to save data locally, and only upload training model parameters to the edge server. At present, FL has been widely used in the Internet of Things (IoT), such as intelligent object detection [6], data sharing [7] and autonomous vehicle (AV) [8].

Service providers with cloud servers need to act as task publishers to mobilize users to participate in FL to jointly train an AI model, and provide users with services such as target recognition and content recommendation[9]. Due to distance limitations, task publishers are unable to directly mobilize users, so they need to collaborate with edge servers close to the user end for training. However, participants in FL are selfish and unwilling to contribute their private data and resources to participate in training. Paper [10] uses evolutionary game to consider data size and privacy costs to motivate users to participate and optimize resource allocation. However, in the case of a large number of users participating in FL, parameter interaction between users and edge servers can easily lead to communication congestion, and reducing the number of users can reduce the performance of the training model. How to balance model performance with communication load faces challenges. In addition, how to reasonably allocate the AI model trained on edge servers to numerous task publishers is also an urgent problem to be solved.

In response to the above issues, we propose a hierarchical federated learning (HFL) framework based on incentive mechanisms, which incentivizes users to efficiently collaborate and allocate AI models reasonably. Specifically, the contributions of this article can be summarized as follows: 1) An HFL framework based on incentive mechanism is proposed. In the lower layer, evolutionary game is used to model the process of users' autonomous selection of edge servers. The replicator dynamic depicts the dynamic nature and bounded rationality of user decisions, so as to achieve the goal of balancing the accuracy of the model and the cost of communication and computing, and further proves the uniqueness and stability of the evolutionary game to achieve equilibrium. 2) In the upper layer, we use the iterative double auction mechanism to reasonably allocate the model to the task publisher. Double auction achieves a many-to-many competitive situation and maximizes social welfare. 3) The experiment verifies the performance of evolutionary game in a large number of user scenarios and the effectiveness of iterative double auction mechanisms.

2 System Model

We consider an HFL architecture, which is composed of the task publisher (TP) layer, the edge layer and the user layer, as shown in Fig. 1. TPs may be individuals or enterprises. The set of TPs is denoted by $\mathcal{P} = \{1, ..., p, ..., P\}$.

The user layer is composed of U devices with certain computing power, represented by the set $\mathcal{U} = \{1, ..., u, ..., U\}$. Each device, as a user, can use private data to train the local model. The edge layer includes K edge servers, such as

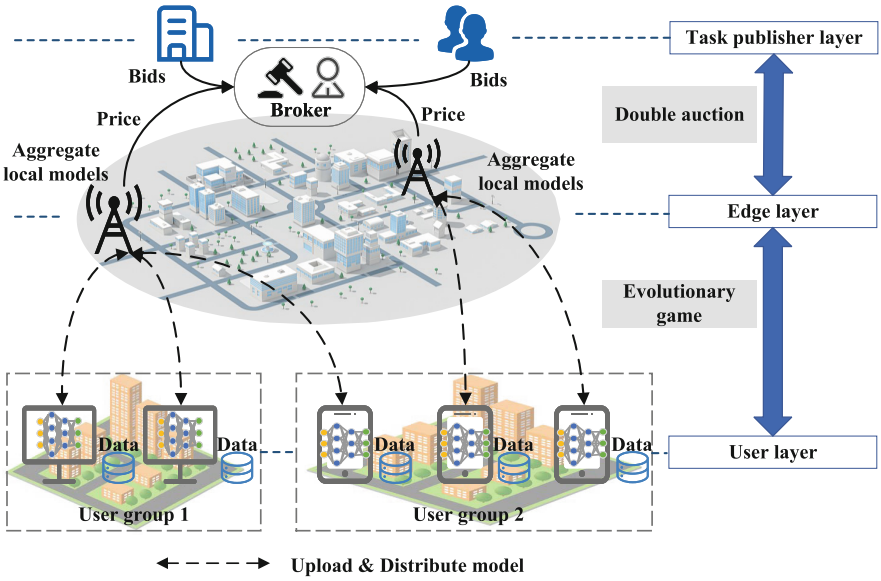

Fig. 1. Incentive mechanism based HFL framework for user collaborative computing

base stations, which receive and aggregate local models from the user layer, represented by the set $\mathcal{K} = \{1, ..., k, ..., K\}$. Users can choose to associate with any edge server and form a user group (UG) by selecting users from the same edge server. The edge server further distributes the aggregated model to each user of the corresponding UG.

