Attention-aided Federated Learning for Dependency-Aware Collaborative Task Allocation in Edge-Assisted Smart Grid Scenarios

Chenyang Wang*, Bosen Jia†, Hao Yu‡, Liandong Chen§, Kai Cheng§ and Xiaofei Wang*
*College of Intelligence and Computing, Tianjin University, Tianjin, China.
†Tianjin International Engineering Institute, Tianjin University, Tianjin, China
‡Information Technology and Electrical Engineering, University of Oulu, Oulu, Finland.
§Information & Telecommunication Branch,
State Grid HeBei Electric Power Company, Shijiazhuang, China.
{chenyangwang, bosenjia, xiaofeiwang}@tju.edu.cn,
hao.yu@oulu.fi, {Chenliandong1, smileck}@126.com

***Abstract*—With the significant improvement of the intelligent capabilities of smart devices accompanied by the increasingly high requirements. Edge computing is regarded as an effective solution to achieve rapid response by deploying applications and tasks close to users. However, many studies only consider complete offloading, or offload tasks to edge servers in any proportion when designing the allocation strategies, ignoring the dependencies between subtasks. To deal with the dynamic environment, some learning-based task allocation methods generally adopt a centralized training way, which leads to the excessive network transmission resource consumption, especially in the smart grid scenario. To tackle the aforementioned challenges, we investigate the collaborative task allocation (CTA) problem by jointly considering the difference between the benefit of the tasks execution under a certain allocation strategy and when all tasks are executed locally. In this paper, the objective is to maximize the system gain, and we propose an attention-aided federated learning algorithm to deal with the CTA problem, named AteFL, by learning a shared model and extracting the system context for better representing the network information. The simulation results also show the superiority of the proposed AteFL algorithm.**

I. Introduction

Recently, with the exceptional growth of smart devices and to pave the way for 5G or beyond 5G communication systems, promotes huge various of high-demanding services and applications, e.g., face recognition [1], sustainable energy supply of smart grid [2] and 3D games [3]. These applications and services have high requirement for data privacy, and are computation and latency sensitive constrained. To cope with the challenges mentioned above, some studies [4], [5] deploy the applications in cloud servers with the beneficial of rapid elasticity and on-demand resource pooling. However, transmitting massive data for application deployment from devices to the remote cloud center usually produce unpredictable latency and excessive network resource consumption. Currently, some studies [6], [7] elaborate data desensitization sharing mechanism, group feature sharing mode, etc. methods to solve the problem of "data island" in the smart grid scenario. However, the data desensitization sharing mechanism has security risks such as difficulty in matching users, easy derivation and backtracking, and even brings legal risks, and does not meet the requirements of external sharing of power data; while the group feature sharing model has slightly better privacy protection, but the features are limited and cannot match other data usage, the coverage and effectiveness of sharing are poor, and the application effect of data sharing is not desire for the system. Therefore, how to ensure the privacy and security of data and users in the multi-party data sharing of various units within the company, and on this basis to achieve data collaborative training, improve the effectiveness and accuracy of the data model, and achieve the effect of data security sharing, is an urgent need to solve question.

Federated learning (FL) [8] is a promising learning method which is collaborative training by multiple decentralized participants and share the training parameters rather than completing the whole training process at the central point. It enables to protect data security and user privacy as well as make full use of decentralized data to improve the performance of models without sharing the raw data.

