

Multi-source DNN task offloading strategy based on in-network computing

Lizi Hu*, Yuhao Chai*, Qin Li**, Weiyuan Li**, Yong Zhang*

**Beijing Key Laboratory of Work Safety Intelligent Monitoring, Beijing University of Posts and Telecommunications, Beijing, 100876, China*

***China Mobile Research Institute, Beijing, China*

lizihu@bupt.edu.cn, chaih@bupt.edu.cn, liqinyjy@chinamobile.com, liweiyuan@chinamobile.com, yongzhang@bupt.edu.cn

Abstract—As applications grow in scale, the centralized computing approach leads to excessive bandwidth requirements and high computational latencies. The traditional computing model regards the network as a transmission pipeline and has not fully explored the potential of network devices. At present, in-network computing is a new type of computing model that delegates application-layer processing functions to the network data plane. It can process traffic during transmission, reduce the cost of network bandwidth transmission, and alleviate the computing pressure and energy consumption of the cloud-side system. However, it needs to consider the redeployment cost. This paper proposes a multi-source DNN task offloading strategy based on in-network computing, which combines edge calculation, in-network computing and cloud computing. At the same time, it makes full use of the traditional routing nodes with no computing ability in the network. Particle swarm optimization (PSO) is used to solve the problem in the offloading scheme optimization. Service level agreement violation (SLAV) is introduced, and the network resources are offloaded in balance while the quality of service of users is satisfied. Simulated experiment results show that the proposed algorithm can reduce the cost and achieve convergence compared with the traditional offloading algorithm. In particular, we can find that there is an optimal deployment scheme in the network, which can make full use of the computing resources and bandwidth resources of the network, significantly reduce the computing pressure and transmission overhead of the whole network, and realize the balanced offloading of node resources.

Keywords—in-network computing, edge-cloud computing, task offloading, particle swarm optimization

I. INTRODUCTION

The popularity of a large number of smart mobile terminals has promoted the development of edge computing and cloud computing. Because computing all AI tasks locally will cost users a lot of time and computing resources, offloading computing tasks to edge computing nodes or cloud service nodes becomes a useful solution [1]. These nodes have more computing resources, and can meet the user's high-quality requirements of services, such as real-time and interactive. Therefore, task offloading optimization is one of the research hotspots in the field of edge computing and cloud computing.

The current research on task offloading optimization usually focuses on two-tier architecture or three-tier architecture of network, including user node, edge node and cloud node. With the emergence of AI service and the expansion of network scale, the traditional single-task offloading scheme can not meet the requirement of users and networks [2], a number of studies have been done to optimize the simultaneous offloading of multiple dependent tasks by minimizing the network cost [3], the total energy consumption of equipment [4,6], the number of workflows [5,6] and the task completion time [7], etc.

The demand for deploying Deep Neural Networks (DNN) models to mobile devices is growing in tandem with the increased computing capabilities of mobile devices. In the existing work, the DNN model is modelled as a directed acyclic graph (DAG). Because a DNN inference task usually contains multiple dependent subtasks, the strategy of subtasks offloading allocation and scheduling among multiple edge nodes or cloud nodes will greatly affect the total DNN task delay. At the same time, the dynamic change in the network environment makes the offloading decision-making become a complex problem, so the DNN inference tasks optimize the offloading and scheduling of subtasks to improve the performance of the whole network [8]. In particular, the constrained cost optimization problem in heterogeneous MEC networks is studied in [9], and a two-phase scheduling algorithm for edge and cloud is proposed to meet the service delay requirements. The heuristic sorting algorithm is used in [10] to solve the problem of tasks adaptive allocation in MEC and cloud nodes and to minimize the average delay. The DNN partition is given in [11] to treat each layer in DNN as a vertex and a DNN can be modelled as a DAG to optimize the scheduling problem for cooperative DNN inference. In addition, some studies have applied reinforcement learning to solve complex dependent task placement problems [12]. In these studies, the optimal strategy of node placement is obtained by establishing Actor-Critic mechanism and non-policy gradient algorithm.

However, in the current study, few studies have taken into account the widespread computing power nodes in networks,

often only using nodes between the edges and the cloud for forwarding. In-network computing is an emerging computing paradigm [13], which can reduce time delay, energy consumption and increase resource utilization by using computing power nodes in the networks to offload DNN inference tasks and transfer tensors. At present, the problem is how to dynamically offload the inference task to the edge node, computing network node and cloud server node [14]. The node placement problem with multiple dependent subtasks at the same time is more complex in in-network computing.

