IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON DEEP LEARNING: SECURITY AND FORENSICS RESEARCH ADVANCES AND CHALLENGES

Received April 25, 2019, accepted May 15, 2019, date of publication May 23, 2019, date of current version June 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2918585

A Heuristic Offloading Method for Deep Learning Edge Services in 5G Networks

XIAOLONG XU[1,2,3], DAOMING LI[1], ZHONGHUI DAI[1], SHANCANG LI[4], AND XUENING CHEN[5]

[1]School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China
[2]Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science and Technology, Nanjing, China
[3]Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology, Nanjing University of Information Science and Technology, Nanjing, China
[4]University of the West of England, Bristol BS16 1QY, U.K.
[5]Student Affairs Office, Qufu Normal University, Jining, China

Corresponding author: Xuening Chen (709403073@qq.com)

This work was supported in part by the National Science Foundation of China under Grant 61702277, and in part by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) Fund.

ABSTRACT With the continuous development of the Internet of Things (IoT) and communications technology, especially under the epoch of 5G, mobile tasks with big scales of data have a strong demand in deep learning such as virtual speech recognition and video classification. Considering the limited computing resource and battery consumption of mobile devices (MDs), these tasks are often offloaded to the remote infrastructure, like cloud platforms, which leads to the unavoidable offloading transmission delay. Edge computing (EC) is a novel computing paradigm, capable of offloading the computation tasks to the edge of networks, which reduces the transmission delay between the MDs and cloud. Therefore, combining deep learning and EC is expected to be a solution for these tasks. However, how to decide the offloading destination [cloud or deep learning-enabled edge computing nodes (ECNs)] for computation offloading is still a challenge. In this paper, a heuristic offloading method, named HOM, is proposed to minimize the total transmission delay. To be more specific, an offloading framework for deep learning edge services is built upon centralized unit (CU)-distributed unit (DU) architecture. Then, we acquire the appropriate offloading strategy by the origin-destination ECN distance estimation and heuristic searching of the destination virtual machines for accommodating the offloaded computation tasks. Finally, the effectiveness of the scheme is verified by detailed experimental evaluations.

INDEX TERMS Edge computing, deep learning, computation offloading, 5G, cloud.

I. INTRODUCTION

As a network technology, Internet of Things (IoT), connecting ubiquitous Mobile Devices (MDs) with sensors via wireless networks, is extended from Internet technology [1]. IoT has been applied in many fields widely, including communication device, healthcare system, logistics management, etc. Currently, massive applications are operating through IoT in these domains, and some of these applications, such as virtual reality, augmented reality and video processing, need some properties that traditional IoT is unable to provide, such as low latency, high throughput and so on [2].

Fortunately, 5G can make up the shortfall in IoT. 5G connects innumerable smart devices, realizing data sharing and interaction. Besides, the basic idea of internet of everything could be provisioned by 5G to enlarge the scope of coverage of IoT, where billions of MDs are connected to the Internet [2]. In terms of speed, 5G runs nearly hundred times faster than 4G [3]. With these observations, 5G basically meets the needs of traditional IoT for computation and data offloading. Benefit from 5G, the real-time applications can acquire the processing resources in time.

With the evolution of 5G, the mobile applications have a higher requirement of computing power to MDs. The battery level of MDs consumes fast when MDs operate computation-intensive applications and such applications can be split into many computation tasks. Currently offloading these computation tasks from MDs to the remote infrastructure with computing power is a fashion way for resource provisioning. One of the most popular offloading strategy is offloading tasks to the cloud platform due to its elastic and unlimited resource [4], [5]. End users are able to visit and hire

The associate editor coordinating the review of this manuscript and approving it for publication was Kim-Kwang Raymond Choo.

the infrastructure resources from cloud conveniently through Internet, which provides favorable conditions for data sharing between these users. Meanwhile, considering the long distance between cloud and MDs, transmission delay of computation tasks should be taken into account [6].

The high-speed mobile networking of 5G and the improvement of IoT promote the development of big data applications. Many applications in 5G, such as face recognition, natural language processing and so on, are able to operate in the terminal [7]. As a powerful analysis tool for big data, deep learning could extract accurate information from the original data of terminal devices. Nowadays, deep learning is applied into IoT and mobile applications for achieving abundant of early results. Considering the limited performance of data transfer networks, transmitting massive data to Cloud for deep learning will consume much energy and produce much transmission delay, which reduces the efficiency of deep learning tasks. In [8], the challenge of cloud-centric system was considered and an autonomous and incremental computing framework and architecture for deep learning based IoT applications was proposed to tackle it. Meanwhile, with the fashion of Edge Computing (EC), offloading deep learning tasks from Cloud to the edge of network becomes possible, which can effectively shorten the transmission delay.

EC is a novel computing paradigm which is powerful to offload the computation tasks of MDs to the Edge Computing Nodes (ECNs) for performing, to alleviate the transmission delay [9]. Taking advantage of the EC paradigm, network load and the transmission delay of deep learning tasks are both optimized, which makes positive contribution to ease the computational burden of MDs and improve life of battery [10].

However, if all the deep learning tasks in MDs are only offloaded to the edge, ECNs would suffer from overload in all probability [11] [12]. On the other hand, if all these tasks are offloaded to cloud, they will consume a mass of transmission delay and the big number of these tasks will bring challenges of cost and energy consumption for cloud [9], [13]–[15]. In order to avoid overload of ECNs, reduce the renting cost of cloud services, as well as alleviate transmission delay between MDs and cloud, how to choose the best offloading destination of the deep learning tasks from cloud and ECNs is still a challenge. Facing with this challenge, a Heuristic Offloading Method (HOM) for deep learning edge services in 5G networks is proposed.

Specially, our main contributions are as follow:

- Present an offloading framework for deep learning edge services in 5G networks.
- Realize the optimal computation offloading by heuristically placing the deep learning tasks to the appropriate computing infrastructure.
- Conduct specific simulation experiment to evaluate the effectiveness of HOM.

The following is the arrangement for the rest of this paper. Section 2 displays an offloading framework for deep learning edge services and the offloading time model of this framework. The Heuristic Offloading Method is devised in Section 3. Simulation results are provided in Section 4. Section 5 is the related work. Section 6 exhibits what we conclude for this paper.

TABLE 1. Symbols and corresponding descriptions.

Symbol	Description
N	The number of MDs
P	The total deep learning tasks that MDs produce
da_i	The size of deep learning task p_i
t_i	The time taken to execute deep learning task p_i
M	The number of DUs
K	The number of ECNs
e_k	The number of VMs that c_k contains
I_m^k	The binary variable reflecting the relationship between c_k and d_m
f_n^m	The binary variable reflecting the relationship between d_m and p_n
T_n	The offloading time of p_n
T_{all}	The total offloading time of P

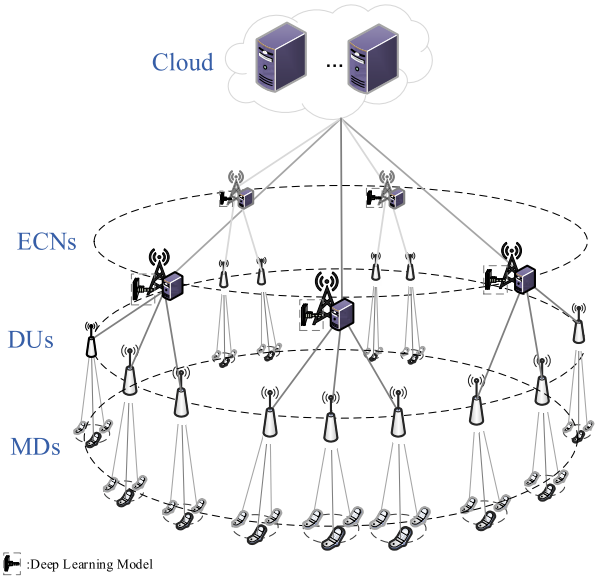


FIGURE 1. Offloading framework built upon CU-DU architecture.