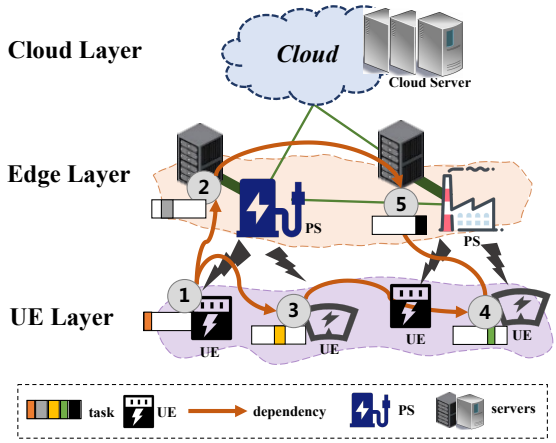


Fig. 1. Architecture Overview

Typically, an application is usually decoupled into multiple

tasks (e.g., electricity inspection or usage) that are deployed in smart grid network with the characteristics of low cost, flexibility and scalability, to achieve the fast response and dynamic deployment of various of and applications [9], [10]. Be aware of the dependencies between tasks, system can know where they are running, whether on a local user equipment (e.g., electricity meters), on edge servers (e.g., power station, PB), or on a remote cloud server. Generally, the dependency of tasks is model by directed acyclic graph (DAG), reflecting the order in which tasks are executed. For instance, the literature [11] discusses the dependent task offloading problem with constrains of limited predetermined task caching, the proposed DAG based model outperforms the other alternatives in term of applications' completion time. However, the studies mentioned above only conduct the deployment strategies under the homogeneous or single edge server environment. Furthermore, transfer a huge number of data to the centralized orchestrator for offline/online training which may lead to the privacy issue and overhead network transmission pressure. Particularly, in the smart grid scenario, the electricity data is sensitive, if the data is adopted to the centralized training way, the smart grid system may be maliciously attacked.

We explore how to design an efficient federated learning framework to protect user privacy while fully solving the unbalanced problem of the amount of local power data and sample feature distribution in smart grids. In this paper, we investigate an efficient decentralized collaborative task allocation strategy for edge computing, where a three-layer network architecture is considered, and the tasks are partitioned into different subtasks according to their inner-dependencies. As illustrated in Fig. 1, the arrow directions indicate that completing the former task is a prerequisite for starting the latter. Particularly, we model the inner-dependencies between the subtasks as DAG, and the attention mechanism is carried out to extract the information of both power UE and PB. Finally, we model the collaborative task allocation problem (CTA) as a Markov Decision Process (MDP), and the attention-aided federated learning algorithm is proposed to derive the optimal decision making. We summarize the main contributions of this paper in the following:

- We introduce the dependency of task as directed acyclic graph (DAG) making it more practical for deployment, the collaborative tasks deployment (CTA) problem is model by jointly considering the optimization of overall task accomplish latency and the energy consumption of UEs, which is proved as *NP-hard*.
- We extract the comprehensive network information by using an attention based task representation method, where system context of both infrastructure and task features is embedded by multi-head attention mechanism.
- We model the CTA problem as a Markov decision process (MDP) and propose an attention-aided federated learning algorithm, named AteFL, to subsequently solve the above problem. Simulation results also show the priorities of the proposed algorithms.

The rest of this article is organized as follows: Section II introduces the system mode and the optimization problem, the proposed algorithm is derived in Section III and the experimental simulation is conducted in Section IV. Finally, we conclude the article in Section V.

II. System Model and Problem Formulation

We consider $M = \{1, 2, ..., m\}$ user equipment consisting of electricity meters in the UE layer connect with $K = \{1, 2, ..., k\}$ edge servers deployed in the power stations (PSs) in Edge layer via cellular links, and a cloud center is deployed to orchestrate the system scheduling. Assume that there are $S = \{S_1, S_2, ..., S_n\}$ applications in the network given by the UEs, each application elaborates a partitioned task DAG, denoted as $S_n^D = \{\mathcal{V}_n, \mathcal{E}_n\}$, where $\mathcal{V}_n = \{v_n^i | i = 1, 2, ..., I\}$ and $\mathcal{E}_n = \{e_n^{ij} | i, j \in \{1, 2, ..., I\}, i \neq j\}$ represents the set of dependencies of tasks, *i.e.*, the precedence relation such that task v_n^i is completed before v_n^j starts.

Specifically, we define the entry task v_n^o when it has no precursor and the task without a successor is called exit task v_n^{I+1}. Each $v_n^i \in \mathbb{S}_n$ is associated with the two-tuples (c_n^i, d_n^i), where c_n^i is the required CPU cycles to finish v_n^i and d_n^i denotes the size of input task. Thus, we have $c_n^0 = c_n^{I+1} = 0$ ensuring the place where task S_n starts and ends.

We employ an allocation variable $\alpha_n^{i,l} = \{0, 1\}$, $l \in \{M, K\}$, e.g., when $l = k$ means that v_n^i is deployed on edge server k, $\alpha_n^{i,l} = 1$ means that v_n^i is deployed, 0 is otherwise. Recall that $\alpha_n^{0,l} = \alpha_n^{I+1,l} = 1$ ensures the beginning and ending allocation of task S_n, respectively. To model the time and energy consumption of task execution, we have the following definitions:

Definition 1 (*Ready Time*): The time that the task ms_n^i has all the prerequisites Let $RT_n^{i,l}$, $l \in \mathcal{I}$ denote the ready time of task ms_n^i executed at *UE Layer* and *Edge Layer*, respectively.