Repeat the FL training process above until the maximum number of iterations T is reached. Assuming FL trains a classification model, the user updates the model parameters through the stochastic gradient descent (SGD) method, and uses the cross entropy loss function as follows:

$$F = -\frac{1}{H}\sum_{h=1}^{H}\sum_{m=1}^{M} y_{hm}\log(s_{hm}), \tag{1}$$

where H is the total number of training samples, M is the number of categories. y_{hm} is the true value, $y_{hm} \in 0, 1$, and when the true categories of sample h is m, $y_{hm} = 1$, otherwise $y_{hm} = 0$. s_{hm} is the prediction probability that sample h belongs to category m.

The lower layer is an evolutionary game process. In the user layer, users randomly select edge servers to form initial UGs. Edge servers need to mobilize more users to join their UGs collaborative training model to improve model accuracy. However, a large number of users choosing the same edge server can easily lead to communication congestion. In order to motivate users, balance model accuracy and communication and computing costs, we model the process of users' dynamic selection of UG to join as an evolutionary game. Users tend to

join UG that can obtain higher returns. After each round of FL training, users adjust their strategies.

The upper layer is a double auction process. The lower layer provides the trained model for the upper level, and the TPs select the model based on their quality. Considering the competition among TPs, an iterative double auction mechanism is adopted, in which P TPs bid for K models of the edge servers. Facilitate iterative interaction between multiple buyers and multiple sellers and adjust bids and pricing through the broker.

3 Lower-Layer Evolutionary Game

3.1 Evolutionary Game Formulation

In this section, we model the dynamic process of users' selecting UG participating in training as the evolutionary game. Evolutionary game includes four basic elements: population, pay off function, dynamics and equilibrium.

Population. We consider modeling evolutionary game among individuals of a single population. All users sets $\mathcal{U}$ participating in FL training are participants in evolutionary game. We divide all users in the population into multiple UGs, and the UG set is expressed as $\mathcal{J} = \{1, ..., j, ..., J\}$, a total of J. Among them, the number of users in UG $j \in \mathcal{J}$ is $n^j = x^j U$, $x^j \in [0, 1]$ and $\sum_{j=1}^{J} x^j = 1$. x^j is the proportion of the users in UG j in the whole population. In other words, users of all UGs form a complete population.

Pay Off Function. The pay off function refers to the expected profit obtained by the users choosing to join a UG, that is, the fitness function. It is related to the UG selected by the users and the proportion distribution of current different UGs. We define the fitness function as the difference between the benefits of joining the UG and the communication and computing costs of participating in the training. Relevant contents are further discussed in Sect. 3.2

Dynamics. The learning and imitation process of users is dynamic, and the game is iterative. In each iteration, the user dynamically chooses to exit a UG and join other UGs or stay in the current UG to maximize its profit. The users' strategy is to select a certain UG to join.

Equilibrium. The result of evolutionary game is to reach a convergent stable state, that is, the UG proportion converges to a point, which is unique and stable. The final UGs state is expressed as vector $\mathbf{x}^* = \left[x^{1*}, ..., x^{j*}, ..., x^{J*}\right]$, and x^{j*} refers to the proportion of users who choose UG j.

3.2 Dynamic Process Modeling

In this section, the fitness function we defined will be described in detail.

The proportion of UG j in the whole population changes with the user adjustment strategy of each iteration, expressed as $x^j(t)$. The fitness of users after joining UG j and participating in FL training at time t is:

$$A^j\left(x^j(t)\right) = \mathcal{L}\left(R^j\left(x^j(t)\right) - E^j_{total}\left(x^j(t)\right)\right), \tag{2}$$

where we assume $\mathcal{L}(\cdot)$ to be a linear utility function. It is a function related to $x^j(t)$. $R^j\left(x^j(t)\right)$ is the accuracy function of FL training model, which is used to express the benefits obtained by users participating in UG j. $E^j_{total}\left(x^j(t)\right)$ is the communication and computing costs generated in training.

In the FL scenario of this article, the more users, the more data they contribute, and the higher the accuracy of the model. According to [11], the relationship between the number of users and accuracy can be expressed as follows:

$$R^j\left(x^j(t)\right) = p\log\left(Ux^j(t)\right) + q. \tag{3}$$

We use the total size of parameters that users in the UG j need to transmit to the edge server after each iteration as the communication cost to represent the communication congestion level of the UG j. The communication cost required for the *t-th* iteration of UG j is defined as follows:

$$C^j_{com}\left(x^j(t)\right) = |w|Ux^j(t), \tag{4}$$

where $|w|$ is the size of the model parameter. In the computation phase of FL, each user trains the local model on local data D_i. The computing energy consumption, i.e. the computing cost, for all users in the UG j is represented as

$$C^j_{cmp} = \sum_{i\in j} \xi_i D_i f_i^2. \tag{5}$$

The number of CPU cycles of the user i needed to execute one unit of data is denoted as ξ_i, and the CPU cycle frequency is denoted as f_i. Therefore, the total cost $E^j\left(x^j(t)\right)$ is defined as follows:

$$E^j_{total}\left(x^j(t)\right) = C^j_{com}\left(x^j(t)\right) + C^j_{cmp}. \tag{6}$$

