AI Service Placement for Multi-Access Edge Intelligence Systems in 6G

Jiaxin Li, Fuhong Lin, Lei Yang, *Senior Member, IEEE,* and Daochao Huang

Abstract—This paper studies the artificial intelligent (AI) task deployment problem of a multi-access edge intelligent system in a 6G network, in which the cloud server broadcasts the AI program to the edge computing nodes. In particular, task nodes can perform remote processing by offloading AI tasks to cloud servers or other edge computing nodes, or they can perform processing tasks locally. In order to minimize the total computing time and energy consumption of all task nodes and maximize the inference accuracy of AI tasks, we jointly optimize the resource allocation and computing offloading decision of each node by solving a mixed-integer non-linear programming (MINLP) problem. In order to efficiently solve this non-convex problem, we propose an alternating direction multiplier method (ADMM) based algorithm, which effectively decomposes the problem into easy-to-handle MINLP subproblems. Through the proposed ADMM-based algorithm, each task node can use local channel state information (CSI) to optimize its calculation mode and resource allocation, which is more suitable for large-scale networks. The simulation results show that this method is significantly better than other benchmarks in various network environments, and the computational complexity is relatively low.

Index Terms—Multi-access edge computing, 6G, artificial intelligent, computing offloading.

1 Introduction

1.1 Background

With the rapid development of a new generation of information technology (e.g., big data, artificial intelligence), there are more and more real-time, computationally intensive, and complex computing tasks in the Internet of Things system. The data growth in the past decade [1] indicates that 5G will soon be unable to support large-scale Internet of Things (IoT) applications in the future. Therefore, 6G, as an extension technology of 5G, is considered a more powerful tool [2] to support millions or even billions of connected devices with high data rates and low latency. To enable 6G networks to have AI capabilities, mobile and the Internet of Things will continue to generate large amounts of data (e.g., user behavior records, audio and video) that reside at the edge of the network. Driven by this trend, there is an urgent need to push the frontier of artificial intelligence to the edge of the network to fully release the potential of 6G networks.

In addition, massive amounts of data will bring huge computing pressure to the core of the network cloud, and at the same time, the data interaction between cloud edges will also put a huge load on the capacity-constrained fronthaul link. In order to alleviate the above pressure and meet the needs of 6G, multi-access edge computing (MEC) [3], [4], which is an emerging paradigm and placing services and computing tasks at the edge of the network. It has been widely regarded as an indispensable part of the 6G network. The multi-access edge computing architecture successfully reduces the transmission delay and bandwidth loss of the network by offloading storage and computing resources to the edge of the network, and has been widely recognized by the academic community.

At the same time, the integration of AI and MEC, namely edge intelligence (EI) [5], [6], [7], can support edge servers to perform inference and training of AI models, so that edge nodes can easily access the AI application. Specifically, the edge server can use the AI models to handle the computing offloading of the edge node to reduce the computing delay and energy consumption of the node [8]. Moreover, the AI service can be deployed at the edge server or mobile node to perform computing tasks. The major challenge of multi-access edge intelligence systems in AI service placement is the trade-off between network latency, energy consumption, and inference accuracy. Therefore, the issue of AI service placement is worthy of attention and study.

1.2 Related Work

In terms of service placement, X. Zhang *et al.* in [9] proposed to combine service placement and edge server, and designed a joint service placement model and edge server deployment. The goal is to maximize the overall profit of all edge servers under the constraints of the storage capacity and computing power of each edge server, the relationship between edge servers, and the number of edge servers. In [10], L. Chen *et al.* studied the layout of collaborative services in dense small cell networks supported by mobile edge computing to solve a series of challenges faced by MEC systems, such as decentralized coordination, spatial demand coupling, and service heterogeneity. In [11], H.

- *Jiaxin Li and Fuhong Lin are with the Department of Computer and Communication Engineering, University of Science and Technology Beijing (USTB), Beijing, 100083, P. R. China.*
  E-mail: b20200318@xs.ustb.edu.cn, FHLin@ustb.edu.cn
  Corresponding author: Fuhong Lin.
- *Lei Yang is with the Department of Computer Science and Engineering, University of Nevada, Reno, NV 89557, USA.*
  E-mail: leiy@unr.edu.
- *Daochao Huang is with National Computer network Emergency Response Technical Team/Coordination Center of China (CNCERT/CC), Beijing, 100029, P. R. China.*
  E-mail: huangdc@cert.org.cn

Manuscript received xxx xxx, xxx; revised xxx xxx, xxx.

Zhou considered the delay constraint as well as the uncertain resource requirements of heterogeneous computation tasks to minimize the energy consumption of the entire MEC system. They proposed a Q-Learning which is a reinforcement learning method to determine the joint policy of computation offloading and resource allocation. In [12], J. Xu *et al.* studied the decision of joint task offloading and service caching in dense cellular networks supported by mobile edge computing to minimize computational delay. In addition to placing service programs on mobile edge servers [9], [12], mobile nodes can also support local execution of applications. M. Gao *et al.* in [13] proposed a computing offloading scheme that minimizes task processing delay while managing server load reasonably. Among them, both the user and the MEC can execute the application, and the user optimizes the workload to be uninstalled to the MEC server. Y. Yang *et al.* in [14] studied the optimization of the user's unloaded data ratio and MEC computing resource allocation under the delay constraint, and minimized the global energy consumption. However, they all assume that the required service programs for task computation are available at edge server and devices. This assumption is not true in multi-access edge intelligence systems since the underlying AI models typically require continuous update. AI model are trained using historical data for optimal results, and the data in the environment may evolve over time. Such change of data distribution may lead to reduced AI inference performance. Therefore, in order to mitigate model performance degradation, the AI services must be updated regularly using newly collected data, and the updated AI service program is broadcasted to edge server or devices for processing tasks. At the same time, in the real-world AI system, task nodes must have applications to perform processing tasks, and it takes time for cloud servers to transmit applications to edge nodes, and the transmission delay and energy consumption of the system should also be taken into consideration.