Definition 2 (*Finish Time*): The time that the task ms_n^i accomplishes all the workload C_n^i. We define $FT_n^{i,l}$, $l \in \mathcal{I}$ as the finish time of task ms_n^i executed at *UE Layer* and *Edge Layer*, respectively.

Definition 3 (*Wireless Receiving Time*): Accordingly, we define $RT_n^{i,wt,l}$, $FT_n^{i,wt,l}$, $l \in \mathcal{I}$ as the ready time and finish time of the task ms_n^i when receiving the wireless channel from the *Edge Layer*.

A. Local Execution Model

If the task v_n^i is executed locally, then it is computed by the local CPU of UE with the computation capable of c_m^{UE} frequency. Accordingly, we have the local processing time as $T_n^{i,m} = \frac{d_n^i}{c_m^{UE}}$. Thus, the finish time of task v_n^i at *UE Layer* is $FT_n^{i,m} = RT_n^{i,m} + T_n^{i,m}$. The corresponding energy consumption of task $_n^i$ at local execution can be obtained as $\epsilon_n^{i,m} = \kappa_m C_n^i (f_m^{UE})^2$ [12], where κ_m is the coefficient related on chip types.

B. Edge Layer Execution Model

When the task v_n^i is decided to execute on Edge layer, it is first transmitted to a edge server k via cellular link, then the computation process is carried out.

Denote the transmission time by $T_n^{i,w,k} = d_n^i / \mu_{m,k}$, $\mu_{m,k}$ is the uplink transmission rate [13]. In this paper, we set

the channel gain as $g^{m,k} = -4th$ power of the distance between UE m and the edge server k. In this case, the energy consumption of UE m is $\epsilon_n^{i,k} = g^{m,k} \times T_n^{i,w,k}$. Thus, the ready time on *Edge Layer* is $RT_n^{i,k} = T_n^{i,w,k}$.

Suppose that each edge server equips the computation ability with the c_m^k CPU frequency, then the execution time of task v_n^i can be calculated as $T_n^{i,k} = \frac{d_n^i}{c_m^k}$. Consequently, the finish time of task v_n^i on *Edge Layer* is $FT_n^{i,k} = RT_n^{i,k} + T_n^{i,k}$. The energy of task execution in edge server k is $\epsilon_n^{i,k} = \kappa_k C_n^i (f_m^k)^2$.

C. Problem Formulation

Our objective is to maximize the system gain in terms of the latency and energy consumption, which indicates the difference between the benefit of the tasks execution under a scheduling strategy and when all tasks are executed locally.

Recall that the allocation variable $\alpha_n^{i,l} = \{0, 1\}$, we calculate the overall Finish Time FT^{all} as the sum of all the Finish Time of tasks in the system, shown as follows:

$$FT^{all} = \sum_{v_n^i \in \mathcal{V}_n} \sum_{l \in \{M,K\}} \alpha_n^{i,l} FT_n^{i,l}. \tag{1}$$

Besides, the overall energy consumption for executing all tasks:

$$E^{all} = \sum_{v_n^i \in \mathcal{V}_n} \sum_{l \in \{M,K\}} \alpha_n^{i,l} \epsilon_n^{i,l}. \tag{2}$$

We hereafter define the latency gain and energy consumption gain as:

$$\begin{cases} FT^{Gain} = FT^m - FT^{all} \\ E^{Gain} = E^m - E^{all} \end{cases}, \tag{3}$$

where the local operations are expressed as:

$$FT^m = \sum_{v_n^i \in \mathcal{V}_n} \sum_{l \in M} \alpha_n^{i,l} FT_n^{i,l} \tag{4}$$

and

$$E^m = \sum_{v_n^i \in \mathcal{V}_n} \sum_{l \in M} \alpha_n^{i,l} \epsilon_n^{i,l}. \tag{5}$$

Note that the FT^m and E^m are intuitive bigger than FT^{all} and E^{all} due to the lower computation abilities of UEs, resulting to the higher latency and energy consumptions. Then the system gain function G is derived as:

$$G = \omega FT^{Gain} + (1 - \omega) E^{Gain}, \tag{6}$$

Finally, the objective of collaborative task allocation problem (CTA) is to maximize the system gain G:

$$\begin{aligned} \max_{\alpha} \quad & G \\ s.t. \quad & \omega_t + \omega_e = 1 \\ & v_n^i \in \mathcal{V}_n \\ & \alpha \in \{0, 1\} \end{aligned}. \tag{7}$$