II. SYSTEM MODEL

In this section, an offloading framework for deep learning edge services in 5G networks is considered. In this framework, some formalized concepts are defined for the sake of building a computational migration optimization model. Table 1 presents the key notations of optimization model and the corresponding descriptions.

A. AN OFFLOADING FRAMEWORK FOR DEEP LEARNING EDGE SERVICES IN 5G NETWORKS

As shown in Fig. 1, an offloading framework is built upon Centralized Unit (CU)-Distributed Unit (DU) architecture. Because the CU-DU architecture is widely distributed by

means of numerous IoT, the offloading framework has a good adaptation in 5G networks [2]. Consequently, 5G urges the evolution of IoT, which makes IoT have more security, more reliability, lower latency, etc.

In general, the offloading framework is divided into four parts: Cloud, ECNs, DUs and MDs. Cloud, equipped with scalable computing resources, is powerful to execute the deep learning tasks by leveraging Virtual Machines (VMs) which are employed for hosting the deep learning model [16]. ECNs, co-located with CUs, which have the similar functions with Cloud, are edge computing nodes, where a certain number of VMs with deep learning models are installed. The ECNs receive a mass of data produced by the surrounding MDs, and they perform pre-processing and execute parts of deep learning tasks it can resolve [7]. Considering the limited resources that ECNs own, each ECN only performs a certain number of deep learning tasks. DUs work as transferring stations for data transmission to offload the data, generated by the surrounding MDs, to the ECNs they connect. MDs, such as mobile phones, laptops and flat computers, are able to be distributed to any places where they have access to 5G networks and offload deep learning tasks at any time they need.

To be specific, in this framework, Cloud accommodates any number of deep learning tasks it receives by extending computing resources. ECNs are connected with Cloud and transmitted deep learning tasks to Cloud. Each ECN is composed of multiple Mobile Edge Computing (MEC) servers and one CU. For MEC servers, their computing and storage resources are expressed in the form of VMs, which gives ECNs the ability of handling deep learning tasks. For CUs, their functions are receiving deep learning tasks, choosing servers that sustain the deep learning tasks and then transmitting these tasks to the chosen servers. For ECNs, any two ECNs are able to conduct data transmission through data channel and each ECN is connected with multiple DUs. Deep learning tasks are sent from DUs to ECNs through data channels. For each DU, its functions are receiving the deep learning tasks from MDs and transmitting these tasks to ECNs. Moreover, each MD is only connected with one DUs. For MDs, they transmit noisy and highly redundant deep learning tasks to ECNs for deep learning.

Assume that N represents the number of MDs and each MD can generate one deep learning task, denoted as $p_n = \{da_n, t_n\}$ $(n = \{1, 2, \ldots, N\})$, where da_n represents the data size of p_n, and T_n represents the running time of p_n. Accordingly, $P = \{p_1, p_2, \ldots, p_N\}$ represents the collection of all the total deep learning tasks that MDs generate. After producing the task set P, MDs transmit the tasks to the base stations for data transmission with DUs, denoted as $D = \{d_1, d_2, \ldots, d_M\}$, where M is the number of DUs, and $d_m(m = \{1, 2, \ldots, M\})$ represents a DU instance. Through DUs, deep learning tasks are transferred to ECNs, denoted as $C = \{c_1, c_2, \ldots, c_K\}$, where K is the number of ECNs and $c_k(k = \{1, 2, \ldots, K\})$ represents an ECN instance. Let e_k represent the number of VMs that ECN $c_k(k = \{1, 2, \ldots, K\})$ contains [17]. We suppose that each deep learning task takes up only one VM, and it chooses a VM as its offloading destination to host which needs to be obtained by the design of offloading strategies. Let z_n denote the offloading strategy of p_n. Correspondingly, the total strategy collection for the deep learning task set P is denoted as $Z = \{z_1, z_2, \ldots, z_N\}$. In this paper, we intend to optimize the offloading time according to the destination of deep learning task, thus, in the following sections, we will introduce the offloading time model at length.

B. OFFLOADING TIME MODEL

The transmission delay is the embodiment of the execution efficiency for the deep learning offloading strategy. The transmission delay is the standard to judge the quality of the offloading strategy. Before calculating the transmission delay, the destination of deep learning tasks and the route these tasks passed should be determined. Cloud and MEC servers are able to be served as the accommodation destination for task implementation. The transmission delay consists of four parts, including the transmission time between MD and DU (MT), the waiting time for the other deep learning tasks transmitted to the hosted ECN (WT), the transmission time from task migration between DU and ECN (DT) and the transmission time for task migration between ECN and the destination VM (CT).

Here are the formulas for the calculation of transmission delay for deep learning task offloading:

Result of a one-to-many relationship between an ECN and DUs, I_m^k is used to express this relationship which is a binary variable which is defined by

$$I_m^k = \begin{cases} 1, & \textit{if } d_m \textit{ connected to } c_k, \\ 0, & \textit{otherwise}. \end{cases} \tag{1}$$

Similarly, there is a one-to-many relationship between a DU and MDs. f_m^n is a binary variable for expressing the relationship between a DU and MDs, which is defined by

$$f_n^m = \begin{cases} 1, & \textit{if } p_n \textit{ is connected to } d_m, \\ 0, & \textit{otherwise}. \end{cases} \tag{2}$$

The calculation of $MT(zx_n)$ and $DT(z_n)$ has similar mathematical formula. $MT(z_n)$ is depended on the transmission rate among MDs, DUs and da_n. The mathematical calculation of $MT(Z_n)$ is specified as follows:

$$MT(z_n) = \frac{da_n}{\lambda_n^m} \tag{3}$$

In (3), λ_n^m is the transmission rate between p_n and d_m. Analogously, $DT(z_n)$ is calculated by

$$DT(z_n) = \frac{da_n}{\lambda_m^k} \tag{4}$$

where λ_m^k is the transmission rate between d_m to c_k.

The calculation of $WT(z_n)$ is decided by the time of the other tasks that are transmitted before p_n taking up DU.

Therefore, $WT(z_n)$ is calculated by

$$WT(z_n) = \sum_{i=1}^{k} (MT(q_i) + DT(q_i)) \tag{5}$$

where k is the number of these tasks transmitted before p_n and q_i is the offloading strategy set for these tasks.