We propose a multi-source DNN task offloading algorithm based on in-network computing, which combines edge computing nodes, computing network nodes and cloud computing nodes. At the same time, it makes full use of the traditional routing nodes in the network. Particle swarm optimization (PSO) is used to solve the problem in the offloading scheme optimization, and the network resources are offloaded in balance while the quality of service of users is satisfied.

The main contributions of this work are summarized below.

(1) A multi-source DNN task offloading model is established in the network including edge computing, in-network computing and cloud computing. The inference task requirement model and offloading cost model are defined in this system.

(2) In order to make full use of the nodes with computing power in the network, the optimization objectives based on load balance, transmission cost and computing cost are established, the difference of computing capacity of heterogeneous computing nodes is fully considered.

(3) A particle swarm optimization algorithm is designed to optimize task offloading and meet the expected quality of service.

II. SYSTEM MODEL AND PROBLEM FORMULATION

We consider a resource-constrained inference tasks offloading scenario, consisting of a set of tasks $\mathbf{W}$, each of which can be divided into several subtasks that may or may not have dependencies, the summary of notations is shown in Table 1.

A. System Model

We assume that the set of nodes $\mathbf{N}$ that can be offloaded including edge nodes, computing network nodes and cloud computing nodes. Each node $\mathbf{N}_k$, $\forall k \in \{1, 2, \dots, m\}$ has heterogeneous location L_i^k , computing resource c^k , unit computing cost $comp_cost^k$ and the proportion of remaining computing resources β^k . At present, the proposed in-network computing model is deployed in-network computing device, combined with edge computing and cloud computing to form an edge-network-cloud network system with computing ability. However, this network model does not take into account the traditional routing nodes without computing ability in the existing network. In this paper, we propose a network model with computing power nodes. The system model diagram is shown in Fig.1, including edge computing nodes, traditional routing nodes without computing ability, in-network

computing (INC) devices with different computing resources and cloud. The primary INC devices with larger computing resources is close to the cloud while the secondary INC devices is close to the edge with smaller computing resources than the primary INC devices.

Different inference task i input from different MEC devices and select the device with computing ability to offload. We state the properties and requirements of task i as $\mathbf{W}_i = \{y_i, T_i, w_{ij}, \mathbf{G}_{ij}, d_{ij}, t_{ij}, x_{ij}^k\}$. In a DNN task, subtask dependency captured in the DAG is shown in Fig.2. $v_i = \{v_1^i, v_2^i, \dots, v_p^i\}$ is the set of subtasks in the DNN task i . Every subtask v_p^i represents a layer of the DNN. In addition, to characterize the dependencies among subtasks, $\mathbf{G}_{ij} = \{g_{ij}, l_{ijz}, \alpha_{ijz}\}$ is the collection of relationships between parent and child subtasks in a DAG. Each subtask unable to begin until all of its parent subtasks have been completed. For each DNN task, we measure both the transmission cost and the computation cost of tasks.

TABLE 1. SUMMARY OF NOTATIONS

Notation	Description
$\mathbf{N}$	$\mathbf{N} = \{N_1, N_2, \dots, N_m\}$, $\mathbf{M} = \{1, 2, \dots, m\}$ is a currently available set of nodes. Each node is defined as $\mathbf{N}_k = \{c^k, comp_cost^k, L_i^k, \beta^k\}, \forall k \in \mathbf{M}$. c^k is the computation resources of node k . $comp_cost^k$ is the unit computation cost node k . L_i^k is the distance between the location of node k and the initial node of task i . β^k is the proportion of remaining computing resources of node k .
$\mathbf{W}$	$\mathbf{W} = \{W_1, W_2, \dots, W_n\}$, $\mathbf{Q} = \{1, 2, \dots, n\}$ is the total number of current inference tasks. Each task can be divided into some subtasks and each subtask is defined as $\mathbf{W}_i = \{y_i, T_i, w_{ij}, \mathbf{G}_{ij}, d_{ij}, t_{ij}, x_{ij}^k\}$. y_i is the number of subtasks of task i . T_i is the deadline of task i . w_{ij} is the computing resources required for subtask j of subtask i . d_{ij} is the amount of transmitted data required for subtask j of task i . t_{ij} is the arrival time of subtask j of task i . $x_{ij}^k, x_{ij}^k \in \{0, 1\}$, is a binary variable, representing whether subtask j of task i is offloaded to node k .
$\mathbf{G}_{ij}$	$\mathbf{G}_{ij} = \{g_{ij}, l_{ijz}, \alpha_{ijz}\}$ is the collection of relationships between parent and child subtasks in a DAG.