Z.h. Lin in [15] studied the service placement problem in MEC system, where the access point places the most up-to-date AI program at user devices to enable local computing/task execution at the user side. To minimize the total computation time and energy consumption of all users, they jointly optimizing the service placement and resource allocation. X. Li in [16] considered an edge intelligence system where multiple end users collaboratively train an AI model under the coordination of an edge server, aiming at minimizing the energy consumption and execution latency of the end users. Similarly, existing studies in MEC systems mainly focus on reducing service delay and energy consumption. For multi-access edge intelligence systems in 6G, the inference accuracy of AI services in MEC systems is also of paramount importance. Our paper studies the AI service placement, considering not only the time and energy consumption of the MEC system, but also the inference accuracy, where we consider the heterogeneous requirements for AI services.

H. Zhou in [17] and [18] develop an effective incentive mechanism to motivate nodes to participate in data offloading, aiming to maximize the revenue of the service provider. Specifically, [11] [17] [18] adopt a learning-based approach to solve the resource allocation problem. In this paper, we adopt the ADMM-based algorithm whose computational complexity and convergence can be theoretically analyzed. In contrast, learning-based optimization algorithms are mostly heuristic. The ADMM algorithm adopted in this paper is much more scalable with provable performance guarantee when the network scale is large.

Moreover, we consider the problem of Churn in communication networks. Churn refers to that nodes frequently join or leave the network, and in [19] [20] [32], Churn is defined as a measure of the dynamic characteristics of the network, such as the node's lifetime. Frequent joining or leaving of nodes may degrade performance of the communication network. For example, it can consume additional network bandwidth and increase the delay. Therefore, it is urgent to consider the survival time of nodes.

1.3 Contributions

In this paper, we focus on the computation time and energy consumption minimization of the MEC system and the inference accuracy maximization of AI service, through the design of AI service placement and binary offloading strategy for resource allocation. In summary, the contribution of this paper is as follows:

First, this paper designs the AI service placement in the MEC system, in which cloud server transmits and places the AI model to mobile edge nodes through the broadcast channel. Specifically, the task node which receives the program can offload partial tasks to the edge computing node or the cloud server for processing, and the remaining tasks can be processed locally. We focus on minimizing the energy consumption and computing time of all task nodes, which are subject to bandwidth, offloading rate, frequency and other constraints. At the same time, we further consider the inference accuracy of AI service and the survival time of nodes.

Second, the optimization problem is formulated as a MINLP problem, which jointly optimizes resource allocation, computational offloading, and service placement. In order to solve this problem, we propose an algorithm based on ADMM, which transforms the MINLP problem into multiple smaller and easier-to-handle subproblems. Therefore, the total computational complexity of ADMM-based algorithms increases linearly with the number of nodes, and it is much more scalable with provable performance guarantee when the network scale is large.

Finally, the simulation results show that the proposed algorithm is significantly better than other benchmarks. At the same time, the proposed method has lower computational complexity and converges faster.

The rest of this article is organized as follows. Section 2 introduces the system model. Section 3 constructs the optimization problem of time and energy consumption. Section 4 proposes a joint optimization algorithm based on ADMM. In Section 5, simulation results are given and the performance of the proposed algorithm is discussed. Finally, the paper is concluded in Section 6.

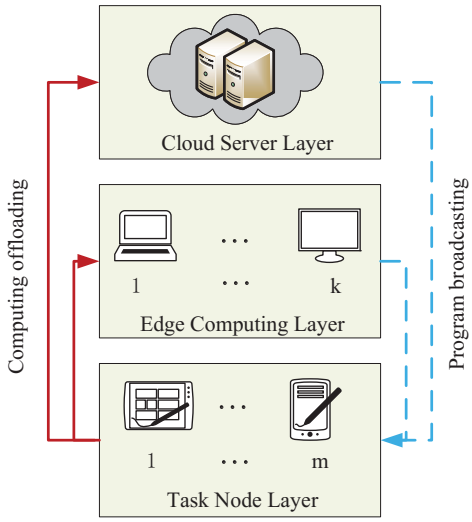


Fig. 1: System model

2 System Model

Consider a cloud server is connected to an access point with a LAN cable, while cloud services are deployed behind wireless access points. task nodes are wirelessly connected to the access point, and processing task edge computing nodes with strong computational power, as shown in Fig. 1. For example, in some exiting works [21] [22], [23], small cloud infrastructures such as cloudlets can be installed behind wireless access points to bring the computing resources closer to nodes. Furthermore, in the future 6G networks, the future wireless scenario is a high-speed network, and the wireless transmission speed is sufficient to support nodes for further computing. Assume that each task node has a certain amount of local data (e.g., images) that needs to be processed by AI services. The cloud server regularly updates the AI model and broadcasts it to the edge and task nodes. The task nodes can use the AI model to process local data, and can also offload local data to a cloud server or edge nodes for processing. We assume that the cloud server broadcasts the AI service program to reduce communication overhead. Moreover, the problem of service placement is a task scheduling problem. In our work, the cloud server first broadcasts the AI model to all nodes, and then the task nodes choose how to complete the task. One option is to offload tasks to the edge nodes. Another option is to offload part of the task to the cloud server. When offloading AI tasks, we consider heterogeneous task requirements on inference accuracy as well as latency. The tasks offloaded to the cloud require higher inference accuracy, while the tasks offloaded to the edge require lower latency.

We assume a frequency division duplex (FDD) mode of operation, in which the uplink computing offloading and downlink program broadcasting are performed on orthogonal frequency bands, denoted as and respectively. In addition, nodes use frequency division multiple access (FDMA) to share uplink bandwidth, and task node applies ( ) bandwidth. In particular, is the uplink wireless channel gain between the cloud server and the node, and is the downlink wireless channel gain between the cloud server and the node.

Assume that the cloud server broadcasts Mbits AI service program with power , In order to ensure that all task nodes in decode correctly, according to the worst-case node in , the cloud server adjusts its rate . It's given by [24]

(1)

where denotes the noise power spectral density and . Then, the time spent on broadcasting AI service programs is [25]

(2)

In this paper, we assume that the AI task is image recognition, where the task of the th user is preprocessed to a resolution of . Let (in pixels) denote the resolution of the new task, where is the weight and height of the task after local preprocessing. Then, the total data volume of the task node can be expressed as , where is the data volume per pixel. The task node offloads bits to the cloud server for computing, and computes bits locally. We can get the offloading ratio according to the local availability of the program. The number of CPU cycles for computing one bit of task data is , which is assumed to be the same for all nodes. Below, we will describe the local computing, cloud computing and edge computing models in detail.