Further, we can get the average fitness of all UGs for the *t-th* iteration as follows:

$$\overline{A}\left(x^j(t)\right) = \frac{1}{J}\sum_{j=1}^{J} A^j\left(x^j(t)\right). \tag{7}$$

In the game, users can exchange the fitness value that can be obtained by choosing different UGs, that is, the information of profit, so as to compare with the current UG. By comparison, users will be more inclined to join UG with

higher profit. Due to the replicator dynamics being applicable to a large population of members with slow learning speeds for random selection in iterative games, we define replicator dynamics to capture and model the dynamic process of user's selecting UG, as follows:

$$\dot{x}^j(t) = f^j\left(\mathbf{x}(t)\right) = \alpha x^j(t)\left(A^j\left(x^j(t)\right) - \bar{A}\left(x^j(t)\right)\right), \tag{8}$$

where α is the positive learning rate of UG, which controls the speed of user adaptation strategy. When the population size is large, the user needs more time to transfer and obtain the status and information between UGs, and the learning speed is often slow.

Considering the reality, when users are large, they cannot learn global information in a short time (the number of every UG's users and the self conditions of other users, etc.). Therefore, the user is an individual with finite rationality, they play the game repeatedly with only limited information (its own conditions, the fitness of the current UG and the average fitness of all UGs). Specifically, when the fitness of the current UG is lower than the average fitness, the probability of users in the current UG joining other UGs will be increased. At a specific time t, $\dot{x}^j(t) = 0$, $\forall j \in \mathcal{J}$, and then it reaches evolutionary equilibrium, that is, all users get the same pay off, and there is no need to adjust the strategy.

The game among users will eventually evolve to a unique and stable equilibrium point. Stability refers to keeping $\dot{x}^j(t) = 0$ in all time periods after a specific time t. Uniqueness refers to the evolution from any initial proportion to the same equilibrium point. Due to the difference of users conditions and the uncertainty of adjusting strategy based on probability, $\dot{x}^j(t)$ will fluctuate in a small range around 0. Therefore, we define the threshold γ. When $\dot{x}^j(t)$ is less than γ, we consider that the evolutionary game reaches equilibrium.

3.3 Proof of Evolutionary Equilibrium

In this section, we prove the existence, uniqueness and stability of evolutionary equilibrium solution.

Existence. Firstly, we prove the boundedness of (8) in Lemma 1.

Lemma 1. *The first order derivatives of* $f^j\left(\mathbf{x}(t)\right)$ *with respect to* $x^q(t)$ *is bounded for* $\forall q \in \mathcal{J}$.

Proof. For ease of presentation, we omit the notations of t in the proof. The first order derivative of $f^j(\mathbf{x})$ with respect to x^q, $\forall q \in \mathcal{J}$, is given by

$$\frac{df^j(\mathbf{x})}{x^q} = \alpha\left[\frac{dx^j}{dx^q}\left(A^j - \overline{A}\right) + x^j\left(\frac{dA^j}{dx^q} - \frac{d\overline{A}}{dx^q}\right)\right]. \tag{9}$$

And then, the $\frac{dA^j}{dx^q}$ is as follows:

$$\frac{dA^j}{dx^q} = \frac{p}{Ux^j}\frac{dx^j}{dx^q} - |w|U\frac{dx^j}{dx^q}. \tag{10}$$

Obviously, $\frac{dA^j}{dx^q}$ and $\frac{d\overline{A}}{dx^q}$ are bounded $\forall q \in \mathcal{J}$, which represents that $|\frac{df^j(\mathbf{x})}{x^q}|$ is bounded. This proof also applies to all time periods.

Uniqueness. Secondly, we prove the uniqueness of (8) in Theorem 1.