Due to the terms $\alpha_n^{i,l} \in \{0, 1\}$ in the objective (7), the problem can be regarded as a mixed binary integer linearly constrained programming (MBILP). The similarly problem is proved as *NP-hard* [14], thereby it is not feasible to solve the problem by heuristic algorithm or dynamic programming because of its high computational and spatial complexity and large scale. Thus, we carry out a DRL-based method to solve the aforementioned problem. Before introducing the proposed DRL framework, it is necessary to embed the representation of nodes and tasks in the network to cope with the excessive state and action space.

III. Attention-aided Federated Learning Design

The implementation process is shown as Fig.2, we first introduce the task representation to extract the system features by using multi-head attention mechanism, to overcome the problem of feature differentiation among various data holders. Then, a decentralized attention-aided federated learning algorithm (AteFL) is proposed to solve the CTA problem.

A. System Features Embedding

We extract the system features from two aspect, i.e., the infrastructure and task. The infrastructure features $f_x = \{f_m, f_k\}, x \in \mathcal{I}$ is based on the calculation of task and energy overhead, related to the features of user devices f_m and edge server f_k, respectively.

Aiming to learn the embedding of the infrastructure status, the features extracted from the infrastructures are fed into the MLP (multilayer perceptron) with 2 hidden layers and 1 output layer, containing 512 neurons on each layer. we have the embedding $\mathbf{w}_x = \mathbf{H} \cdot f_x$, where the transformation matrix $\mathbf{H}$ is used to map the features of the infrastructure status f_x. In order to capture the overall structure of the allocation of task and make the optimal policy for each task $ms_n^i \in MS_n$. Considering the dependencies, we mainly embed the information of the CPU requirement C_n^i and the input size of subtask P_n^i by employing the same transformation matrix $\mathbf{H}$ to map the subtasks' features $f_y = \{f_y^{C_n^i}, f_y^{P_n^i}\}$, $y \in v_n^i$. Similarly, we can obtain the corresponding embedding as $\mathbf{w}_y = \mathbf{H} \cdot f_y$.

Furthermore, we consider that the infrastructure and the allocated subtask on that forms a structure, and there exists a meta-chain $\mathbf{O}$ of subtask DAG which indicates the information and allocation process passing by along with the system execution. Moreover, the structure self-attention $\beta_{xy,k}^{\mathbf{O}}([\mathbf{w}_x, \mathbf{w}_y], \mathbf{O})$ is used to indicates the importance of the subtask embedding y to the infrastructure embedding x on the meta-chain orchestrated by edge server k. Note that it can be shared by the infrastructure-task pairs when they are in the same meta-chain, since the mapping patterns is similar to each other under a certain meta-chain. Thus, the masked attention mechanism is deployed to inject the graph structure information. In this way, we have the local self-attention expression as:

$$\tau_{xy,k}^{\mathbf{O}} = softmax(\beta_{xy,k}^{\mathbf{O}}) = \frac{exp(\psi(\mathbf{w}^{\mathbf{O}} \cdot [\mathbf{w}_x || \mathbf{w}_y]))}{\sum_{h \in N_h^{\mathbf{O}}} exp(\psi(\mathbf{w}^{\mathbf{O}} \cdot [\mathbf{w}_x || \mathbf{w}_h]))}, \tag{8}$$

where $\mathbf{w}^{\mathbf{O}}$ indicates the chain-attention vector for the task meta-chain $\mathbf{O}$, $N_r^{\mathbf{O}}$ is the set neighbor tasks of infrastructure h on the task meta-chain $\mathbf{O}$, and $\psi(\cdot)$ denotes the LeakyReLU function with the concatenation operation $[\cdot||\cdot]$.