2.1 Local Computing Mode

For the task node , we denote as its local CPU frequency, and the constraints is . Then the local task execution time is [26]

(3)

and the local task execution energy consumption is [27]

(4)

where represents the computing energy efficiency coefficient.

2.2 Cloud Computing Mode

Let be the transmit power of task node . Then the uplink data rate of task node is [24]

(5)

The time it takes to offload the task is [26]

(6)

The energy consumption it takes to offload the task is [27]

(7)

Assume that the cloud server computes the tasks of each task node with CPU frequency . Then the processing task time of task node on the cloud server is [26]

(8)

TABLE 1: Variables list

Definition	Notation
The broadcasting rate of cloud server	r_0
The time spent on broadcasting AI service programs	t_0
The local task execution time of task node m	t_m
The local task execution energy consumption of task node m	e_m
The uplink data rate of task node m	r_m
The time spent on offloading the task to the cloud server	t_m^u
The energy consumption spent on offloading the task to the cloud server	e_m^u
The processing task time of task node m on the cloud server	t_m^c
The transmission data rate of task node m and edge node k	r_m^k
The time spent on offloading the task to the edge node k	$\bar{t}_m^u$
The energy consumption spent on offloading the task to the edge node k	$\bar{e}_m^u$
The time spent on processing tasks at the edge node k	$\bar{t}_m^e$
The energy consumption spent on offloading the task to the edge node k	$\bar{e}_m^e$
The resolution of the task	s_m^2
The summation of the analytics accuracy of the task node m	A_m

2.3 Edge Node Computing Mode

For the edge nodes k, the transmission data rate of task node m and edge node k is [24]

$$r_m^k = a_m W_U \log_2\left(1+\frac{p_m g_m^k}{W_U N_0}\right), \forall m \in M, \tag{9}$$

The time which task node m offload the task to the edge computing node k is [26]

$$\bar{t}_m^u = \frac{\eta_m L_m}{r_m^k}, \forall m \in M, \tag{10}$$

The corresponding energy consumption is [27]

$$\bar{e}_m^u = p_m \bar{t}_m^u = \frac{\eta_m p_m L_m}{r_m^k}, \forall m \in M, \tag{11}$$

The time spent on processing tasks at the edge node k is [26]

$$\bar{t}_m^e = \frac{\eta_m L_m C}{f_k}, \forall m \in M, \tag{12}$$

The corresponding energy consumption is [27]

$$\bar{e}_m^e = \xi f_k^3 \bar{t}_m^e = \xi f_k^2 \eta_m L_m C, \forall m \in M. \tag{13}$$

Actually, the cloud server is much better than the node in terms of transmission power, and the size of the computing result is usually much smaller than the size of the input data. Therefore, the time it takes to download computing results to nodes from the cloud server and other edge computing nodes can be ignored [28], [29].

2.4 Inference Accuracy versus Offloading Data Volume

In multi-access edge intelligence systems, the inference accuracy of tasks is one of the key issues affecting the QoE of IoT devices. According to the existing work [30], under the fixed neural network model, the inference accuracy generally increases with the increase of the input size, but the computing time and energy consumption of task nodes will increase due to the increase of offloading data volume.

According to the accuracy model based on actual experiments obtained in [30] and [31], the inference accuracy is highly dependent on the input size of the DNN model, and the following monotonic non-decreasing function can be constructed, i.e., $f(s_m^2) = 1 - 1.58e^{-6.5\times10^{\ -3}s_m}$. According to the above function, the larger the input size is, the better the precision is. Therefore, we model the analytical accuracy $A_m = \chi(s_m^2)$, where$\chi(s_m^2)$ is a concave function with respect to the data input size s_m^2.

In multi-access edge intelligence systems, the AI service models at edge and cloud are different such that the model sizes can be different, which would result in different inference performance. For the model at edge, the inference accuracy can be low, but the delay and the energy consumption of offloading the data to the edge are low. For the model at cloud, the inference accuracy can be high, but the delay and the energy consumption would be high due to the long distance between the nodes and the cloud. In this paper, we consider the heterogeneous task requirements such that nodes may have different requirements on inference accuracy, delay, and energy consumption.

2.5 Node Lifetime

In this paper, the lifetime of the nodes in the P2P network is considered, which refers to the time period that the nodes in the P2P network experience from joining to leaving, i.e., $Lifetime = time\ of\ leave - time\ of\ join$. Yao et al. [32] studied the characteristics of node heterogeneity in unstructured P2P networks, and calculated the distribution function $H(x)$ of the residual lifetime of any node in the system with the session duration and offline duration of a single node as variables, which is $H(x) = (1+\frac{x}{\beta})^{1-\alpha}$. The definitions of these variables are listed in Table I.

3 PROBLEM FORMULATION

This paper considers the AI service placement problem with heterogeneous task requirements in a multi-access edge intelligence system. According to the task requirements of each task node, the tasks in each task node can be offloaded to either cloud or the edge. The tasks offloaded to the cloud require higher inference accuracy, while the tasks offloaded to the edge require lower latency.

In this paper, we aim to optimize computing time, energy cost (TE) and inference accuracy. We first discuss the two computing modes: 1) joint cloud and local computing, and 2) joint edge and local computing. The computing mode selection is recorded as a binary vector b_m. If $b_m = 1$, the cloud server is selected for joint computing with the task nodes; if $b_m = 0$, the edge nodes and the task nodes are selected for joint computing.