Theorem 1. *For any initial condition* $\mathbf{x}(0)$*, there exists a unique evolutionary equilibrium to the dynamics defined in (8).*

Proof. According to Lemma 1, we have proven that the $f^j(\mathbf{x}(t))$ is bounded and continuously differentiable. Thus, the maximum absolute value of its partial derivative is a Lipschitz constant. According to the Mean Value Theorem, there exists a constant c between $x_1(t)$ and $x_2(t)$ such that $\frac{|f^j(x_1(t))-f^j(x_2(t))|}{(x_1(t)-x_2(t))} = \frac{df^j(c)}{dx^q}$. So that we can define the relation

$$|f^j(x_1(t)) - f^j(x_2(t))| \leq \Gamma\,|x_1(t) - x_2(t)|, \tag{11}$$

where $\Gamma = max\left\{\frac{df^j(c)}{dx^q}\right\}$ and Any $x_1(t)$, $x_2(t)$ belongs to vector $\mathbf{x}$ for $\forall t$. Following the Lipschitz condition, it shows that for any initial value $\mathbf{x}(0)$, (8) has a unique solution x^{j*}.

Stability. Thirdly, we prove the stability of (8) in Theorem 2.

Theorem 2. *For any initial condition* $\mathbf{x}(0)$*, the evolutionary equilibrium to the dynamics in (8) is stable.*

Proof. We define the Lyapunov function

$$Q(\mathbf{x}(t)) = \left(\sum_{j=1}^{J} x^j(t)\right)^2, \tag{12}$$

which is positive definite since:

$$Q(\mathbf{x}(t)) \begin{cases} = 0, & \text{if } \mathbf{x}(t) = 0 \\ > 0, & \text{otherwise.} \end{cases} \tag{13}$$

Taking the first-order derivative with of $Q(\mathbf{x}(t))$ with respect to t,

$$\frac{dQ(\mathbf{x}(t))}{dt} = 2\left(\sum_{j=1}^{J} x^j(t)\right)\left(\sum_{j=1}^{J} \dot{x}^j(t)\right). \tag{14}$$

Because that at any point of time $\sum_{j=1}^{J} x^j(t) = 1$, the replicator dynamics have to equate to zero for this to make (14) equate to zero. Specifically,

$$\sum_{j=1}^{J} \dot{x}^j(t) = 0, \forall t. \tag{15}$$

$\frac{dQ(\mathbf{x}(t))}{dt} = 0$ satisfies the Lyapunov conditions required for stability, as defined in the Lyapunov's second method for stability [12].

From above, we prove the uniqueness and stability of evolutionary equilibrium. In Algorithm 1, we show the whole process of users selecting UGs to participate in FL training.

Algorithm 1. Users Dynamic Evolutionary Game

Require: Users and the UGs set $\mathcal{J} = \{1, ..., j, ..., J\}$;
Ensure: Evolutionary equilibrium solution $\mathbf{x}^*$;
Each user select a UG;
for $t = 1, 2, ..., T$ **do**
  Pay Off Computation
  for $j \in \mathcal{J}$ **do**
    Derive $A^j(x^j(t)) = R^j(x^j(t)) - E^j_{total}(x^j(t))$
  end for
  User Strategy Adjustment
  for $j \in \mathcal{J}$ **do**
    Derive $\dot{x}^j(t) = f^j(\mathbf{x}(t)) = \alpha x^j(t)(A^j(x^j(t)) - \bar{A}(t))$ and $x^j(t)$
  end for
end for
Derive $\mathbf{x}^* = \left[x^{1*}, ..., x^{j*}, ..., x^{J*}\right]$

4 Upper-Layer Double Auction

In the upper layer, a double auction mechanism is designed to encourage edge servers to actively response to the task publishers' requests and find out the balanced solution.

Specifically, the task requester, edge servers, and broker can be seen as an auction market, where the task requesters act as buyers who expect that the edge servers can contribute models with lower loss values. The model is provided by the lower layer, and the model loss value is calculated by (1). The loss that requester expects can be expressed as a loss vector $\boldsymbol{F}_p = \{F_{p,1}, F_{p,2}, ..., F_{p,K}\}$). The edge servers act as sellers who prefer to consume less resources to train models and the loss of models that the edge servers are willing to contribute can be

represented as $\boldsymbol{F}_k = \{F_{k,1}, F_{k,2}, ..., F_{k,P}\}$. We use a strictly concave and decreasing utility function $U_p(\boldsymbol{F}_p)$ to measure the task publishers' received revenue with respect to its request loss $\boldsymbol{F}_p$ in (16).

$$U_p(\boldsymbol{F}_p) = a \sum_{p=1}^{P} \log(b \frac{1}{F_{p,k}} + 1) \tag{16}$$

where log can catch the concave nature, and a indicates the level of the buyers' purchasing intention.

Similarly, a strictly convex utility function $C_k(\boldsymbol{F}_k)$ is used to measure edge servers' total cost, which indicates the sum of communication and computation energy consumption required to achieve model loss $\boldsymbol{F}_k$ for all users under the edge servers.