The calculation of $CT(z_n)$ is divided into two cases in terms of the location of the destination VM of p_n. Since VMs are installed in the Cloud and MEC servers, the deep learning task p_n can employ any unoccupied VM from the Cloud or MEC servers. Therefore, $CT(z_n)$ is calculated by

$$CT(z_n) = \begin{cases} \frac{da_n}{\lambda_k^c}, & \text{if } p_n \text{ chooses Cloud} \\ \frac{da_n}{\lambda_{k1}^{k2}}, & \text{if } p_n \text{ chooses MEC servers} \end{cases} \tag{6}$$

where λ_k^c is the transmission rate between c_k and the cloud platform, and λ_{k1}^{k2} is the transmission rate between c_{k1} and c_{k2}.

In conclusion, the transmission delay of p_n, denoted as T_n, is the summation of $MT(z_n)$, $WT(z_n)$, $DT(z_n)$ and $CT(z_n)$, which is calculated by

$$T_n = \sum_{m=1}^{M} \sum_{k=1}^{K} f_n^m I_m^k (MT(z_n) + WT(z_n) + DT(z_n) + CT(z_n)) \tag{7}$$

and the total occupied time of all the deep learning tasks, denoted as T_{all}, is calculated by

$$T_{all} = \sum_{n=1}^{N} T_n \tag{8}$$

C. PROBLEM FORMULATION

The problem considered in this paper is to select the optimal offloading strategy for the deep learning tasks to minimize T_{all}. In other words, the ultimate goal is to identify a proper destination VM for hosting each deep learning task to optimize the overall transmission time in the offloading system. Therefore, the problem formulation is expressed as follows:

$$minimize(T_{all}) \tag{9}$$

III. HOM: HEURISTIC OFFLOADING METHOD FOR DEEP LEARNING EDGE SERVICES IN 5G NETWORKS

A. METHOD OVERVIEW

In tradition, the deep learning tasks are offloaded to the cloud platform which has massive computing resources that the users don't need to consider whether these tasks are able to be accommodated or not [18], [19]. This solution seems to be simple and useful, but after being carefully considered, the transmission delay that this solution spends is too much due to the slow transmission rate between Cloud and MDs and the large size of the deep learning tasks. In order to solve this problem, ECNs, which have high transmission rate to MDs, are coming up with to host some tasks with urgent time requirements. It's worth mentioned that ECNs have a certain amount of computing power, which causes the appearance of another computation offloading policy. The policy considers all of deep learning tasks are offloaded to the ECNs which they are connected to, which only pays attention to the high transmission rate between ECNs and MDs, overlooking the depletable computing resources of ECNs. In other words, a single ECN doesn't have enough resource to support all deep learning tasks for resource provisioning at the same time. These tasks need to wait for the tasks that occupied the VMs on the relevant ECN to release these VMs. This method consumes much transmission delay for this behavior above. In this session, a heuristic offloading algorithm, named HOM, is designed to solve the optimization of computation offloading transmission delay.

The aim of HOM is to find the ideal destination VM from ECNs and Cloud to shorten the offloading time. HOM avoids the waste of waiting time for the occupied resources of ECNs. The detailed procedures of this method are as follow:

- According to the distribution of ECNs, we confirm shortest offloading path between any pair of original and destination ECNs.
- Tasks are transmitted to their own local ECN, and then determine the best offloading destination from ECNs and Cloud.
- According to the offloading strategy above, we evaluate the offloading time of tasks.

B. SHORTEST OFFLOADING PATH CONFIRMATION

On account of the complicated route of ECNs, there are multiple paths between any two ECNs. For shortening transmission delay of deep learning tasks, acquiring the shortest path between any two ECNs is necessary. An example below shows the process of acquiring the shortest path.

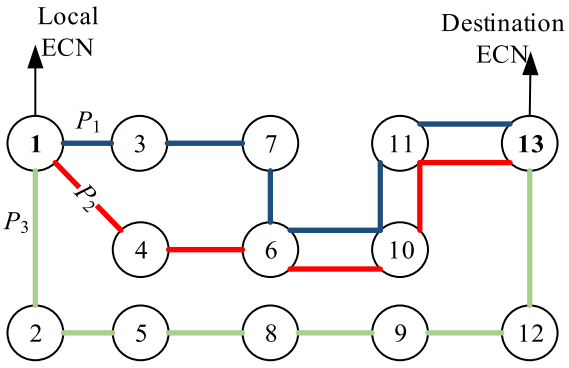


FIGURE 2. An example of a deep learning-enabled edge computing network.

Fig. 2 shows an example of a deep learning-enabled edge computing network. In Fig. 2, there are 13 nodes distributed geographically which represent the ECNs, denoted as c_1, $c_2, \ldots,$ and c_k. The links between different nodes for connecting ECNs are treated as the data channels. In this example,

c_1 is the local ECN and c_{13} is the destination ECN which the task is decided to be offloaded to.

There are three routes between c_1 and c_{13}, which is selected for comparison: $P_1 = \{c_1, c_3, c_7, c_6, c_{10}, c_{11}, c_{13}\}$, $P_2 = \{c_1, c_4, c_6, c_{10}, c_{11}, c_{13}\}$, $P_3 = \{c_1, c_2, c_5, c_8, c_9, c_{12}, c_{13}\}$. In Fig.2, the blue, red and green paths are denoted as P_1, P_2 and P_3 respectively. For finding the shortest path among P_1, P_2 and P_3, the transmission rates of these paths are considered as follows.

Assume that the transmission rates between connected ECNs, denoted as τ, are all equal. Let τ_{k2}^{k1} be the transmission rate of the shortest route between the ECNs $k1$ and $k2$. If the ECN $k2$ is connected with another ECN $k3$, there is an internal relationship among τ_{k3}^{k1}, τ_{k2}^{k1} and τ:

$$\frac{1}{\tau_{k3}^{k1}} = \frac{1}{\tau} + \frac{1}{\tau_{k2}^{k1}} \tag{10}$$

Deduced from (10), τ_{k3}^{k1} can be calculated by

$$\tau_{k3}^{k1} = \frac{\tau_{k2}^{k1} \cdot \tau}{\tau_{k2}^{k1} + \tau} \tag{11}$$

According to (10) and (11), it is obvious that the value of τ_{k3}^{k1} is smaller than that of τ_{k2}^{k1}. It is proved that the more data channels path contains, the lower the transmission rate is. Therefore, the shortest path means the least number of channels.

In the above example, the channels number of P_1, P_2, P_3 are 6, 5 and 6 respectively, thus the path P_2 is the shortest path among these three paths. This is the process of acquiring the shortest path between c_1 and c_{13}.

Through analysis, the paths between any two ECNs can be seen as a weighted undirected graph. The transmission rate can be understood by weight. The problem is transformed to find the shortest path on the weighted undirected graph. There are many methods solving this problem, like Dijkstra Algorithm and Floyd Algorithm. Consequently, we choose the Dijkstra Algorithm to solve this problem, where the input is a two-dimensional matrix that denotes the transmission rate between any two ECNs [20], [21]. We use a two-dimensional array to record the output of Dijkstra Algorithm, denoted as s [22]–[24].

C. OFFLOADING PATH IDENTIFICATION

The offloading strategies for a task are classified as 3 cases by the destination location the task locates in.