	$g_{ij}, z \in g_{ij}$, is the set of numbers representing whether subtask z is the parent of subtask j . l_{ijz} is the link distance between subtask z and subtask j after placement. α_{ijz} is the tensor transmission ratio from subtask z to subtask j of task i .
$unit_tran_cost$	The unit transmission cost.
$unit_comp_cost^k$	The unit computation cost of node k .
$tran_time$	The transmission delay per unit distance.
$comp_time_{ij}^k$	The computation time of offloading of node k for subtask j in task i .
$SLAV_{ij}^k$	Service level agreement violation, indicating the service level of node k for subtask j in task i .

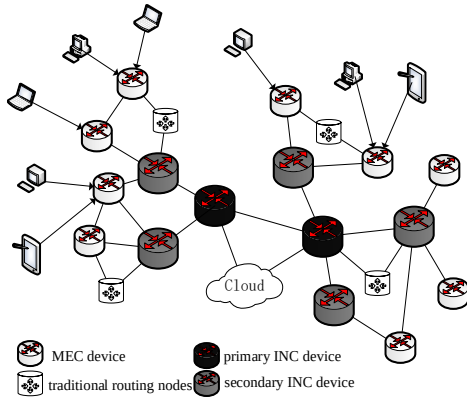


Figure 1. Edge-network-cloud network system model

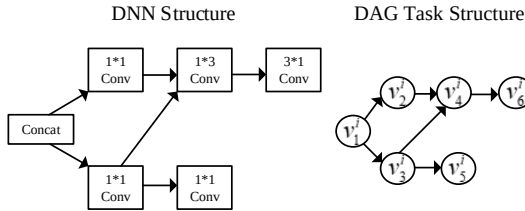


Figure 2. An example of DAG for a DNN task

B. Transmission Cost Model

The transmission cost $tran_cost_i$ of task i is the sum of the transmission costs of all subtasks. The transmission cost of subtask j includes the offloading cost $offloading_cost_{ij}$ of transmitting subtask j to the offloaded node and the tensor cost $tensor_cost_{ij}$ of transmitting tensor from all predecessor subtasks to subtask j .

The offloading cost $offloading_cost_{ij}$ depends on the amount of data and the distance between the initial node of task i and the offloaded node. The offloading cost $offloading_cost_{ij}$ of subtask j in task i is calculated as:

$$offloading_cost_{ij} = \sum_{k=1}^m x_{ij}^k d_{ij} L_i^k \cdot unit_tran_cost \quad (1)$$

The tensor cost $tensor_cost_{ij}$ depends on the tensor size from predecessor subtasks and the distance between predecessor subtasks and subtask j . The tensor cost $tensor_cost_{ij}$ of subtask j in task i is calculated as:

$$tensor_cost_{ij} = \sum_{z \in g_{ij}} \sum_{k=1}^m x_{ij}^k g_{ijz} \alpha_{ijz} d_{iz} l_{ijz} \cdot unit_tran_cost \quad (2)$$

The transmission cost $tran_cost_i$ of task i is calculated as:

$$tran_cost_i = \sum_{j=1}^{y_i} offloading_cost_{ij} + \sum_{j=2}^{y_i} tensor_cost_{ij} \quad (3)$$

C. Computation Cost Model

The computation cost $comp_cost_i$ of task i is the sum of the computation costs of all subtasks. We define $SLAV_{ij}^k$ as service level agreement violation, which describes the service level of node k offloading subtask j in task i .

$$SLAV_{ij}^k = \frac{x_{ij}^k w_{ij}}{\beta^k c^k - \sum_{i=1}^n \sum_{j=1}^{y_i} x_{ij}^k w_{ij}}, \beta^k c^k > \sum_{i=1}^n \sum_{j=1}^{y_i} x_{ij}^k w_{ij} \quad (4)$$

where $\beta^k c^k$ is the remaining computing resources of node k . When $\beta^k c^k \leq \sum_{i=1}^n \sum_{j=1}^{y_i} x_{ij}^k w_{ij}$, the node k can not satisfy the offloading requirements of subtask j . The reassignment job offloading policy $\{x_{ij}^k\}$ is adopted until SLAV of all nodes is satisfied.