The task publishers and edge servers cannot independently decide the request loss and accept loss to reach an agreement because of their contradictory intentions. There is a need for a broker to help determine the equilibrium solution. From a social perspective, the optimal solution is when trading maximizes social welfare. The Social Welfare Optimization (SWO) problem is expressed as (17):

$$\max_{\boldsymbol{F}_p, \boldsymbol{F}_k} \sum_{p=1}^{P} U_p(\boldsymbol{F}_p) - \sum_{k=1}^{K} C_k(\boldsymbol{F}_k) \tag{17}$$

$$s.t. 0 \leq F_p \leq F^{max} \forall p \in \mathcal{P} \tag{17a}$$

$$0 \leq F_k \leq F^{max}, \forall k \in \mathcal{K} \tag{17b}$$

$$F_p = F_k, \forall p \in \mathcal{P}, \forall k \in \mathcal{K} \tag{17c}$$

Constraints (17a) and (17b) indicate that the loss value has to satisfy the tolerate value. (17c) indicates the requested loss should equal to the accepted loss when the deal is completed.

However, the optimal solutions $\boldsymbol{F}_p^*$ and $\boldsymbol{F}_k^*$ of the SWO problem cannot be solved directly because the utility functions of the task publishers and edge servers are confidential. An iterative double auction mechanism is designed to tackle the asymmetric information problem. The broker manages them to iteratively interact and adjust their bidding strategy. Assume the bids of buyers are $\boldsymbol{b}_p$ and the bids of sellers are $\boldsymbol{b}_k$, then the SWO problem can be converted to the form in (18) according to the BAP problem in paper [13]:

$$\max_{\boldsymbol{F}_p, \boldsymbol{F}_k} \sum_{p=1}^{P} \sum_{k=1}^{K} (\boldsymbol{b}_p \log \boldsymbol{F}_p - \frac{1}{2} \boldsymbol{b}_k \boldsymbol{F}_k^2) \tag{18}$$

$$s.t. 0 \leq F_p \leq F^{max} \forall p \in \mathcal{P} \tag{18a}$$

$$0 \leq F_k \leq F^{max}, \forall k \in \mathcal{K} \tag{18b}$$

$$F_p = F_k, \forall p \in \mathcal{P}, \forall k \in \mathcal{K} \tag{18c}$$

where constraints of (18) are the same as (17a)-(17c). The $\log \boldsymbol{F}_p$ and $\boldsymbol{F}_k^2$ are used to capture the concave nature of publishers' utility and convex nature of servers' utility.

According to (18), the relationship between the bids and the optimal loss solution can be obtained. If buyers and sellers can submit bids as above, the broker can get an optimal solution. The complete solution process is not described in detail here. We have referred to the classic iterative double auction paper [13].

5 Performance Evaluation

In this section, the performance of evolutionary game and double auction algorithms is mainly demonstrated.We assume that each user is assigned a computing task, and each user trains a CNN model on the local MNIST image dataset. In order to obtain the specific values of p and q in (3), we fitted the relationship between the number of users and model accuracy in the experimental scenario of this article. Each user is assigned 600 different images as a training set, and 1000 different images as a test set. The fitting process is shown in Fig. 2. The data for each point is the average obtained after 100 repeated experiments.

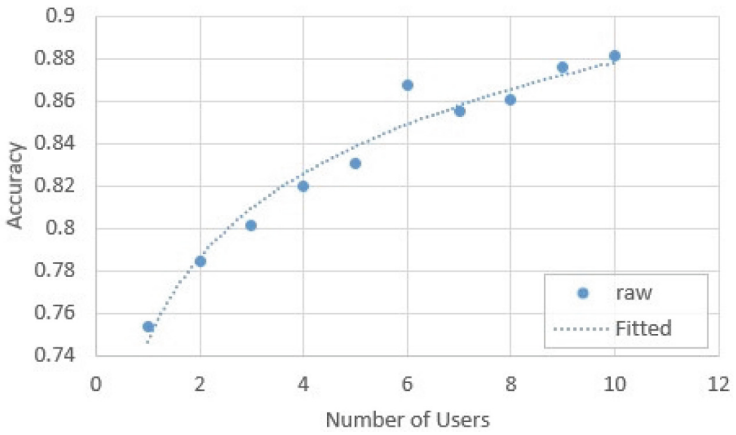

Fig. 2. Accuracy fitting

The fitting correlation coefficient $R^2 = 0.9153$, which is highly consistent. The relationship is as follows:

$$R^j\left(x^j(t)\right) = 0.0571 \log\left(U x^j(t)\right) + 0.7468. \tag{19}$$

Table 1. Simulation parameters

CPU Cycle (ξ_i)	5
Total number of users (U)	[50,75,100,125,150]
Parameters Size (w_i)	0.02 M
Local Data Size(D_i)	3.76 M
Threshold(γ)	0.001
Epoch	30
Batch Size (B)	128

All parameter settings in the experiment are shown in Table 1.