- *Case 1:* Offload this task to an unoccupied VM in local ECN. This case has the shortest transmission delay which is almost zero.
- *Case 2:* Offload this task to unoccupied VMs in ECNs except local ECN. The transmission delay of this case is calculated by (6). Because of the complicated route between ECNs, finding a route to make the average transmission rate λ_{k1}^{k2} be the shortest is a problem.
- *Case 3:* Offload this task to Cloud. da_n/λ_k^c is transmission delay of $CT(z_n)$ in this case.

Case 1 is the best choice obviously, thus offload deep learning task to local ECN when local ECN has unoccupied VMs. When there is no idle VMs in local ECN, it's time to choose the better solution between case 2 and case 3.

For case 2, choosing which ECN to offload deep learning tasks is the key to the question. We use s, the output of Dijkstra Algorithm, to solve this problem.

Here is an example for the offloading of deep learning tasks p_n. Assuming that the ECN c_1 in Fig. 2 is local ECN of the task p_n and the transmission rate τ between any adjacent ECNs is set to 540 (M/s). Therefore, the transmission rates between local ECN c_1 and all ECNs could be calculated by (11) and the results are $\{\infty, 540, 540, 540, 270, 270, 270, 180, 135, 180, 135, 108, 108\}$ (M/s), donated as τ_{c1}. The corresponding ECNs, denoted as C, are $\{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9, c_{10}, c_{11}, c_{12}, c_{13}\}$. In order to determine the ideal ECN, the ECN which has higher transmission rate to local ECN is given priority. Therefore, sort τ_{c1} from large to small [25], [26] and change C according to τ_{c1}. After do this, τ_{c1} and C are changed to $\{\infty, 540, 540, 540, 270, 270, 270, 180, 180, 135, 135, 108, 108\}$ (M/s) and $\{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_{10}, c_9, c_{11}, c_{12}, c_{13}\}$ respectively. Then, check whether MEC servers of these ECNs have idle VMs or not in terms of the sequence of C. Last, the first ECN which has idle VMs is chosen as the ideal ECN. Algorithm 1 *OPI* (*s, local*) describes the determination of the ideal ECN in detail.

In order to make choice effectively, rank s from the largest to the smallest. After that, search for ECNs and its task case of computing resource in sequence according to s. While we find an ECN meeting $e_k > r_k$, this ECN c_k is decided to be the destination of offloading. That is what *OPI* (*s, local*) does. Moreover, *OPI*(*s, local*) also contributes transmission rate of the shortest path between local ECN and ideal ECN. The process of *OPI*(*s, local*) is ranking s, changing the sequence of *no*(line 4 to line 11), traversing ECNs in terms of the sequence of *no* and determining the ideal ECN of the deep learning task (line 12 to line 16).

Through *OPI*(*s, local*), the serial number of the ideal ECN, denoted as*b_no*, and its transmission rate of shortest path, denoted as *b_s*, are acquired.

For case 3, the calculation of $CT(z_n)$ relies on the transmission rate between local ECN and Cloud according to (6). [27]

D. OFFLOADING DESTINATION CONFIRMATION

Algorithm 2 *ODC*(*b_s, b_no, local, C2W*) is considered to determine the offloading destination and calculate the offloading time of target task, where *C2W* is the transmission rate between Cloud and ECNs.

Considering the offloading destination confirmation, target task searches for local ECN at first. If local ECN has unemployed VMs, this task will choose local ECN as offloading destination. While local ECN doesn't have usable VMs, this task will search for usable ECN by Algorithm 1 *OPI*(*s, local*) and then compare the transmission rate of the usable ECN

Algorithm 1 Offloading Path Identification *OPI*(*s, local*)

Require: *s*, *local*
Ensure: *b_no*, *b_s*
1: **for** *i* in *K* **do**
2: $no(i) = i$;
3: **end for**
4: **for** *i* in $K - 1$ **do**
5: **for** $j = i + 1$ in *K* **do**
6: **if** $s(local, j) > s(local, i)$ **then**
7: exchange s(local, j) and $s(local, i)$
8: exchange $no(j)$ and $no(i)$
9: **end if**
10: **end for**
11: **end for**
12: **for** *i* in *K* **do**
13: **if** $no(i)$ has unoccupied VMs **then**
14: $b\_no = no(i)$, $b\_s = s(i)$
15: **end if**
16: **end for**
17: **return** *b_no*,*b_s*

Algorithm 2 Offloading Destination Confirmation *ODC*(*b_s*, *b_no, local, C2W*)

Require: *b_no*, *b_s*, *local*,*C2W*
Ensure: T_n
1: **for** *m* in *M* **do**
2: find d_m which is connected to p_n
3: $num = 0$
4: **for** *n* in *N* **do**
5: find MDs which are connected to d_m
6: get*NO*of MDs get*NO*of MDs
7: *get* da_n of MDs
8: *num*++
9: **end for**
10: **end for**
11: sort *NOs* according to da_n
12: **for** *i* in *num*
13: **if** $NO(i) \neq n$ **then**
14: Add time of $NO(i)$'s transmission to $WT(z_n)$
15: **end if**
16: **end for**
17: **if** *local* has unoccupied VMs **then**
18: $CT(z_n) = 0$, employ a VMs from *local*
19: **else**
20: $s = \max(b\_s, C2W(local))$
21: $CT(z_n) = da_n/s$
22: **end if**
23: $T_n \mathrel{+}= MT(z_n) + DT(z_n) + CT(z_n) + WT(z_n)$
24: **return** T_n

with Cloud's. The strategy which has bigger transmission rate is chosen as the destination strategy.

Considering the calculation of offloading time, the calculation of $WT(z_n)$ is a problem, because DU only transmits one deep learning task at the same time [28]. For calculating $WT(z_n)$, it is necessary to know when p_n is invoked by local ECN. In this paper, we schedule the sequence of deep learning tasks being invoked by data size. Because the offloading time of deep learning task which has bigger data size is more in the same case, task which has smaller data size is priority selection. An example below shows the process of calculating the offloading time of p_4.

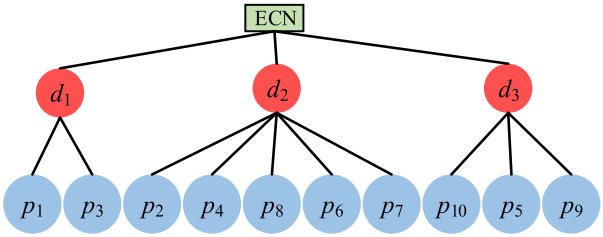


FIGURE 3. An example of route distribution of DUs and deep learning tasks under ECN c_1.

Fig. 3 shows an example of the distribution of the DUs and the deep learning tasks covered within an ECN, denoted as c_1. In Fig. 3, there are 1 ECN (i.e., c_1), 3 DUs (i.e., d_1, d_2, and d_3) and 10 MDs (i.e., $p_1 \sim p_{10}$). Let $s = \{20, 50, 60, 30, 10, 60, 20, 50, 70, 80\}$ represent the data size of these tasks. In Fig. 3, the deep learning tasks are divided into three subsets: $l_1 = \{p_1, p_3\}$, $l_2 = \{p_2, p_4, p_8, p_6, p_7\}$, $l_3 = \{p_{10}, p_5, p_9\}$. Tasks from l_1 are all transmitted to d_1. Analogously, tasks from l_2 and l_3 are transmitted to d_2 and d_3 respectively. d_1, d_2 and d_3 are all connected to c_1.