The computation cost $comp_cost_i$ of task i is the sum of the computation costs of all subtasks. The computation cost of subtask j depends on the amount of data and the service level of offloaded node k . Thus, the computation cost $comp_cost_i$ of task i is calculated as:

$$comp_cost_i = \sum_{j=1}^{y_i} \sum_{k=1}^m SLAV_{ij}^k \cdot d_{ij} \cdot unit_comp_cost^k \quad (5)$$

D. Problem Formulation

We aim to minimize the total cost of all subtask offloading decisions with the following constraints. For each task, both the transmission cost and the computation cost are considered. The multi-source DNN task offloading problem is defined as follows:

$$\text{Minimize cost} = \sum_{i=1}^n (tran_cost_i + comp_cost_i) \quad (6)$$

$$s. t. x_{ij}^k g_{ijz} (t_{ij} - t_{iz} - comp_time_{ij}^k - l_{ijz} \cdot tran_time) \geq 0 \quad (7)$$

$$\forall i \in N, j \in y_i, k \in M, z \in g_{ijz} \quad (7)$$

$$t_{ij} + comp_time_{ij}^k \leq T_i \quad (8)$$

$$x_{ij}^k \in \{0,1\} \quad (9)$$

Constraint (7) is the time constraint of the subtask j , indicating that the start time t_{ij} of subtask j must be after the end time of all its predecessor tasks. Constraint (8) is the demand constraint of task i , indicating that the completion time of the last subtask j can not exceed the deadline of task i . Constraint (9) is the inherent constraint of offloading association.

III. DNN TASK OFFLOADING ALGORITHM

In this section, the DNN task offloading strategy based on PSO algorithm are introduced. First, the task is input by the

MEC device and the subtasks of the task are sorted. Then, PSO algorithm is used to constantly update the offloading position. When convergence is achieved, the optimal offloading position is obtained.

Subtask sorting: First, topological sorting is used for each subtask within each task to realize the transformation from graph to list and obtain the topological sorting priority of each subtask. Select the vertex whose input degree is 0 and output it to the list. Delete this vertex and all outgoing edges from the network. Continue to loop until completion and get the topological sorting priority of each subtask [15].

Then, we define the global priority E_{ij} of subtask j in task i . The global priority depends on the deadline of task i and the topological sorting priority of subtask j . The global priority E_{ij} of subtask j in task i is calculated as:

$$E_{ij} = \frac{T_i}{\sum_{l=1}^n T_l} \cdot e_j, \quad e_j > 0 \quad (10)$$

where T_i is the deadline of task i and e_j is the topological sorting priority of subtask j in task i .

Particle swarm optimization: PSO algorithm is a stochastic optimization technique. Its basic idea is to find the optimal solution through cooperation and information sharing among individuals in the swarm. PSO algorithm designs particles with moving speed and moving direction. Each particle searches the optimal solution respectively in the search space, records it as the current particle extreme value, shares its with other particles in the whole particle swarm and compares to get the current global extreme value. All particles adjust their speed and position according to their current particle extreme value and the current global extreme value.

In the previous section, we have modelled the system model and the task relationships. In the PSO algorithm, the particle coding adopts integer coding and the set of all particle is the swarm. The particle position vector $\mathbf{X} = \{x_{ij}^k\}$ is used to indicate that all subtasks are offloaded to the corresponding MEC device, INC device or cloud. The particle constraint velocity vector is used to represent the maximum speed and this constraint is represented by V_{max} . If the particle flies too fast, it is likely to fly directly past the optimal solution position. The fitness function of the particle is the total cost of all subtask offloading decisions with the constraints, as shown in Eq. (6).

The DNN task offloading strategy based on PSO algorithm is shown in Table 2.

Step 1: Initialize the swarm. We give the initial velocity vector and initial position vector of the particle. We also give the swarm size $L = \{1, 2, \dots, l\}$ and set various parameters.

Step 2: Calculate the fitness value. We calculate the fitness value of the particle according to Eq. (6) and save it. We compare it with the historical optimum fitness value of the particle, save the smaller value as the optimum fitness value.

Step 3: Find the global optimum fitness value of the swarm. The smallest optimum fitness value of the swarm is used as the global optimum fitness value and every particle preserves it.