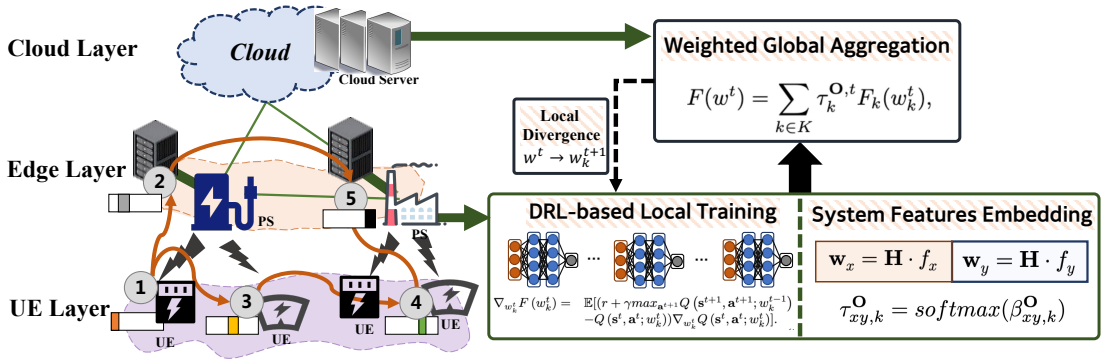


Fig. 2. Implementation of AteFL scheme

B. Task Allocation Markov Decision Process

We consider the subtask allocation learning is in the continuous action spaces, and an infinite-horizon Markov Decision Process (MDP) is deployed as follows:

1) System State: The system state at time lot t can be described with two-tuple as $\mathbf{S}^t = \{\mathbf{s}^t\}_{\times(x,y)} = \{\mathbf{s}_x^t, \mathbf{s}_y^t\}$ indicating the system infrastructure state and task placement state, respectively.

2) System Action: The system decides which task is executed in which infrastructure. We define the system action space as $\mathbf{A}^t = \{\mathbf{a}_n^t\}_{\times\mathbb{S}_n} = \{\alpha_n^{i,l}\}, \forall n, i \in ms_n^i, l \in \mathcal{I}$.

3) System Reward: It is calculated as the weighted sum of current reward r, we formulate the system reward according to the optimization objective, expressed as $\mathbf{R}^t(\mathbf{S}^t, \mathbf{A}^t) = \omega FT^{Gain} + (1-\omega)E^{Gain}$.

C. DRL-based Local Task Allocation Training Process

In this paper, we focus on maximize the system gain function under a certain allocation policy, the allocation policy $\pi(\mathbf{a}^t|\mathbf{s}^t) : \mathbf{S}^t \rightarrow \mathbf{A}^t$ is designed for mapping from the system state under the dynamic environment to the action. We consider the stochastic policy by augmenting the cumulative system reward in the finite-horizon scenario. The objective is to find the optimal allocation policy π^* with a discount rate $\gamma \in [0, 1)$ as:

$$\pi^\star = arg\ max_\pi \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}^t,\mathbf{a}^t)\sim\pi} \left[\gamma^t \cdot r\left(\mathbf{s}^t, \mathbf{a}^t\right)\right], \quad (9)$$

Accordingly, the optimal action-value function $Q^*(\mathbf{s}^t, \mathbf{a}^t) = max_\pi \mathbb{E}[r^t|(\mathbf{s}^t, \mathbf{a}^t), \pi]$ is defined as the maximum expected reward by the allocation strategy after realizing the system state $\mathbf{s}^t$ and taking the action $\mathbf{a}^t$. In this way, the Q-function iteration formula can be derived according to Bellman operator in the following if the optimal $Q^*(\mathbf{s}^{t+1}, \mathbf{a}^{t+1})$ can satisfy for all possible action $\mathbf{a}^{t+1}$:

$$Q^*(\mathbf{s}, \mathbf{a}) = \mathbb{E}[r + \gamma max_{\mathbf{a}^{t+1}} Q^*(\mathbf{s}^{t+1}, \mathbf{a}^{t+1}) \mid (\mathbf{s}, \mathbf{a})]. \quad (10)$$

We use the DQN [15] to realize the local task allocation training process. The training process is shown in Fig. xxx. An approximator orchestrated by local edge server k for updating the parameter w_k^t of the MLP is used to estimate the action-value function, $Q(\mathbf{s}, \mathbf{a}; w) \approx Q^*(\mathbf{s}, \mathbf{a})$. Then a Q-network is trained by minimizing the following loss function at each iteration t:

$$F(w_k^t) = \mathbb{E}[(r + \gamma max_{\mathbf{a}^t} Q\left(\mathbf{s}^t, \mathbf{a}^t; w_k^{t-1}\right) \mid \left(\mathbf{s}^{t-1}, \mathbf{a}^{t-1}\right) - Q\left(\mathbf{s}^{t-1}, \mathbf{a}^{t-1}; w_k^{t-1}\right))^2], \quad (11)$$

we can obtain the following gradient by differentiating the above function:

$$\nabla_{w_k^t} F\left(w_k^t\right) = \mathbb{E}[(r + \gamma max_{\mathbf{a}^{t+1}} Q\left(\mathbf{s}^{t+1}, \mathbf{a}^{t+1}; w_k^{t-1}\right) - Q\left(\mathbf{s}^t, \mathbf{a}^t; w_k^t\right))\nabla_{w_k^t} Q\left(\mathbf{s}^t, \mathbf{a}^t; w_k^t\right)]. \quad (12)$$