The comparison algorithm selects K-means[14] and spectral clustering algorithm [15]. Using the two clustering algorithms, users are divided into two groups based on their location, and their locations are randomly generated.

5.1 Performance of Evolutionary Game

The experiment sets all users to be divided into two UGs for dynamic evolutionary game, with strategy A selecting UG 1 and strategy B selecting UG 2.

Figure 3 shows the process of the evolution of strategy A proportion ultimately converging to the Nash equilibrium point under different initial strategy distributions. The initial proportion of selecting strategy B for a given population is 0.8, 0.7, and 0.8, respectively. The users in the population constantly change their own strategy choice, and eventually tend to be stable, and choose the strategy that is most beneficial to them. From the figure, it can be seen that under different initial ratios, the final convergence is $\mathbf{x}^* = [0.56, 0.44]$.

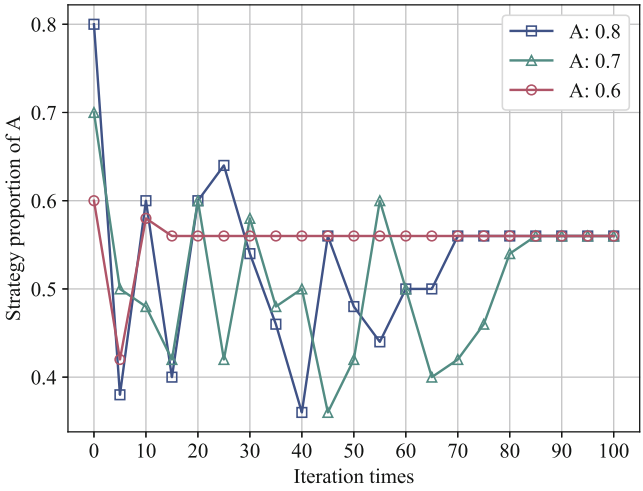

Fig. 3. Strategy proportion equilibrium with different initial proportion

Figure 4 analyzes the changes in strategy proportion under different user densities. Simulate scenarios with 50, 75, and 100 users at the same initial proportion. It can be seen that the more users involved, the slower the convergence speed. This is because the more users there are, the more time they need to adapt to the strategy. It can be seen that evolutionary games are suitable for a large number of user scenarios, and can reach stable states in different population sizes, and individuals can learn the optimal strategy.