Step 4: Update the particle velocity and position. Each particle updates its velocity and position in the iteration. In iteration $t+1$, the velocity $V_{l,t+1}$ is calculated as:

$$V_{l,t+1} = \begin{cases} \omega \times V_{l,t} + c_1 \times r_1 \times (P_{l,best} - P_{l,t}) \\ + c_2 \times r_2 \times (G_{best} - P_{l,t}), & V_{l,t+1} < V_{max} \\ V_{max}, & V_{l,t+1} > V_{max} \end{cases} \quad (11)$$

where ω is the inertia weight used to adjust the search range of the solution space, $P_{l,best}$ is the optimum fitness value of the particle l , G_{best} is the global optimum fitness value of the swarm. r_1 and r_2 are random numbers in the interval $[0,1]$. c_1 and c_2 are two learning factors representing the weights of acceleration terms that guide each particle toward $P_{l,best}$ and G_{best} , respectively.

In iteration $t+1$, the position $P_{l,t+1}$ is calculated as:

$$P_{l,t+1} = P_{l,t} + V_{l,t+1} \quad (12)$$

Step 5: Check whether the algorithm is complete. When the number of iterations is less than the maximum number of iterations, repeat step 2, step 3 and step 4. Otherwise, the iteration terminates.

TABLE 2. DNN TASK OFFLOADING STRATEGY BASED ON PSO

Algorithm DNN task offloading strategy based on PSO
1: Initially, set initial values including iteration T, swarm size L, particle velocity, particle position, inertia weight ω , learning factor parameters c_1 and c_2 and other parameter conditions
2: for $i = 1$ to L do
3: while satisfy the constraints do
4: Calculate the fitness value of each particle and update the optimum fitness value of each particle
5: Update the speed and position of each particle
6: end while
7: Update the global optimum fitness value
8: end for
9: Output optimal offloading solution

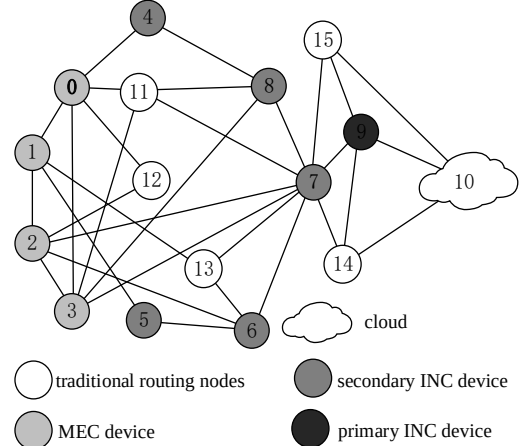


Figure 3. Network topology

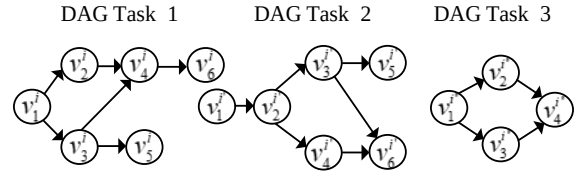


Figure 4. DAG structure for three types of DNN tasks

IV. EXPERIMENTAL RESULTS AND ANALYSIS

We realize PSO algorithm through Python and simulate the offloading scenarios in the computing power network.

We consider the network model in Fig. 1 and on the basis of the traditional computing power network, we use the traditional routing nodes existing in the network shown in Fig. 3. In the network topology, parameters such as links are set randomly. In this simulation, the link is set to 100Mbps. Consider the heterogeneous nodes, we set the computing resource to 2 GHz for each MEC device, to 5 GHz for each primary INC device and to 3 GHz for each secondary INC device. The default unit computation cost of node is set to 1. For cloud, we set the computing resource to 10 GHz and the default unit computation cost to 1.12. The proportion of remaining computing resources β^k is randomized to (0.6,0.8). The default input is three types of DNN tasks with different DAG, as shown in Fig. 4. For inference task i , service access is randomly performed from four arbitrary edge nodes and the task deadline is randomized to (20,75) seconds. The transmission ratio α_{ijz} of each subtask is set to (0.01,0.05) and the amount of transmitted data d_{ij} is randomized to (1.5, 2) Mbit. By default, the distance between subtasks is the number of hops between offloaded nodes and the unit of transmission is set to 0.5. In addition, we set the inertia weight ω to 0.08 and the speed constraint V_{max} to 10. We set the learning factor c_1 to 0.2 and c_2 0.5. We set the swarm size to 10 and the number of iterations to 20 [16]. During the evaluation process, the value of each parameter may change.