- Case 1:

The entire task of node m is processed by itself and the cloud server, and the local computing delay is

$$t_m^1 = t_0 + t_m = \frac{S}{r_0} + \frac{(1-\eta_m) L_m C}{f_m}, \tag{14}$$

Cloud computing latency is

$$t_m^2 = t_m^u + t_m^c = \frac{\eta_m L_m}{r_m} + \frac{\eta_m L_m C}{f_0}, \tag{15}$$

The total computing time of task node is the maximum one of the local computing and cloud computing delays because local computing and cloud computing are executed in parallel, which given by [33]

(16)

The energy consumption of task node is the sum of cloud computing, task offloading, and the energy consumption of local computing:

(17)

Case 2:

The entire task of node is processed by itself and edge nodes, and the local computing delay is

(18)

The computing delay of other edge computing node is

(19)

The total computing time of node is given by [33]

(20)

The energy consumption of task node is the task offloading, energy consumption of local computing and edge node computing:

(21)

For edge computing nodes , we also consider the survival time of edge nodes . According to [20], the residual lifetime of any surviving edge node in the system obeys Pareto lifetimes, so the distribution function is given by

(22)

From the above discussion, it can be seen that the total computing time of task node is

(23)

The total energy consumption of task node is

(24)

The summation of the analytics accuracy of the task node is

(25)

Next, we define the performance metric TE as the weighted sum of computing time and energy consumption , where and are the weighting factors that satisfy . The objective function can be viewed as a weighted sum method for general multi-objective optimization problems. As stated in Proposition 3.9 in [36], in the case of positive weights, minimizing P1 can effectively solve the multi-objective optimization problem. Furthermore, the weights themselves reflect the relative importance (preference) between energy consumption, latency, and precision accuracy. For example, when the battery power of the task node is low, and the user only cares about the energy consumption, the node can set . In recent some exiting work [34] [35], such weighted sum methods have been widely used. We mainly use joint optimization computing mode selection , computing of offload ratio , local CPU frequency , uplink bandwidth allocation and the resolution of the task to minimize the total TE and maximize the total analytics accuracy of the task. By introducing auxiliary variables and , , the problem is formulated as:

(26a)

(26b)

(26c)

(26d)

(26e)

(26f)

(26g)

(26h)

(26i)

(26j)

(26k)

(26l)

where (26b) represents the offloading ratio constraints, (26c) represents local CPU frequency constraints. (26d) and (26e) are restrictions on bandwidth allocation.(26j)-(26k) are the constraints on the auxiliary variables and . (26l) represents the minimum analytics accuracy requirement constraints.

Problem (P1) is non-convex because the objective function and constraints are both non-convex. However, we can observe that given, (P1) is jointly convex for . Therefore, we can use CVX [37] which are convex optimization tools to get the optimal solution . In order to obtain the optimal value

, we can list all the possible which is one of the most direct methods, and then find a pair of minimum target values on the feasible space of which can use a multi-dimensional search. However, the multi-dimensional search method is computationally infeasible, because even if is small, for example, , the search space is very large, not to mention the exhaustive search for the optimal , and its computational complexity increases exponentially with the increase of .

Proposition 1. *The problem of AI service placement (P1) is NP-hard.*

Proof. We first briefly describe the capable facility location problem (CFLP) [38] (A comparison of heuristics and relaxations for the capacitated plant location problem) which is a well-known NP-hard problem. Then we reduce the CFLP to the problem P1 and show that the problem P1 is also NP-hard.

Description of CFLP. In the capable facility location problem, we suppose that there are facilities and customers. Let to represent the needs of customer , each customer can divide its needs into multiple parts which are sent to different facilities for production; at the same time, we assume that each facilities has a production capacity , which is the maximum amount of product that can be produced by facilities . In order to minimize the sum of the opening cost and the assignment cost and meet the total demand assigned to a facility must not exceed its capacity. We wish to choose: (i) which of the facilities to open and (ii) the assignment of customers to facilities.

Reducing CFLP to P1. By treating each instance of CFLP as a special case of problem P1, we can reduce CFLP to problem P1.

The consumer with demand in CFLP is mapped to the node with computing demand in problem P1.
The opened facility with production capacity in CFLP is mapped to the cloud server with node lifetime limit in problem P1.
The opening cost of facilities in CFLP is mapped to the cost of computing time in problem P1.
The cost of shipping products in CFLP is mapped to the total energy cost in problem P1.

Hence, the NP-hard problem CFLP can be reduced to problem P1. Thus, problem P1 is NP-hard. □

4 ADMM-BASED JOINT OPTIMIZATION ALGORITHM

4.1 Reformulation of (P1)

In this section, an ADMM-based algorithm [39] is proposed to solve (P1). The main thought is to decompose (P1) into parallel easier-to-handle and smaller MINLP problems. As shown in (1) and (2), the downloading time is decided by the choice of task nodes , which is . Then can be equivalently used as the optimization variable of (P1), and then add the following constraints on (P2):

(27)

We can observe that and are coupled under constraints (27) and (26e). In order to decompose (P1), we introduce artificial variables and reformulate (P1) as:

(28a)

(28b)

(28c)

(28d)

(28e)

(28f)

(28g)

where

and

(29)

Besides,

(30)

where

(31)

Now we can use ADMM [39] to effectively decompose (28). By introducing multipliers under constraint (28f), the augmented Lagrangian of (28) can be expressed as:

–

–

where , and , and is a fixed step size. Correspondingly, the dual function is

(32)

and the dual problem is

(33)

4.2 ADMM Iterative Processing

The ADMM algorithm [39] solves the dual problem (29) by iteratively updating , , and . We denote the values in the th iteration as . Then, in the th iteration, the variables are updated in the following order:

Step 1: Solving problem (33) by updating the local variables which using one-dimensional search method [40] in the th iteration.

we update the local variables which is given by

(34)

We notice that (34) can be decomposed into parallel subproblems, and each subproblem is solved:

(35)

where

– – (36)

For and , (35) can be equivalent to

– – (37)

and

– (38)

Since variable and variable are coupled, subproblems (37) and (38) are non-convex. However, given , the subproblem is a strictly convex problem and is easy to solve. Therefore, a simple one-dimensional search method can be used, such as the golden section search [40], which can find the optimal value and the corresponding optimal value . Finally, we can choose or to make the target value in (33) smaller as , the corresponding optimal solution is . By solving suproblems, the optimal solution of (32) is obtained as . Given the accuracy parameter , the golden section search method requires — iterations to solve each subproblem (37) and (38) respectively. We assumed that the complexity of solving each subproblem (35) is and is proportional to the number of subproblems, thus the overall computational complexity of Step 1 is . Moreover, the subproblems are solved in parallel, and the computational time of step 1 remains unchanged during parallel computing.