The waiting time for p_4 needs to be achieved by l_2 since all the tasks in l_2 including p_4 are covered by the same DU d_2. The storage size for the computing tasks in l_2, denoted as s_2, is set to $\{50, 30, 50, 60, 20\}$. Then, s_2 is sorted in the decreasing order of the storage size and l_2 is adjusted accordingly. As a result, s_2 and l_2 are changed to $\{60, 50, 50, 30, 20\}$ and $\{p_6, p_2, p_8, p_4, p_7\}$ respectively. The transmitting order for all the computing tasks that needs to be offloaded is determined by the task sequence inl_2. In this example, p_6, p_2 and p_8 are transmitted through d_2 before p_4. Therefore, $WT(z_4)$ is the accumulation of *MT* and *DT* for p_6, p_2 and p_8. Finally, the total time of p_4 is the accumulation of $MT(z_4)$,$WT(z_4)$, $DT(z_4)$ and $CT(z_4)$.

Algorithm 2 *ODC*(*b_s*, *b_no, local, C2W*) describes the calculation of T_n in detail, which aims at achieving a collection of deep learning tasks that are co-located with p_n in the same ECN (line 1 to line 10), sorting the collection in descending order of storage (line 11), calculating $WT(z_n)$ (line 12 to line 16), confirming the offloading destination and calculating $CT(z_n)$ (line 17 to line 22), calculating the transmission delay of target task (line 23).

E. OFFLOADING TIME EVALUATION

T_{all} is the cumulative transmission delay of all the deep learning tasks, which is the important index for evaluating the offloading strategy. Algorithm 3 $OTC(T_n)$ describes the

Algorithm 3 Offloading Time Calculation $OTC(T_n)$

Require: T_n
Ensure: T_{all}
1: *Dijkstra* Algorithm
2: **for** n in N **do**
3: **for** m in M **do**
4: find d_m which is connected to p_n through f_n^m
5: calculate $MT(z_n)$
6: find c_k which is connected to d_m through I_n^m
7: calculate $DT(z_n)$
8: $local = k$
9: **end for**
10: *Algorithm 1 OPI*(*s, local*)
11: T_n = *Algorithm 2 ODC*(*b_s, b_no, local, C2W*)
12: add T_n to T_{all}
13: **end for**
14: **return** T_{all}

calculation of T_{all}, where *Dijkstra Algorithm* is used to get the shortest path and the transmission rate between any two ECNs (line 1), Algorithm 1 is used to find the usable ECN with the highest transmission rate for task p_n (line 10), and Algorithm 2 is used to confirm the offloading destination and calculate the transmission delay of the task p_n (line 11). [29]

IV. SIMULATION EXPERIMENT

In order to evaluate the performance of HOM, a series of simulation experiments are conducted. Before performing experiments, it is necessary to set the simulation parameter, whose function is enriching the offloading framework to meet the requirements of simulation experiments. Moreover, two comparative methods, described in the following section, are searched to evaluate the performance of HOM. Above all, through changing the number of deep learning tasks, the influence of transmission delay is evaluated.

A. SIMULATION SETUP

An application scenario is taken into consideration that the deep learning tasks are faced with a difficulty of the destination confirmation between Cloud and ECNs. On this basis, assume that there are K ECNs, M DUs and N MDs. There are e_k VMs are accommodated in the k-th ECN. Besides, each ECN controls multiple DUs and each DU covers some MDs. Here, all simulation parameters are list in Table 2. According to the relationship between any two ECNs, a network topology of ECNs in our simulation is shown in Fig. 4.

In order to show the performance of HOM, two basic offloading methods different from HOM are employed to compare. The two comparative strategies are introduced as follows.

- Cloud-Offloading (CO): The deep learning tasks are all offloaded to Cloud. Cloud owns enough computing resources that the users have not to worry about the dynamic resource requirements of the deep learning tasks.

TABLE 2. Simulation parameter.

Description of Parameter	Value
The number of DUs M	50
The number of ECNs K	10
The transmission rate between a MD and a DU α	1200M/s
The transmission rate between an ECN and Cloud δ	220M/s
The transmission rate between ECNs γ	540M/s
The transmission rate between a DU and an ECN β	540M/s

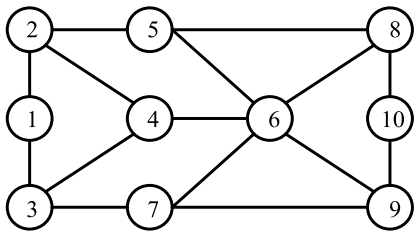


FIGURE 4. Network topology of ECNs in our simulation.

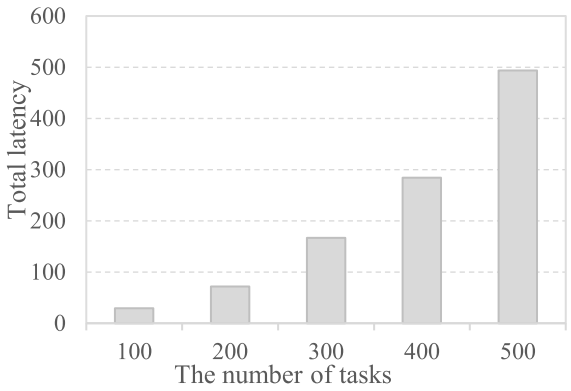


FIGURE 5. Comparison of HOM's total latency by different number of deep learning tasks.

- Local-Offloading (LO): The deep learning tasks are all offloaded to their 'local ECN' sequentially. If the 'local ECN' does not have enough unoccupied VMs to support the coming deep learning task, this task will wait in queue until the 'local ECN' have enough VMs for it. This method will not stop until all deep learning tasks are offloaded into MEC servers.

B. PERFORMANCE ANALYSIS OF HOM

The aim of HOM is to shorten the transmission delay of deep learning tasks and to take full advantages of computing resources for ECNs. We observe the growth of transmission delay through gradually increasing the number of tasks. Fig. 5 shows the comparison of total transmission delay for HOM in different number of deep learning tasks.

From Fig. 5, the total transmission delay is very short when the number of deep learning tasks is 100. With the increase of deep learning tasks, transmission delay of these tasks is becoming higher and higher. HOM works well when ECNs

have adequate computing resources for tasks. Note that the transmission delay in our experiments does not contain the running time of all the deep learning tasks.

C. COMPARISON ANALYSIS

HOM, CO and LO are performed and analyzed with the same experimental environment configuration. We evaluate the performance of each offloading method by comparing the transmission delay of deep learning tasks. Furthermore, the number of employed VMs in ECNs, the range of running time for deep learning tasks, the range of data volume for the deep learning tasks and the capacity of ECNs are taken into consideration for evaluating the offloading efficiency. The comparison results of these metrics are respectively shown in Fig. 6, Fig. 7, Fig. 8, Fig. 9 and Fig. 10.