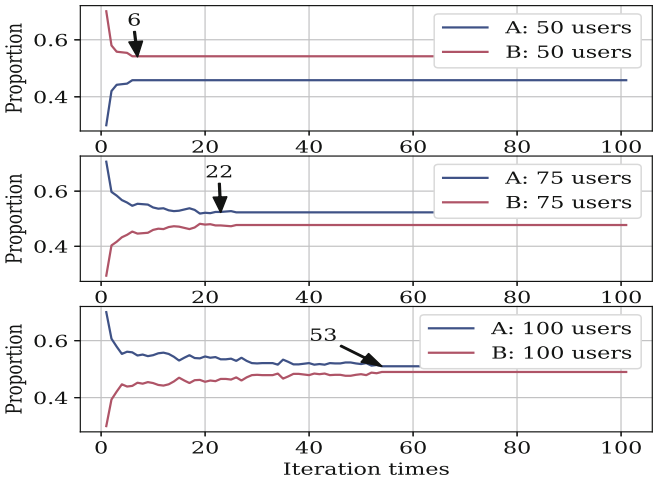

Fig. 4. Strategy proportion equilibrium with different number of users

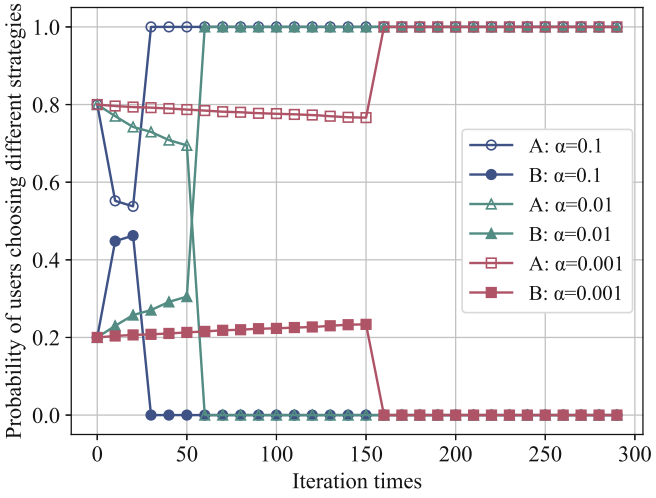

Fig. 5. Individual probability equilibrium with different learning rate

Figure 5 shows the probability changes of users choosing different strategies when different learning rate α are set under the same initial conditions. As can be seen, the larger α is, the faster the probability of choosing strategies A and B converges, that is, the faster the population adapts to the strategy, and the final convergence value is 0 or 1.

Figure 6 shows the changes in profit, accuracy, and cost corresponding to user choices A and B under different user densities. Among them, the profit is calculated by (2). It can be seen that as the number of users increases, both cost and accuracy are on the rise. The profit is gradually decreasing because as the number of users increases, the speed of accuracy increase is slower than the speed of cost increase. Therefore, in practical applications, the number of users participating in training should be selected based on the demand for accuracy and tolerance for cost.

Figure 7 shows a comparison of the total profits of different algorithms under different user numbers. Sum the benefits of different strategies to obtain the total population benefits. It can be seen that using the evolutionary game algorithm yields the highest total profits, and the more users there are, the more significant the effect, proving the effectiveness of the proposed evolutionary game algorithm. The K-means and spectral clustering algorithms can not balance the accuracy and cost when clustering, so as to maximize the interests of users and the whole.