Step 2: Solving problem (33) by updating the global variables which using bisection search method [41] in the th iteration.

After obtaining , we update the global variable as:

(39)

According to the definition of in (31), must be kept as optimal. Therefore, we can equivalently express (39) as the following convex optimization problem:

– –

(40)

Then, the closed form of the optimal value can be directly obtained as:

(41)

where . Then we denote the Lagrangian multiplier which the constraint is , we can get the optimal closed form of as:

(42)

Since is non-increase with , we can get the optimal by performing a bisection search over , where is a large enough value until is satisfied, and then compare the result with the condition of , which the condition that . The specific algorithm can refer to [41]. Thus, the computational complexity of step 2 to solve (40) is . Moreover, under given Lagrangian multiplier , we can get for the nodes in parallel, so the computational time of step 2 remains unchanged when performing parallel computing.

Step 3: Solving problem (33) by updating the multipliers in the th iteration.

After obtaining the variable , it's should be considered the variable to maximize , which can be updated the multiplier as

(43)

Obviously, the computational complexity of step 3 is also . Similarly, we update (43) in parallel for nodes, the computational time of step 3 also remains unchanged when performing parallel computing.

In short, we perform the three continuous steps until the preset stop conditions are met. Generally, the stopping conditions is determined by two thresholds: absolute tolerance and relative tolerance . Algorithm 1 gives the pseudo code based on ADMM algorithm (P1). The dual problem (33) is convex at , so the convergence of the algorithm is guaranteed. In addition, the convergence of the ADMM-based algorithm is not sensitive to the choice of step size [42], we set without loss of generality. The complexity of an ADMM iteration is because the complexity of each of the three steps is respectively. Moreover, during parallel computing, the computational time of the above three steps remain unchanged, thus the computational time of one ADMM iteration remains unchanged and the complexity increases linearly with .

Algorithm 1 ADMM-Based Algorithm

1: **Initialize** ; ; ; ; ; ; ;
2: **Repeat**
3: **for** each node **do**
4: Update local variables by solving (34);
5: **end for**
6: Update global variables by solving (39);
7: Update multipliers by solving (42);
8: ;
9: **Until** and ;
10: **return** as an approximate solution to (P1).

5 Simulation Results

This section evaluate the performance of the proposed algorithm through numerical simulations. In the simulations, it is supposed that the uplink and downlink bandwidth as , and the noise power spectral density is . It is assumed that the average channel gain follows to the free-space path loss model , where the antenna gain is , the carrier frequency is , and the distance from node to the cloud is . The downlink channel gain follows the Rayleigh fading channel model , where is an independent exponential random variable with unit mean. For the sake of briefness, it is assumed that the downlink and uplink downlink channel gains of the nodes are equal. Assuming that the distance between nodes and cloud server is equal and m. In order not to lose generality, it is assumed that the weighting factors of all nodes are the same. Pareto parameter , , . The part of simulation parameters are summarized in Table II. For performance comparison, the following three representative benchmarks are be considered [43]:

1) Cloud only: all the nodes offload their tasks to the cloud, .
2) Node only: all the nodes perform computations locally, .
3) Edge nodes only: all the nodes offload their tasks to the edge node, and .

TABLE 2: Simulation parameter setting

Parameters	Notation	Value
Weighting factors of node m	$\mu_m^T = \mu$	0.1
Weighting factors of node m	$\mu_m^E = 1 - \mu$	0.9
The number of CPU cycles	C	1000
CPU frequency of cloud	f_0	10GHz
The computing energy efficiency coefficient	ξ	10^{-28}
The power of cloud	p_0	1W
The transmit power of node m	p_m	0.1W
The local CPU frequency	f_m	1GHz
The remaining edge nodes	k	3
The input size of node m	s_m	400*400
The data volume per pixel	σ	32

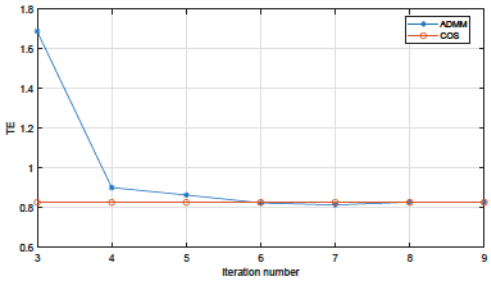


Fig. 2: The convergence performance of the proposed ADMM-based algorithm

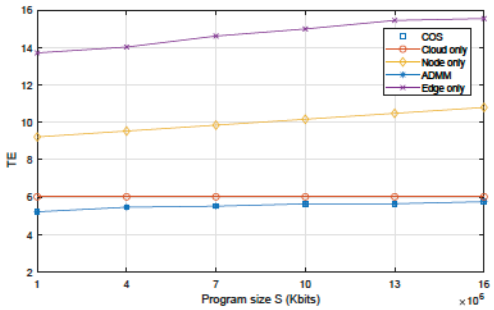


Fig. 3: Total TE versus the program size S

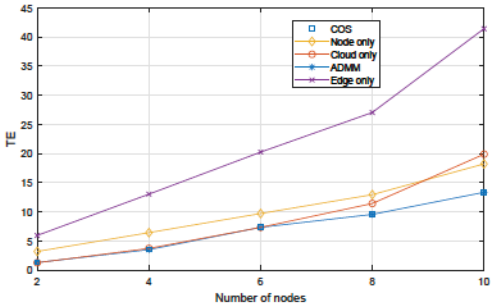


Fig. 4: Total TE versus the number of nodes M

4) Centralized offloading scheme (COS): Using the interior-point method to solve problem (26), which can find the optimal computation offloading decisions of task nodes [44].