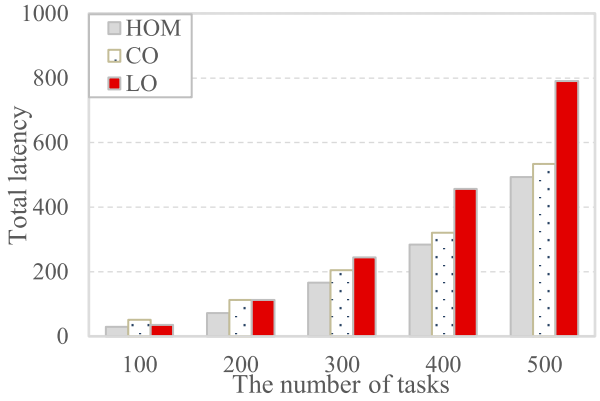


FIGURE 6. Comparison of the total latency of HOM, CO, and LO by different number of deep learning tasks.

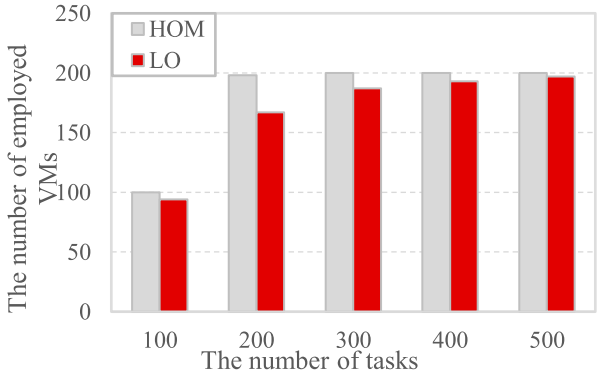


FIGURE 7. Comparison of the number of employed VMs of HOM and LO by different number of deep learning tasks.

1) Comparison of the total computation offloading transmission delay: Generally, the offloading transmission delay has huge changes when the number of deep learning tasks for different computing methods has changed. We compare the total transmission delay of HOM, CO and LO by changing the number of deep learning tasks. Fig.6 shows the result, where we discover that the total transmission delay of three methods all increases fast, but the transmission delay of HOM is shorter than the other two methods. Meanwhile,

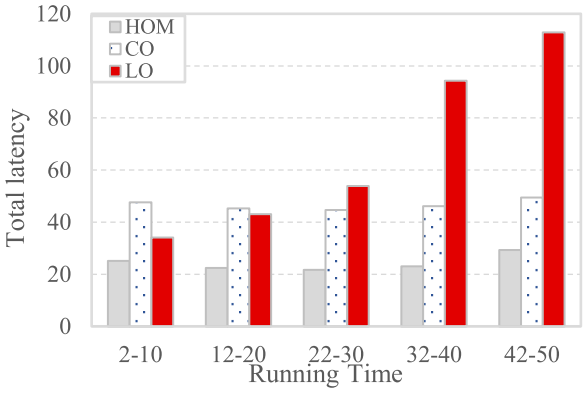


FIGURE 8. Comparison of the total latency of HOM, CO, and LO by different range of tasks' running time.

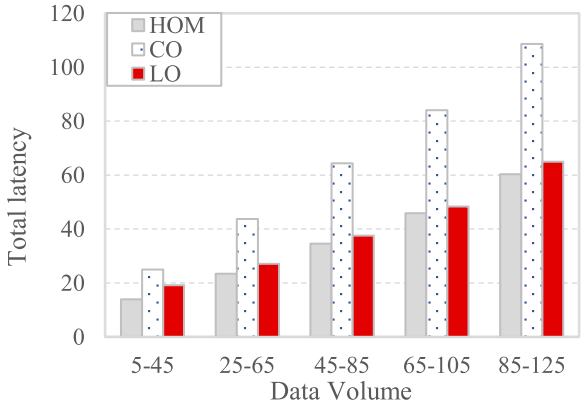


FIGURE 9. Comparison of the total latency of HOM, CO and LO by different range of tasks' data volume.

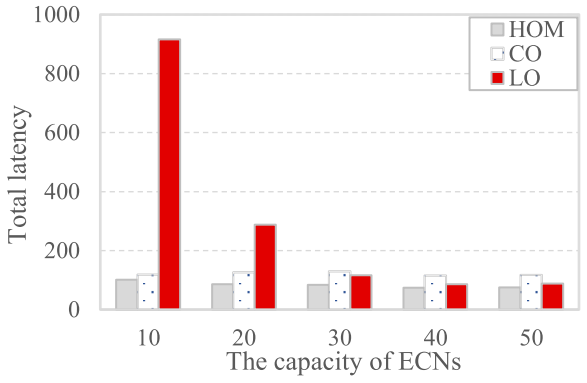


FIGURE 10. Comparison of the total latency of HOM, CO, and LO by different capacities of each ECN.

compared with the other two methods, the total transmission delay of LO is more affected by the number of deep learning tasks. With these observations, HOM achieves the shortest the offloading transmission delay among these three methods.

2) Comparison of the number of employed VMs in ECNs: In our experiments, each deep learning task only employs one VM and all these tasks are offloaded in terms of the computation offloading method. The number of the employed VMs in ECNs decides whether the resource of ECNs is taken full advantage or not. As exhibited in Fig. 7, the number of

VMs that HOM employs is equal or larger than LO employs. There is no VM of ECNs employed to offload deep learning tasks when the offloading method is CO. When these tasks are few, the 'local ECN' owns enough VMs to support the offloading of these tasks. In this situation, the number of employed VMs for HOM is equivalent to LO. When the deep learning tasks are not all supported by their 'local ECN', HOM finds other ECNs for holding tasks but LO makes the tasks in queue sequentially wait for the 'local ECN' to release the occupied VMs. Therefore, the performance of HOM is better than any other two methods for this metric.

3) Comparison of the range of running time: We compare the total transmission delay by different range of running time for the deep learning tasks. This experiment is on the premise that the number of tasks is less than the capacity of all ECNs. So, this experiment is conducted when the total number of the deep learning tasks is set to 100 and the capacity of each ECN is set to 20. In Fig. 8, the total transmission delay by HOM and CO is not affected by the running time. The total transmission delay of LO is closely related to the running time. The longer the range of running time for tasks is, the higher total transmission delay of LO is. From the aspect of running time, the performance of HOM is the best in these three offloading strategies.

4) Comparison of the range of data volume: In Fig. 9, the range of data volume for deep learning tasks is compared. In this experiment, the total number of tasks is set to 100 and the capacity of each ECNs is set to 20. Through Fig. 10, we know that the total transmission delay of HOM is lower than any other two methods. The total transmission delay of these three methods is all affected by the data volume but CO is more obvious. From the aspect of data volume, the performance of HOM is better.

5) Comparison of the capacity for each ECN: The capacity of each ECN is considered to evaluate the total transmission delay. In this experiment, the total number of deep learning tasks is set to 200. Meanwhile, the range of running time and data volume are stationary. In Fig. 10, the total transmission delay of HOM and CO are not changed for the different capacities of the ECNs. Oppositely, when the capacity is small, LO needs to spend more time waiting for the 'local ECN' to release the occupied VMs, as shown in Fig. 10. It is noteworthy that total transmission delay of HOM is always lower than CO generates by different capacities of each ECN. Therefore, from all the comparison analysis presented above, HOM exhibits the best performance for the offloading of the deep learning tasks.