Then the local update of parameter of w_k^t is:

$$w_k^{t+1} = w_k^t - \eta^t \nabla F_k(w_k^t), \quad (13)$$

where η^t is the step size.

D. Attention-Aided Federated Aggregation Process

Generally, the federated aggregation process employ a subset of the participants from the local training devices, then averages the parameters. However, due to the heterogeneity of local training capabilities in terms of different allocated task DAG, computing abilities, and model qualities. It is unpractical to treat the contributions equally of all local models, to this end, we propose a weighted aggregation process to measure the different contributions of local model to global model by considering the local system representation via attention mechanism. In this way, the global aggregation process is expressed as:

$$F(w^t) = \sum_{k\in K} \tau_k^{\mathbf{O},t} F_k(w_k^t), \quad (14)$$

where $\tau_k^{\mathbf{O},t}$ is the obtained from system features embedding process, indicating the importance of the local model by jointly considering both infrastructure and task features.

The process of the proposed AteFL is shown in Algorithm.1. Input the system state $\{\mathbf{s}^t\}_{\times(x,y)}$ and action $\{\mathbf{a}^t\}_{\times\mathbb{S}_n}$ (Line 3), compute the current reward and next system state at each episode (Line 4-9). The parameters are updated afterwards in the local edge server k by Eq. (13) and then sent to cloud center for aggregation operation (Line 10-15). Then parameters are weighted aggregated by considering the system representation via Eq.(14) (Line 16-20), finally the updated parameters are sent back to each edge server (Line 21). The

Algorithm 1 AteFL Algorithm

Initialize: System features f_x, f_y;
Number of iterations T;
Step size η.
for $t = 0, 1, 2, ..., T$ **do**
  Obtain the system state $\{\mathbf{s}^t\}_{\times(x,y)}$ and action $\{\mathbf{a}_n^t\}_{\times\mathbb{S}_n}$;
  for each episode **do**
    Get system action $\mathbf{a}^t$ according to input system state $\mathbf{s}^t$ in actor network, holding $\mathbf{a}^t \sim \pi_{\varkappa}(\mathbf{a}^t \mid \mathbf{s}^t)$.
    Obtain current reward r^t and next system state $\mathbf{s}^{t+1}$.
    Store the quadruplet into replay memory $\mathcal{D} \leftarrow \{(\mathbf{s}^t, \mathbf{a}^t, r^t, \mathbf{s}^{t+1})\} \cup \mathcal{D}$.
    Randomly sample a batch of $\mathcal{N}$ samples from $\mathcal{D}$.
  end for
  for each edge server k **do**
    Obtain the local self-attention by Eq.(8) and the local parameters by Eq.(12).
    Set $w_k^{t+1} = w_k^t - \eta^t \nabla F_k(w_k^t)$.
    Return w_k^{t+1}.
  end for
  Send w_k^{t+1} to cloud center.
  for Cloud center C **do**
    Receive w_k^{t+1} from each local edge server k.
    Update the global model according to Eq.(14).
    Input the global model: $w_k^{t+1} = w^{t+1}$.
  end for
  Cloud C disperses w_k^{t+1} to each edge server k.
end for

complexity is derived hereafter, there are T iterations in the outer loop, $\mathcal{N}$ samples in one episode in the first inner loop, the local update occurs in K edge servers in the second inner loop, and 1 global aggregation operation in cloud in the last inner loop, thus, the complexity of AteFL is $\mathcal{O}(T \cdot max\{\mathcal{N}, K\})$.

IV. Experiment

A. Simulation Settings

In this section, we conduct the experimental simulations in the python environment, the main parameter values of computation ability of UE layer f^m and edge layer f_m^k are uniformly set as 1.0-1.2GHz and 2.4-2.5GHz, respectively. The discount rate γ is 0.85 and batch size is 128. Other parameters of simulations in this work are similar to [13]. The comparisons of system reward under different learning rate is shown in Fig. 3. It can be seen that when the learning rate is set as 1e-4 and 5e-5, the performance increases sharply at first and sudden drops down after 125 episodes, finally converges to 34. while when learning rate is 1e-5, the performance shows a stable increasing and convergence trend, thus we choose this value for the simulation to guarantee the stability of the proposed AteFL.