The convergence performance of the ADMM-based algorithm proposed in this paper and COS is shown in Figure 2. Here, the number of task nodes is set to 2. We observe that the ADMM-based algorithm gradually decreases in the first 6 iterations, and reaches a stable state after 7 iterations, showing that the algorithm based on ADMM-based proposed converges faster. In addition, the total energy consumption of the algorithm after convergence is close to COS, indicating that the algorithm can achieve relatively good performance after multiple distributed iterations.

In Fig. 3, we set the following parameters, the path loss exponent is $d_e = 3.5$, the remaining edge computing nodes $K = 3$, and the distance between the node and the remaining edge nodes is $d_{m,k} = 200 + 10\,(|m - k|)$. We compared the TE performance achieved by different schemes when the program size S changes. We observe that the proposed TE performance based on the ADMM method is lower than the other three benchmarks. This shows the benefits of joint optimization of service placement, computational offloading, and resource allocation for all nodes. Except that the program size S has no effect on the total TE of the cloud-only and edge node-only computing solution, the total TE of all solutions increases as S increases. This is because task nodes in the cloud-only and edge-only computing solutions do not need to download business programs and need offload all tasks to cloud and edge nodes. In addition, we observe that when S becomes larger, the proposed method tends to cloud computing. This shows that nodes tend to offload computing tasks when the cost of downloading programs is greater than the benefit of local computing or when the offloading delay is low.

In Fig. 4, we set $d_e = 3.5$, $S = 8\,Mbits$. We show the TE performance of different schemes when the node m is from 2 to 10. We observe that as m increases, the total TE increases, and the total TE achieved by the proposed algorithm is getting lower and lower than the three benchmarks. This is because in these three benchmarks, as m increases, the uplink/downlink bandwidth which allocates each node becomes smaller, and each tasks node needs more time to offload computing tasks and download programs from the edge servers.

In Fig. 5, we set $d_e = 3.5$, $S = 32\,Mbits$. We show the TE of different schemes when the distance d_m between the cloud server and the node changes. We observe that as the distance increases, the TE also increases, and the ADMM-based scheme is lower than the three benchmark schemes mentioned. This is because, as the distance becomes larger, the node needs more time to download the application, which consumes more energy.

Fig. 6 shows the relationship between TE and the computing energy efficiency coefficient, where $d_e = 3.5$, $S = 32\,Mbits$. We observe that the proposed algorithm TE increases with the increase of the energy efficiency coefficient. This is because as the energy efficiency coefficient increases, the energy consumption of our local calculations also increases. Among them, the cloud-only computing solution does not change energy efficiency, which can be derived from the previous formula. We also noticed that the node-only and the edge node-only computing scheme increases with the increase in energy efficiency, which can be observed in Figure 7, and the algorithm we proposed

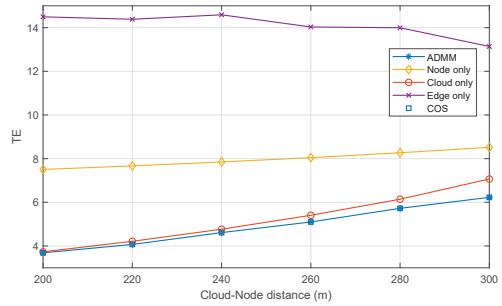


Fig. 5: Total TE versus the distance between Cloud and Node

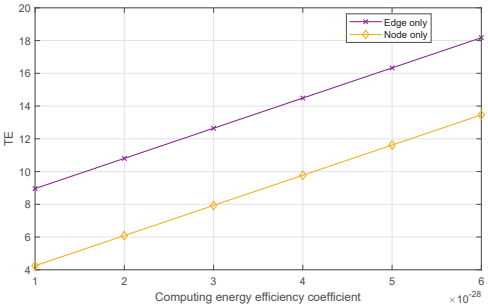


Fig. 7: Total TE versus the computing energy efficiency coefficient

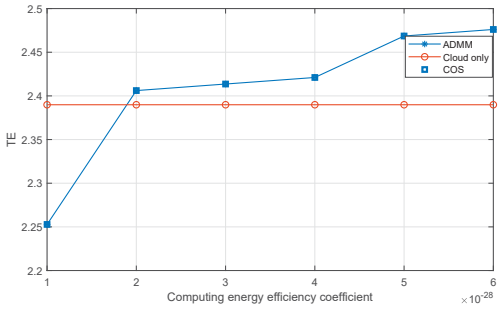


Fig. 6: Total TE versus the computing energy efficiency coefficient

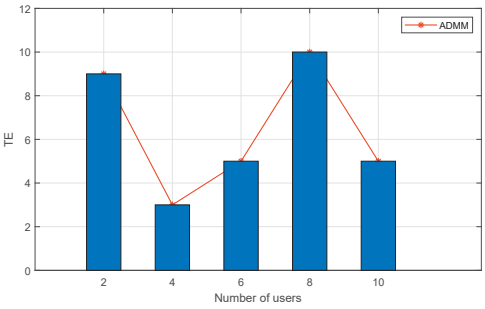


Fig. 8: Average iterative number of algorithm 1 versus the number of nodes

is lower than the node-only and the edge-only computing scheme.

In Fig. 8, it describes the computational complexity of the proposed ADMM-based algorithm. Here, we set , , and then plot the average number of iterations of Algorithm 1 when the number of nodes changes. Interestingly, we observe that the ADMM-based method has an almost constant number of iterations under different within the consideration range, that is, . In addition, since the computational complexity of one ADMM iteration is , the overall computational complexity of the ADMM-based method is also . Therefore, the application of ADMM-based algorithm in the large-scale Internet of Things where the network scale dominates the overall complexity is effective.

Fig. 9 shows the impact of on the computing time and energy consumption of different algorithms. When increases, the MEC system emphasizes the impact of computation time on performance. As a result, the ADMM-based algorithm trades the energy consumption for the computing time. This simulation result also shows that, as compared to the baseline algorithm, the algorithm reaches the minimum TE when the weights change.