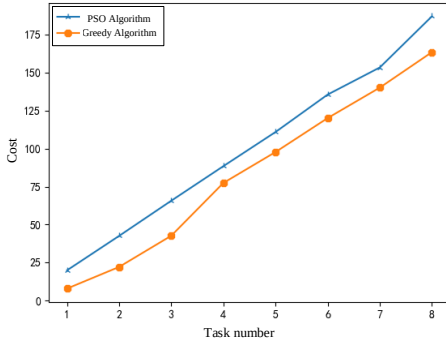


Figure 5. Task number and cost under different algorithms

Fig. 5 shows the total network cost of greedy algorithm and PSO algorithm in the same simulation environment. It can be seen that the cost of the greedy algorithm is always lower than that of PSO algorithm. As the number of input inference tasks increases, the cost difference between the two algorithms decreases and the cost of PSO algorithm tends to the cost of greedy algorithm.

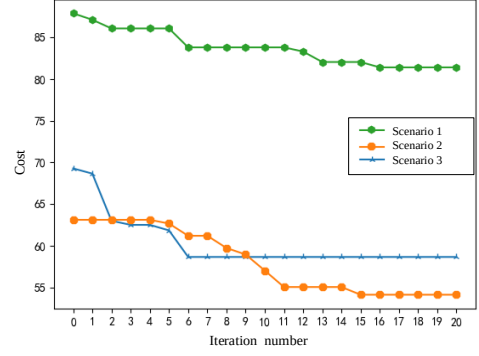


Figure 6. Iteration number and cost under different scenarios

We define three experimental network scenarios. Scenario 1: Traditional network without INC devices. Scenario 2: All network devices have in-network computing ability. Scenario 3: Network devices include INC devices and traditional routing devices without computing ability. We conducted experiments in three network scenarios, as shown in Fig. 6. It can be seen that in scenario 1, subtasks can only be offloaded in MEC and cloud with computing ability. The iteration is easy to converge and the cost is relatively high. In scenario 2, the convergence speed is slow and there are no traditional routing devices in scenario 2, so INC devices can achieve the lowest cost of the network through the pass-by-pass calculation. However, scenario 2 is an idealized model while scenario 3 has a low cost and is more prepared to implement the value in practice.

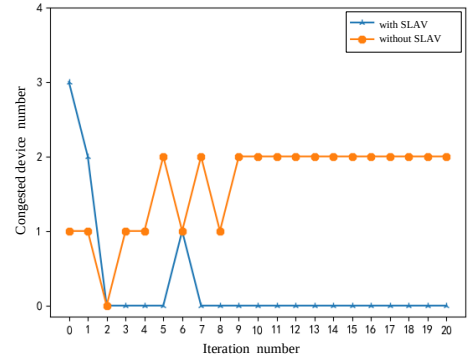


Figure 7. Impact of SLAV on network node congestion

In addition, Fig. 7 verifies that the SLAV parameter plays a role in resource balancing in the global network. If the computing resource of a network device exceeds 80% the device is defined as a congested device. In the task offloading algorithm, the network with SLAV parameter optimization decreases the number of congested nodes and tends to be stable with the increase of iteration times, while the network without SLAV parameter optimization only considers the minimization cost, does not consider the balance of computing resources of nodes in the network.

V. CONCLUSIONS

This paper aims at the multi-source DNN task offloading problem, which combines edge computing, in-network computing and cloud computing. At the same time, it makes full use of the traditional routing nodes in the network with no computing capacity and takes the resource cost as the utility function of the problem optimization. PSO algorithm is used to solve the problem in the offloading scheme optimization, SLAV is introduced and the network resources are offloaded in balance while the quality of service of users is satisfied. Simulated experiment results that there is an optimal deployment in the network, which can make full use of the computing resources and bandwidth resources of the network, significantly reduce the computational pressure and transmission overhead of the whole network, and achieve the balanced offloading of node resources.

ACKNOWLEDGMENT

This work is funded by Beijing University of Posts and Telecommunications-China Mobile Research Institute Joint Innovation Center.