B. Simulation Results

We demonstrate the simulation results in terms of the performance of average system cost and the system reward. All the results are obtained by the average values for 20 times.

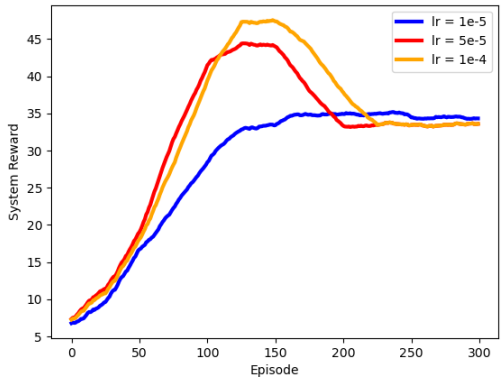


Fig. 3. System reward performance under different learning rate.

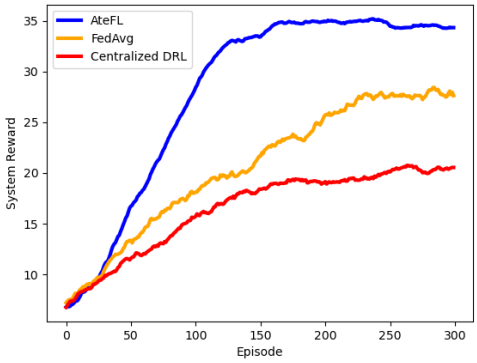
To evaluate the proposed AteFL, some state-of-the-art schemes are deployed for comparison as follows:

- *FedAvg*: The federated learning algorithm elaborates an average-sum operation in global aggregation process [16].
- *Centralized DRL*: The task allocation is orchestrated in the centralized cloud by deploying the DQN algorithm [15].

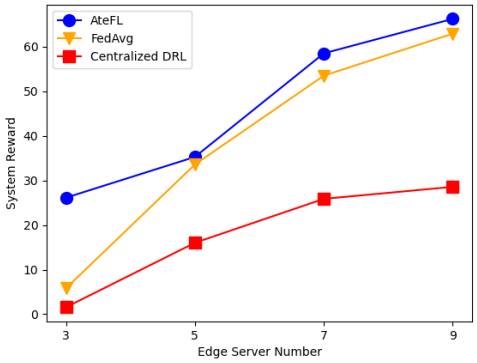
We first demonstrate the system reward under different episodes, shown as 4(a). The proposed AteFL shows a rpaid growth when the system starts and converges to 35 at last. It is observed that it improves the performance of 10% and 15%, compared to the FedAvg and the centralized DRL algorithms, respectively. The main reason is that the utilization of the decentralized manner reduces the latency of data transmission in the network compared to the centralized DRL. Besides, we use the attention mechanism to quantify the contribution of local training parameters, which effectively improves system training process. Then, the performance under different edge server number is illustrated in Fig. 4(b), the proposed AteFL outperforms the FedAvg and centralized DRL with up to 5% and 45%, respectively. From Fig. 4(c), it can be seen that the proposed AteFL and FedAvg show a continuous growth trend, while the centralized DRL tends to converge after 10 tasks. From Fig. 4(d), the proposed shows a fast convergence trend, and improves the performance with up to 7.5% and 15% compared to FedAvg and contralized DRL, respectively.

V. Conclusion

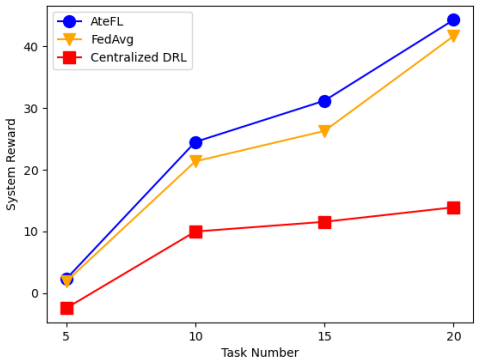
We have proposed an attention-aided federated learning algorithm (AteFL) to deal with the collaborative task allocation (CTA) problem. The dependency of task has been modeled by directed acyclic graph (DAG), and the multi-head attention mechanism has been used to extract the system context, which indicated the weight of the contributions in the global aggregation process. Then the decentralized federated learning has been employed to solve the CTA problem. The experimental simulation results showed the priorities of the proposed algorithm.