6 Conclusion

This paper studies the task deployment problem under the multi-user mobile edge intelligent system, in which the cloud server broadcasts the AI program to the edge computing node. In particular, task nodes can perform remote processing by offloading tasks to cloud servers or other edge computing nodes, or perform processing tasks locally. In order to pursue the minimum total computing time and energy consumption of all task nodes, a MINLP problem is proposed by jointly optimizing the resource allocation and computing offloading decision of each node. After that, we designed an algorithm based on ADMM to solve the non-convexity problem, which effectively decomposes the problem into easier-to-handle MINLP subproblems. Through the proposed ADMM-based algorithm, each task node can optimize its calculation mode and resource allocation, which is more suitable for large-scale networks. The simulation results show that this method is significantly better than other representative benchmark methods in various network environments. At the same time, the computational complexity is relatively low, and it is suitable for large-scale networks.

Acknowledgments

This work was supported in part by the National Science Foundation Project of P. R. China (No. 61931001), NSF under Grants IIS-1838024, CNS-1950485, and OIA-2148788.

References

[1] G. Liu et al., "Vision, requirements and network architecture of 6G mobile network beyond 2030," *China Communications*, vol. 17, no. 9, pp. 92-104, Sept. 2020.

[2] X. Tang et al., "Computing power network: The architecture of convergence of computing and networking towards 6G requirement," *China Communications*, vol. 18, no. 2, pp. 175-185, Feb. 2021.

[3] N. Abbas, Y. Zhang, A. Taherkordi and T. Skeie, "Mobile edge computing: a survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450-465, Feb. 2018.

[4] M. Muniswamaiah, T. Agerwala and C. C. Tappert, "A survey on cloudlets, mobile edge, and fog computing," *IEEE Access*, vol. 5, pp. 9348-9358, May. 2017.

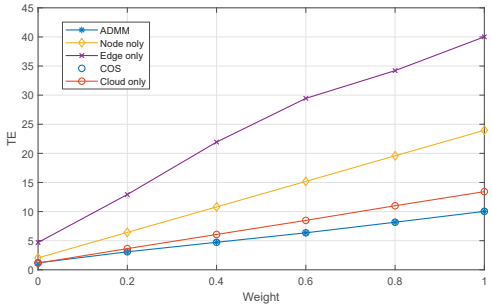


Fig. 9: Total TE versus

[5] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar and A. Y. Zomaya, "Edge intelligence: the confluence of edge computing and artificial intelligence," *IEEE Internet Things J.*, vol. 7, no. 8, pp.7457-7469, Aug. 2020.

[6] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo and J. Zhang, "Edge intelligence: paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738-1762, Aug. 2019.

[7] K. Jiang, C. Sun, H. Zhou, X. Li, M. Dong and V. C. M. Leung, "Intelligence-Empowered Mobile Edge Computing: Framework, Issues, Implementation, and Outlook," *IEEE Network*, vol. 35, no. 5, pp. 74-82, 2021.

[8] H. Hu and C. Jiang, "Edge intelligence: challenges and opportunities," *2020 International Conference on Computer, Information and Telecommunication Systems (CITS)*, vol. 6, pp. 1-5, Oct. 2020.

[9] X. Zhang, Z. Li, C. Lai and J. Zhang, "Joint edge server placement and service placement in mobile edge computing," *IEEE Internet Things J.*, pp. 1-1, Nov. 2021.

[10] L. Chen, C. Shen, P. Zhou and J. Xu, "Collaborative Service Placement for Edge Computing in Dense Small Cell Networks," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 377-390, Feb. 2021.

[11] H. Zhou, K. Jiang, X. Liu, X. Li and V. C. M. Leung, "Deep Reinforcement Learning for Energy-Efficient Computation Offloading in Mobile-Edge Computing," *IEEE Internet Things J.*,vol. 9, no. 2, pp. 1517-1530, Jan. 2022.

[12] J. Xu, L. Chen and P. Zhou, "Joint Service Caching and Task Offloading for Mobile Edge Computing in Dense Networks," *IEEE INFOCOM 2018 - IEEE Conf. on Computer Commun.*, pp. 207-215, 2018.

[13] M. Gao et al., "Computation Offloading with Instantaneous Load Billing for Mobile Edge Computing," *IEEE Trans. on Services Comput.*, pp. 1-1, May. 2020.

[14] Y. Yang, Y. Hu and M. C. Gursoy, "Deep Reinforcement Learning and Optimization Based Green Mobile Edge Computing," *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1-2, 2021.

[15] Z. Lin, S. Bi and Y. -J. A. Zhang, "Optimizing AI Service Placement and Resource Allocation in Mobile Edge Intelligence Systems," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7257-7271, Nov. 2021.

[16] X. Li, S. Bi and H. Wang, "Optimizing Resource Allocation for Joint AI Model Training and Task Inference in Edge Intelligence Systems," *IEEE Wireless Communications Letters*,vol. 10, no. 3, pp. 532-536, Mar. 2021.

[17] H. Zhou, T. Wu, H. Zhang and J. Wu, "Incentive-Driven Deep Reinforcement Learning for Content Caching and D2D Offloading," *EEE Journal on Selected Areas in Communications*,vvol. 39, no. 8, pp. 2445-2460, Aug. 2021.

[18] H. Zhou, X. Chen, S. He, J. Chen and J. Wu, "DRAIM: A Novel Delay-Constraint and Reverse Auction-Based Incentive Mechanism for WiFi Offloading," *IEEE Journal on Selected Areas in Communications*,vol. 38, no. 4, pp. 711-722, Apr. 2020.

[19] D. Leonard, Z. Yao, V. Rai and D. Loguinov, "On Lifetime-Based Node Failure and Stochastic Resilience of Decentralized Peer-to-Peer Networks," *IEEE/ACM Transactions on Networking*,vol. 15, no. 3, pp. 644-656, June 2007.

[20] Z. Yao, X. Wang, D. Leonard and D. Loguinov, "On Node Isolation Under Churn in Unstructured P2P Networks with Heavy-Tailed Lifetimes," *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pp. 2126-2134, 2007.

[21] M. Chen, S. Guo, K. Liu, X.Liao and B. Xiao, "Robust Computation Offloading and Resource Scheduling in Cloudlet-Based Mobile Cloud Computing," *IEEE Transactions on Mobile Computing*, vol. 20, no. 5, pp. 2025-2040, 2021.