V. RELATED WORK

Researches on deep learning have made great achievements recently. Deep learning has been used in nearly all fields to optimize algorithm and increase efficiency. In [7], the authors led deep learning into the edge computing environment and an algorithm for increasing the number of tasks as far as possible was proposed, which optimizes the network and protect the user privacy. As a matter of fact, there are lots of research achievements on deep learning and [30] has summarized some of them. In order to meet the requirements of gradually stricter users, deep learning still plays an important role in the evolution of 5G.

In 5G networks, IoT will be expended massively from 4G to meet the requirements of the future applications such as security, ultra-low latency, massive connection networks and so on. Just in time, 5G provides the technical advantages of low latency, high reliability, security and so on [31]. In [32], the infrastructure for deploying the base stations in 5G was devised and the solutions for high bandwidth transmitters were considered. In [33], Kitao *et al*. exploited an evaluation tool for 5G, which is used to evaluate the performance of 5G through propagation characteristics. 5G not only has advantages, but also needs improvement. In the city networks, massive connection networks are required by applications, which causes the production of many heterogeneous IoT networks [2]. These heterogeneous IoT networks bring many challenges for 5G. To adapt the changes in 5G, we have to consider how to choose ECNs and Cloud legitimately to offload tasks [34]. In [35], a distributed and offloading framework named DisCO was developed to offload the data-intensive or computationally-intensive part of the applications to the optimal mobile edge server. In addition, there are also other investigations that have been realized to offload the tasks to the ECNs. In [36], Ridhawi *et al*. envisioned a service-composition collaborative framework, which was real-time and context-aware, achieving load balancing between mobile and edge nodes. In [10], studying the condition where tasks are uploaded to a MEC server in a signal cell by multiple mobiles, an algorithm named SMSEF is proposed to realize the solving process.

On the other hand, some researches pay close attention to the consumption of energy and try their best to make consumption lower. In [4], an offloading system model and an innovative architecture called "MVR" were proposed to contribute to computation offloading. MVR works in the Edge Cloud platform by fine-grained offloading, conducive to improve the application performance and mitigate the resource burden. In [37], Xu *et al*. proposed a method of energy-aware computing offloading, named EACO, which is aimed to reduce the consumption of energy. In [38], Yang *et al*. focused more on a small-cell network scenario and an algorithm achieving global convergence was developed to achieve energy efficiency. In [39], a novel privacy- preserving and scalable service recommendation approach based on SimHash was proposed for a cross-cloud service recommendation scenario. In [40], Wang *et al*. proposed a green service composition approach for the problem of network resource consumption and energy of the composite services. In [41], an approximate approach balancing the workload between edge nodes was proposed for cost reduction and QoS improvement.

However, the consumption of time is also of great significance and there are few papers taking it into account. Fortunately, in this paper, a heuristic offloading method is

presented to shorten time latency for the deep learning edge services in 5G networks. We consider the latency for subchannels together with the resource situation of the edge nodes. At last, the proposed offloading method is verified to be suitable for the computation offloading of the deep learning tasks in 5G networks.

VI. CONCLUSION

Computation offloading is a key technique for the deep learning edge services in 5G networks. In order to shorten the transmission delay of deep learning tasks, a heuristic offloading method is devised in this paper. Specifically, we build an offloading framework within CU-DU architecture and analyze the offloading time on this basis. Then, the offloading process of deep learning tasks is stated in detail. Last, we demonstrate the usability of the proposed method HOM through experimental evaluations.

REFERENCES

[1] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for Internet of Things: A survey," *IEEE Internet Things J.*, vol. 3, no. 1, pp. 70–95, Feb. 2016.

[2] S. Li, L. Da Xu, and S. Zhao, "5G Internet of Things: A survey," *J. Ind. Inf. Integr.*, vol. 10, pp. 1–9, Jun. 2018.

[3] R. M. Rao, M. Fontaine, and R. Veisllari, "A reconfigurable architecture for packet based 5G transport networks," in *Proc. 5GWF*, Silicon Valley, CA, USA, Jul. 2018, pp. 474–477.

[4] X. Wei, S. Wang, A. Zhou, J. Xu, S. Su, S. Kumar, and F. Yang, "MVR: An architecture for computation offloading in mobile edge computing," in *Proc. EDGE*, Honolulu, HI, USA, Jun. 2017, pp. 232–235.

[5] X. Wang, L. T. Yang, H. Liu, and M. J. Deen, "A big data-as-a-service framework: State-of-the-art and perspectives," *IEEE Trans. Big Data*, vol. 4, no. 3, pp. 325–340, Sep. 2018.

[6] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing a key technology towards 5G," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, Sep. 2015.

[7] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan./Feb. 2018.

[8] M. Song, K. Zhong, J. Zhang, Y. Hu, D. Liu, W. Zhang, J. Wang, and T. Li, "*In-situ* AI: Towards autonomous and incremental deep learning for IoT systems," in *Proc. HPCA*, Vienna, Austria, Feb. 2018, pp. 92–103.

[9] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[10] F. Wei, S. Chen, and W. Zou, "A greedy algorithm for task offloading in mobile edge computing system," *China Commun.*, vol. 15, no. 11, pp. 149–157, Nov. 2018.

[11] X. Xu, S. Fu, Y. Yuan, Y. Luo, L. Qi, W. Lin, and W. Dou, "Multi-objective computation offloading for workflow management in cloudlet-based mobile cloud using NSGA-II," *Comput. Intell.*, Dec. 2018. doi: 10.1111/coin.12197.

[12] X. Xu, M. Z. A. Bhuiyan, L. Qi, X. Zhang, and W. Dou, "Load aware management of cloudlets for a wireless area metropolitan network," in *Proc. SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI*, Guangzhou, China, Oct. 2108, pp. 1283–1288.

[13] X. Wang, L. T. Wang, X. Xie, J. Jin, and M. J. Deen, "A cloud-edge computing framework for cyber-physical-social services," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 80–85, Nov. 2017. doi: 10.1109/MCOM.2017.1700360.

[14] J. Zhang, L. Qi, Y. Yuan, X. Xu, and W. Dou, "A workflow scheduling method for cloudlet management in mobile cloud," in *Proc. SmartWorld/SCALCOM/UIC/ATC/CBD-Com/IOP/SCI*, Guangzhou, China, Oct. 2108, pp. 932–937.

[15] X. Xu, Q. Liu, L. Qi, Y. Yuan, W. Dou, and A. X. Liu, "A heuristic virtual machine scheduling method for load balancing in fog-cloud computing," in *Proc. BigDataSecurity/HPSC/IDS*, Omaha, NE, USA, May 2018, pp. 83–88.

[16] Y. Huang, X. Ma, X. Fan, J. Liu, and W. Gong, "When deep learning meets edge computing," in *Proc. IEEE 25th Int. Conf. Netw. Protocols (ICNP)*, Toronto, ON, Canada, Oct. 2017, pp. 1–2.

[17] X. Xie, T. Yuan, X. Zhou, and X. Cheng, "Research on trust model in container-based cloud service," *Comput., Mater. Continua*, vol. 56, no. 2, pp. 273–283, Sep. 2018.

[18] J. Zhang, N. Xie, X. Zhang, K. Yue, W. Li, and D. Kumar, "Machine learning based resource allocation of cloud computing in auction," *Comput., Mater. Continua*, vol. 56, no. 1, pp. 123–135, 2018.