REFERENCES

- [1] Islam A, Debnath A, Ghose M, et al. A survey on task offloading in multi-access edge computing[J]. Journal of Systems Architecture, 2021, 118: 102225.
- [2] Liu J, Ren J, Zhang Y, et al. Efficient dependent task offloading for multiple applications in MEC-cloud system[J]. IEEE Transactions on Mobile Computing, 2021.
- [3] Wu B, Zeng J, Shao S, et al. New Game-theoretic Approach to Decentralized Path Selection and Sleep Scheduling for Mobile Edge Computing[J]. IEEE Transactions on Wireless Communications, 2022.
- [4] He W, Gao L, Luo J. A Multi-Layer Offloading Framework for Dependency-Aware Tasks in MEC[C]//ICC 2021-IEEE International Conference on Communications. IEEE, 2021: 1-6.
- [5] Gao Y, Liu W. Multiple Workflows Offloading Based on Improved Deep Q-Network in Mobile Edge Computing[C]//2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD). IEEE, 2022: 1281-1286.
- [6] Mao N, Chen Y, Guizani M, et al. Graph mapping offloading model based on deep reinforcement learning with dependent task[C]//2021 International Wireless Communications and Mobile Computing (IWCMC). IEEE, 2021: 21-28.
- [7] Sahni Y, Cao J, Yang L, et al. Multihop offloading of multiple DAG tasks in collaborative edge computing[J]. IEEE Internet of Things Journal, 2020, 8(6): 4893-4905.
- [8] Liao H, Li X, Guo D, et al. Dependency-Aware Application Assigning and Scheduling in Edge Computing[J]. IEEE Internet of Things Journal, 2021, 9(6): 4451-4463.
- [9] Lou J, Tang Z, Zhang S, et al. Cost-Effective Scheduling for Dependent Tasks With Tight Deadline Constraints in Mobile Edge Computing[J]. IEEE Transactions on Mobile Computing, 2022.
- [10] Liu J, Ren J, Zhang Y, et al. Efficient dependent task offloading for multiple applications in MEC-cloud system[J]. IEEE Transactions on Mobile Computing, 2021.
- [11] Y. Duan and J. Wu, "Computation Offloading Scheduling for Deep Neural Network Inference in Mobile Computing," 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), 2021, pp. 1-10, doi: 10.1109/IWQOS52092.2021.9521304.



Lizi Hu is working toward the master's degree from the Beijing University of Posts and Telecommunications, Beijing, China. Her research interests include communication and resource allocation.

- [12] Tang Z, Lou J, Zhang F, et al. Dependent task offloading for multiple jobs in edge computing[C]//2020 29th International Conference on Computer Communications and Networks (ICCCN). IEEE, 2020: 1-9.
- [13] Wang H, Wang Y, Taleb T, et al. Special issue on security and privacy in network computing[J]. World Wide Web, 2020, 23(2): 951-957.
- [14] Mai T, Yao H, Guo S, et al. In-network computing powered mobile edge: Toward high performance industrial IoT[J]. IEEE network, 2020, 35(1): 289-295.
- [15] Kahn A B. Topological sorting of large networks[J]. Communications of the ACM, 1962, 5(11): 558-562.
- [16] Tang Z, Lou J, Zhang F, et al. Dependent task offloading for multiple jobs in edge computing[C]//2020 29th International Conference on Computer Communications and Networks (ICCCN). IEEE, 2020: 1-9.



Yuhao Chai is working toward the master's degree from the Beijing University of Posts and Telecommunications, Beijing, China. His research interests include game theory, 5G networks and resource allocation.



Qin Li is a senior researcher of China Mobile Research Institute. She received her master's degree from Chongqing University and has engaged in technical and application research on core network and content network for more than 10 years. Her research interest covers network intelligence, future network architecture and digital twin network.



Wei yuan Li, is a researcher at China Mobile Research Institute. She received a BE degree from Beijing Jiaotong University and a PhD in Information and signaling processing from Chinese Academy of Sciences. Her current research is in intelligence network.



Yong Zhang received the Ph.D. degree from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China in 2007. He is a Professor with the School of Electronic Engineering, BUPT. He is currently the Director of Fab.X Artificial Intelligence Research Center, BUPT. He is the Deputy Head of the mobile internet service and platform working group, China communications standards association. He has authored or coauthored more than 80 papers and holds 30 granted China patents. His research interests include Artificial intelligence, wireless communication, and Internet of Things.