[22] S. Yang, F. Li, M. Shen, X. Chen, X. Fu and Y. Wang, "Cloudlet Placement and Task Allocation in Mobile Edge Computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5853-5863, 2019.

[23] H. Chen, D. Zhao, Q. Chen and R. Chai, "Joint Computation Offloading and Radio Resource Allocations in Small-Cell Wireless Cellular Networks," *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 3, pp. 745-758,, 2020.

[24] 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.*, pp. 1-1, Sept. 2021.

[25] G. Peng, H. Wu, H. Wu and K. Wolter, "Constrained Multiobjective Optimization for IoT-Enabled Computation Offloading in Collaborative Edge and Cloud Computing," *IEEE Internet Things J.*,vol. 8, no. 17, pp. 13723-13736, Sept. 2021.

[26] L. Lei, H. Xu, X. Xiong, K. Zheng and W. Xiang, "Joint Computation Offloading and Multiuser Scheduling Using Approximate Dynamic Programming in NB-IoT Edge Computing System," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5345-5362, Jun. 2019.

[27] S. Hu and G. Li, "Dynamic Request Scheduling Optimization in Mobile Edge Computing for IoT Applications," *IEEE Internet Things J.*,vol. 7, no. 2, pp. 1426-1437, Feb. 2020.

[28] L. Huang, S. Bi, and Y. J. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581-2593, Nov. 2020.

[29] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784-1797, Mar. 2018.

[30] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," *IEEE INFOCOM 2018*, pp. 756-764, 2018.

[31] W. Sun, J. Liu, and Y. Yue, "AI-enhanced offloading in edge computing: When machine learning meets industrial IoT," *IEEE Netw.*, vol. 33, no. 5, pp. 68-74, 2019.

[32] Z. Yao, D. Leonard, X. Wang and D. Loguinov, "Modeling Heterogeneous User Churn and Local Resilience of Unstructured P2P Networks," *Proceedings of the 2006 IEEE International Conference on Network Protocols*, pp. 32-41, 2006.

[33] B. Lin, X. Lin, S. Zhang, H. Wang and S. Bi, "Computation Task Scheduling and Offloading Optimization for Collaborative Mobile Edge Computing," *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 728-734, 2020.

[34] Q. Wei, Z. Zhou and X. Chen, "DRL-Based Energy-Efficient Trajectory Planning, Computation Offloading, and Charging Scheduling in UAV-MEC Network," *2022 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 1056-1061, 2022.

[35] H. Ma, P. Huang, Z. Zhou, X. Zhang and X. Chen, "GreenEdge: Joint Green Energy Scheduling and Dynamic Task Offloading in Multi-Tier Edge Computing Systems," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 4322-4335, 2022.

[36] M. Ehrgott, *Multicriteria Optimization*, New York, NY, USA: Springer, 2006.

[37] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar. 2014.

[38] G. Cornuéjols, G. Nemhauser, and L. Wolsey, "The uncapicitated facility location problem," *Cornell University Operations Research and Industrial Engineering*, 1983.

[39] G. Zhang and R. Heusdens, "Bi-alternating direction method of multipliers over graphs," *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3571-3575, 2015.

[40] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge,U.K.: Cambridge Univ. Press, 2004.

[41] Z. Lin, S. Bi and Y. -J. A. Zhang,"Optimizing AI Service Placement and Resource Allocation in Mobile Edge Intelligence Systems," *IEEE Trans. Wireless Commun.*, vol. 20,no. 11, pp. 7257-7271, Nov. 2021.

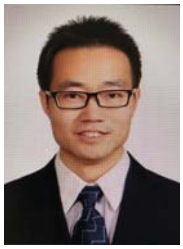
[42] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson,"Optimal parameter selection for the alternating direction method of multi-

pliers (ADMM): Quadratic problems," *IEEE Trans. Autom. Control*, vol. 60,no. 3, pp. 644C658, Mar. 2015.
[43] X. Li, S. Bi and H. Wang, "Optimizing Resource Allocation for Joint AI Model Training and Task Inference in Edge Intelligence Systems," *IEEE Wireless Communications Letters*, vol. 10,no. 3, pp. 532-536, March 2021.
[44] C. Chi, W. Li, and C. Lin, *Convex optimization for signal processing and communications: From fundamentals to applications.* Boca Raton,FL, CRC Press, Feb. 2017.

Jiaxin Li received the M.S. degree from the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China, in 2020. She is currently pursuing the Ph.D. degree with Department of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, P. R. China. Her current research interests include multi-access Edge computing and artificial intelligence.

Fuhong Lin received his M.S. degree and Ph.D. degree from Beijing Jiaotong University, Beijing, P. R. China, in 2006 and 2010, respectively, both in Electronics Engineering. Now he is a professor in Department of Computer and Communication Engineering, University of Science and Technology Beijing, P. R. China. His research interests include Edge/Fog Computing, Network Security, and AI. His two papers won "Top 100 most Cited Chinese Papers Published in International Journals" in 2015 and 2016. He won "Provincial and Ministry Science and Technology Progress Award 2" in 2017, 2019 and 2021.

Lei Yang (Senior Member, IEEE) is an associate professor with the Department of Computer Science and Engineering at University of Nevada, Reno. He received his Ph.D. degree from the School of Electrical Computer and Energy Engineering, Arizona State University, Tempe, in 2012 and was a postdoctoral scholar at Princeton University and an assistant research professor with the School of Electrical Computer and Energy Engineering, Arizona State University. His research interests include big data analytics, AI/ML for cyber-physical systems, edge intelligence in IoT and 5G, data privacy and security. His research was featured in National Science Foundation Science360 News. His papers have won the Best Paper Award Runner-Up award at IEEE INFOCOM 2014 and the Best Paper Award at IEEE PES General Meeting 2022. He serves as the Associate Editor of the IEEE Transactions on Wireless Communications.



Daochao Huang received the Ph.D. degree from Beijing Jiaotong University, Beijing, P. R. China, 2013, in Electronic Engineering. His research interests include data center networking, software defined network, joint/edge computing and network security.