[19] S. Wang, L. Zhang, Y. Zhang, J. Sun, C. Pang, G. Tian, and N. Cao, "Natural language semantic construction based on cloud database," *Comput., Mater. Continua*, vol. 57, no. 3, pp. 603–619, 2018.

[20] G. Wang, H.-T. Che, and H.-B. Chen, "Feasibility-solvbility theorems for generalized vector equilibrium problem in reflexive banach spaces," *Fixed Point Theory Appl.*, vol. 2012, pp. 1–13, 2012.

[21] G. Wang and H.-T. Che, "Generalized strict feasibility and solvability for generalized vector equilibrium problem with set-valued map in reflexive Banach spaces," *J. Inequalities Appl.*, vol. 2012, Mar. 2012, Art. no. 66.

[22] B. Liu, B. Qu, and N. Zheng, "A successive projection algorithm for solving the multiple-sets split feasibility problem," *Numer. Funct. Anal. Optim.*, vol. 35, no. 11, pp. 1459–1466, Jul. 2014.

[23] H. Chen, Y. Wang, and G. Wang, "Strong convergence of extragradient method for generalized variational inequalities in Hilbert space," *J. Inequalities Appl.*, vol. 2014, Jun. 2014, Art. no. 223.

[24] B. Qu, B. Liu, and N. Zhang, "On the computation of the step-size for the CQ-like algorithms for the split feasibility problem," *Appl. Math. Comput.*, vol. 262, no. 1, pp. 218–223, Jul. 2015.

[25] Z. Shi and S. Wang, "Modified nonmonotone Armijo line search for descent method," *Numer. Algorithms*, vol. 57, no. 1, pp. 1–25, May 2011.

[26] M. Sun and Q. Bai, "A new descent memory gradient method and its global convergence," *J. Syst. Sci. Complex.*, vol. 24, pp. 784–794, Aug. 2011.

[27] Y. Wang, X. Sun, and F. Meng, "On the conditional and partial trade credit policy with capital constraints: A Stackelberg model," *Appl. Math. Model.*, vol. 40, no. 1, pp. 1–18, Jan. 2016.

[28] S. Li and Y. Zhang, "On-line scheduling on parallel machines to minimize the makespan," *J. Syst. Sci. Complex.*, vol. 29, no. 2, pp. 472–477, Apr. 2016.

[29] S. Lian and Y. Duan, "Smoothing of the lower-order exact penalty function for inequality constrained optimization," *J. Inequalities Appl.*, vol. 2016, Dec. 2016, Art. no. 185.

[30] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, to be published.

[31] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *J. Parallel Distrib. Comput.*, vol. 127, pp. 160–168, May 2019.

[32] N. Wolff, S. Chevtchenko, A. Wentzel, O. Bengtsson, and W. Heinrich, "Switch-type modulators and PAs for efficient transmitters in the 5G wireless infrastructure," in *IEEE MTT-S Int. Microw. Symp. Dig. Int. Microw. Workshop Ser. 5G Hardw. Syst. Technol. (IMWS-5G)*, Dublin, Ireland, Aug. 2018, pp. 1–3.

[33] K. Kitao, A. Benjebbour, T. Imai, Y. Kishiyama, M. Inomata, and Y. Okumura, "5G system evaluation tool," in *Proc. IEEE Int. Workshop Electromagn., Appl. Student Innov. Competition (iWEM)*, Nagoya, Japan, Aug. 2018, pp. 1–2.

[34] Y. Yu, "Mobile edge computing towards 5G: Vision, recent progress, and open challenges," *China Commun.*, vol. 13, no. 2, pp. 89–99, 2016.

[35] K. Ko, Y. Son, S. Kim, and Y. Lee, "DisCO: A distributed and concurrent offloading framework for mobile edge cloud computing," in *Proc. ICUFN*, Milan, Italy, Jul. 2017, pp. 763–766.

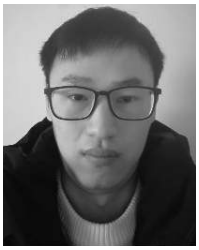
[36] I. A. Ridhawi, M. Aloqaily, Y. Kotb, Y. A. Ridhawi, and Y. Jararweh, "A collaborative mobile edge computing and user solution for service composition in 5G systems," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 11, p. e3446, Jun. 2018.

[37] X. Xu, W. Dou, X. Zhang, and J. Chen, "EnReal: An energy-aware resource allocation method for scientific workflow executions in cloud environment," *IEEE Trans. Cloud Comput.*, vol. 4, no. 2, pp. 166–179, Apr./Jun. 2016.

[38] L. Yang, H. Zhang, M. Li, J. Guo, and H. Ji, "Mobile edge computing empowered energy efficient task offloading in 5G," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6398–6409, Jul. 2018.

[39] Y. Xu, L. Qi, W. Dou, and J. Yu, "Privacy-preserving and scalable service recommendation based on SimHash in a distributed cloud environment," *Complexity*, vol. 2017, Dec. 2017, Art. no. 3437854.
[40] S. Wang, A. Zhou, R. Bao, W. Chou, and S. S. Yau, "Towards green service composition approach in the cloud," *IEEE Trans. Serv. Comput.*, to be published.
[41] Y. Guo, S. Wang, A. Zhou, J. Xu, J. Yuan, and C.-H. Hus, "User allocation-aware edge cloud placement in mobile edge computing," *Softw., Pract. Exper.*, Feb. 2019. doi: 10.1002/spe.2685.

XIAOLONG XU received the Ph.D. degree from Nanjing University, China, in 2016. He was a Research Scholar with Michigan State University, USA, from 2017 to 2018. He is currently a Lecturer with the School of Computer and Software, Nanjing University of Information Science and Technology. He has published over 40 peer review papers in international journals and conferences, including TCC, TBD, JNCA, SPE, FGCS, CI, CCPE, ICSOC, and ICWS. His research interests include fog computing, edge computing, the IoT, cloud computing, and big data. He received the Best Paper Award from the IEEE CBD 2016.



DAOMING LI is currently pursuing the B.S. degree in computer science and technology with the School of Computer and Software, Nanjing University of Information Science and Technology. His research interests include 5G and mobile edge computing.

ZHONGHUI DAI is currently pursuing the B.S. degree in computer science and technology with the School of Computer and Software, Nanjing University of Information Science and Technology. Her research interests include 5G and mobile edge computing.

SHANCANG LI received the B.Sc. and M.Sc. degrees in mechanics engineering and the Ph.D. degree in computer science from Xi'an Jiaotong University, Xi'an, China, in 2001, 2004, and 2008, respectively. He is currently a Senior Lecturer with the Department of Computer Science and Creative Technologies, University of the West of England, Bristol, U.K. His current research interests include digital forensics for emerging technologies, cyber security, the IoT security, data privacy-preserving, the Internet of Things, Blockchain technology, and the lightweight cryptography in resource constrained devices.

XUENING CHEN received the bachelor's degree from Qufu Normal University, in 2006, and the master's degree from East China Normal University, in 2009. She is currently a Lecturer with the Student Affairs Office, Qufu Normal University, China. Her research interests include services computing, psychology, and intelligent systems.