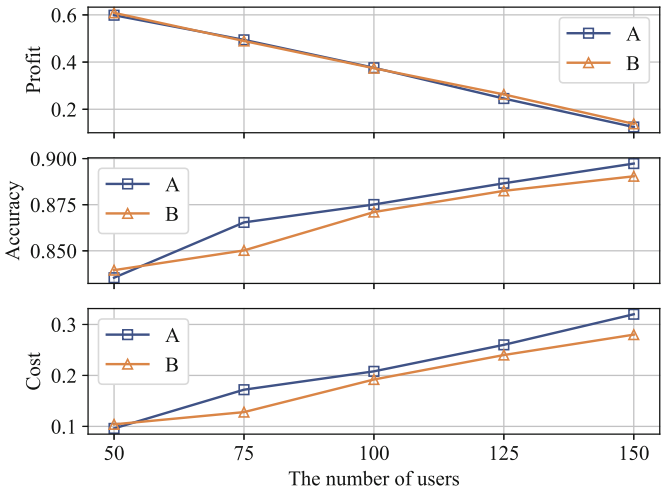

Fig. 6. Changes in profit, accuracy, and cost with different number of users

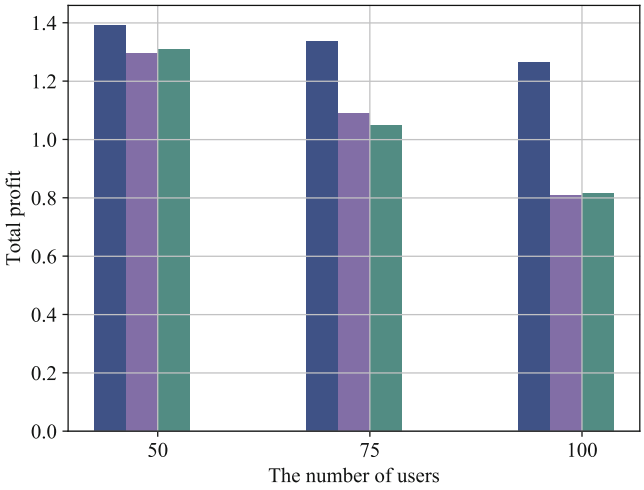

Fig. 7. Total profit

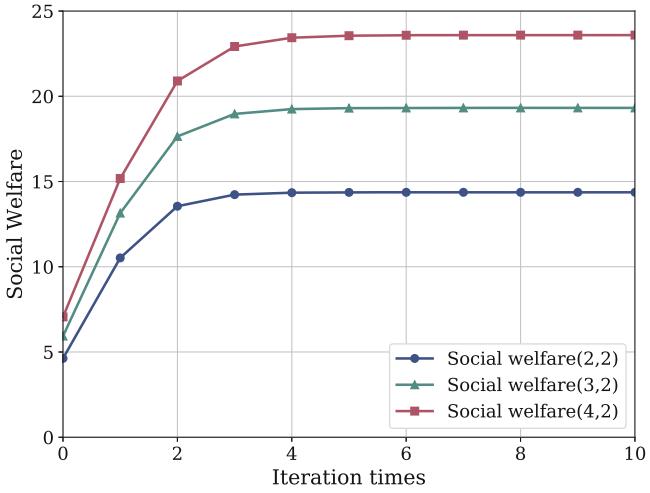

Fig. 8. Social welfare with different numbers of task publishers

5.2 Performance of Double Auction

In this part, we evaluate social welfare with the different numbers of buyer-seller matches (2 buyers-2 sellers, 3 buyers-2 sellers and 4 buyers-2 sellers). The buyers' purchasing intention factor a is set to 2. Fig. 8 shows that the total social welfare converges as the iteration number increases, and the welfare increases with the buyers increasing. This is because more members can contribute more external utility. The trend of the social welfare proves that the designed iterative double

auction mechanism can effectively incentivize transactions between publishers and edge servers and converge total market utility to the maximum value.

6 Conclusion

In this paper, we propose an HFL framework based on incentive mechanism consisting of multiple TPs, edge servers and users. We use the evolutionary game to assist users to make decisions by balancing model accuracy and training cost, avoiding communication congestion. Design iterative double auction algorithm between TPs and edge servers, allocate models reasonably, and achieve maximum social welfare. The simulation demonstrates the effectiveness of the proposed framework. In the future, we will consider more factors that affect user choices in our algorithms.

References

1. Hagos, D., Engelstad, P.E., Yazidi, A., Kure, Ø.: General TCP state inference model from passive measurements using machine learning techniques. IEEE Access **6**, 28372–28387 (2018)
2. Afify, A. A., Mokhtar, B.: Machine Learning-based Services Provisioning for Intelligent Internet of Vehicles. In: 2021 IEEE 7th World Forum on Internet of Things (WF-IoT), pp. 51–54. IEEE (2021)
3. Tang, F., Mao, B., Kato, N., Gui, G.: Comprehensive survey on machine learning in vehicular network: technology, applications and challenges. IEEE Commun. Surv. Tutor. **23**(3), 2027–2057 (2021)
4. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Int. Things J. **3**(5), 637–646 (2016)
5. McMahan, H. B., et al.: Communication-efficient learning of deep networks from decentralized data. In: the 20 th International Conference on Artificial Intel-ligence and Statistics (AISTATS), IEEE (2016)
6. Zhou, X., Liang, W., She, J., Yan, Z., Wang, K.I.-K.: Two-layer federated learning with heterogeneous model aggregation for 6G supported internet of vehicles. IEEE Trans. Veh. Technol. **70**(6), 5308–5317 (2021)
7. Lu, Y., Huang, X., Zhang, K., Maharjan, S., Zhang, Y.: Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles. IEEE Trans. Veh. Technol. **69**(4), 4298–4311 (2020)
8. Zhao, N., Wu, H., Yu, F.R., Wang, L., Zhang, W., Leung, V.C.M.: Deep-reinforcement-learning-based latency minimization in edge intelligence over vehicular networks. IEEE Int. Things J. **9**(2), 1300–1312 (2022)
9. Hammoud, A., Otrok, H., Mourad, A., Dziong, Z.: On demand fog federations for horizontal federated learning in IoV. IEEE Trans. Netw. Serv. Manage. **19**(3), 3062–3075 (2022)
10. Ng, J.S., et al.: A hierarchical incentive design toward motivating participation in coded federated learning. IEEE J. Sel. Areas Commun. **40**(1), 359–375 (2022)
11. Zou, Y., Feng, S., Xu, J., Gong, S., Niyato, D., Cheng, W.: Dynamic Games in Federated Learning Training Service Market. In: 2019 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), pp. 1–6. IEEE (2019)

12. Tsuneo, Y.: Lyapunov Stability Theory. Analysis and Control. MIT Press, Foundations of Robotics (2003)
13. Li, Z., Yang, Z., Xie, S.: Computing resource trading for edge-cloud-assisted internet of things. IEEE Trans. Indust. Inf. **15**(6), 3661–3669 (2019)
14. Tanir, D., Nuriyeva, F.: On selecting the Initial Cluster Centers in the K-means Algorithm. In: 2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT), pp. 1–5. IEEE (2017)
15. Lahmar, I., Zaier, A., Yahia, M. Bouallegue, R.: A New Self Adaptive Fuzzy Unsupervised Clustering Ensemble Based On Spectral Clustering. In: 2020 17th International Multi-Conference on Systems, Signals & Devices (SSD), pp. 1–5. IEEE (2020)