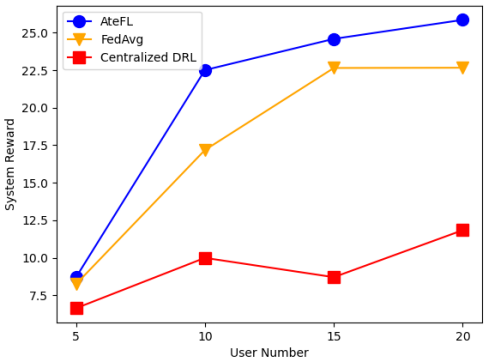
(a) System reward under different episodes.



(b) System reward under different edge server number.



(c) System reward under different task number.



(d) System reward under different user number.

Fig. 4. Performance of system reward under different number of episodes, edge servers, tasks and UEs.

ACKNOWLEDGMENT

This work was supported partially by the Science and Technology Research of State Grid Corporation of China "Research on Data Sharing and Model Fusion Technology of Power Marketing based on Federated Learning", Grant No. 5700-202113262A-0-0-00, and the Chinese Government Scholarship (NO. 202006250167) awarded by China Scholarship Council.

REFERENCES

[1] Q. Meng, S. Zhao, Z. Huang, and F. Zhou, "Magface: A universal representation for face recognition and quality assessment," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 225–14 234.

[2] M. L. Tuballa and M. L. Abundo, "A review of the development of smart grid technologies," *Renewable and Sustainable Energy Reviews*, vol. 59, pp. 710–725, 2016.

[3] C. A. J. B. Gomes, Gilzamir and Y. L. Nogueira, "Two level control of non-player characters for navigation in 3d games scenes: A deep reinforcement learning approach," in *SBGames*, 2021, pp. 182–190.

[4] H. D. Kabir, A. Khosravi, S. K. Mondal, S. Rahman, Mustaneer, and R. Buyya, "Uncertainty-aware decisions in cloud computing: Foundations and future directions," *ACM CSUR*, vol. 54, no. 4, pp. 1–30, 2021.

[5] H. Sun, S. Wang, F. Zhou, L. Yin, and M. Liu, "Dynamic deployment and scheduling strategy for dual-service pooling based hierarchical cloud service system in intelligent buildings," *IEEE T Cloud Comput*, 2021.

[6] X. Jiang, R. Chen, D. Chen, Y. Cai, J. Liu, and J. Zhang, "Research on privacy protection of power users based on big data desensitization technology," 2021.

[7] Y. Yang and W. Zhang, "Research on digital medical data security protection technology," in *2nd International Conference on Internet of Things and Smart City (IoTSC 2022)*, vol. 12249. SPIE, 2022, pp. 314–320.

[8] X. Wang, C. Wang, X. Li, V. C. Leung, and T. Taleb, "Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9441–9455, 2020.

[9] F. Guo, B. Tang, M. Tang, H. Zhao, and W. Liang, "Microservice selection in edge-cloud collaborative environment: A deep reinforcement learning approach," in *2021 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2021 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. IEEE, 2021, pp. 24–29.

[10] H. Tian, X. Xu, T. Lin, Y. Cheng, C. Qian, L. Ren, and M. Bilal, "Dima: Distributed cooperative microservice caching for internet of things in edge computing by deep reinforcement learning," *WWW*, pp. 1–24, 2021.

[11] G. Zhao, H. Xu, Y. Zhao, C. Qiao, and L. Huang, "Offloading tasks with dependency and service caching in mobile edge computing," *IEEE TPDS*, vol. 32, no. 11, pp. 2777–2792, 2021.

[12] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun Surv Tut*, vol. 19, no. 4, pp. 2322–2358, 2017.

[13] J. Chen, Y. Yang, C. Wang, H. Zhang, C. Qiu, and X. Wang, "Multi-task offloading strategy optimization based on directed acyclic graphs for edge computing," *IEEE Internet Things J.*, 2021.

[14] X. Li, X. Wang, Z. Han, and V. C. Leung, "Hierarchical edge caching in device-to-device aided mobile networks: Modeling, optimization, and design," *IEEE J-SAC*, vol. 36, no. 8, pp. 1768–1785, 2018.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[16] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.