Received May 13, 2020, accepted June 4, 2020, date of publication June 16, 2020, date of current version June 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3002813

A Collaborative Service Deployment and Application Assignment Method for Regional Edge Computing Enabled IoT

YAN CHEN[1], **YANJING SUN**[1,2], **(Member, IEEE), TIANXIN FENG**[1], **AND SONG LI**[1], **(Member, IEEE)**
[1]School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 22116, China
[2]School of Communication and Information Engineering, Xi'an University of Science and Technology, Xi'an 710054, China

Corresponding author: Yanjing Sun (yjsun@cumt.edu.cn)

This work was supported in part by the Future Scientists Program of China University of Mining and Technology under Grant 2020WLKXJ026, and in part by the Postgraduate Research and Practice Innovation Program of Jiangsu Province under Grant KYCX20_1991.

ABSTRACT Edge computing brings powerful computing ability to the proximity of IoT devices to guarantee latency constraints, making it one essential technology for supporting intelligent applications in future Internet of Things (IoT). Collaboration among edge computing servers (ECSs) with limited resources is an efficient solution to enhance the capability of edge network, and placement of ECSs and service functions (SFs) impose significant influences on system performance. This paper explores the collaboration among ECSs by considering the simultaneous and heterogeneous consumption of different computing resources. The service deployment and application assignment in regional edge computing enabled IoT (EdgeIoT) are investigated. A collaborative service deployment and application assignment (ColSDA) algorithm is proposed to render the final edge service deployment strategy, including the placement of ECSs and SFs as well as the assignment of applications to ECSs. In ColSDA, the minimum number of ECSs to be placed is obtained by the proposed minimum resource ration increase (MinRI) algorithm. Computing loads are then balanced by the load-balancing reassignment (LBRA) algorithm. After placing ECSs, a search and swap (SeSw) algorithm is proposed to further increase the number of tasks processed by locally deployed ECSs. Simulation results demonstrate that the number of required ECSs under the premise of guaranteeing the quality of service (QoS) can be significantly reduced by establishing collaboration among ECSs. Besides, the proposed ColSDA algorithm can provide the service deployment and application assignment strategy for a given region EdgeIoT as expected.

INDEX TERMS Edge computing, service deployment, application assignment, the Internet of Things.

I. INTRODUCTION

Edge computing moves the capability of processing sophisticated tasks generated by emerging intelligent applications to the proximity of applications, making it promising architecture for future Internet of Things (IoT) to remedy the deficiency of cloud computing about latency that is more difficult to address in next-generation mobile networks [1], [2]. So, edge computing has attracted sights from both academic and industrial areas, and result in tons of works that have contributed to the development of edge computing [3]–[5].

The associate editor coordinating the review of this manuscript and approving it for publication was Adnan Kavak.

In early research related to task offloading strategies, each radio access network (RAN) was pre-installed with an edge computing server (ECS) with computing ability [6]–[8]. An ECS is set to provide computing services only for the IoT devices in the RAN where it is located, even in studies considering a multi-server system [9]. A cloud server is usually deployed to assist the processing of overloaded-tasks that cannot be accomplished by capacity-limited ECSs. However, the collaboration provided by a remote cloud server may be insufficient to satisfy the latency and reliable requirements of intelligent applications due to unpredictable network latency and jitter, and expenditure would be significantly increased. Collaboration among a set of adjacent ECSs is a feasible and economical solution that can both enhance the capability of edge network and ensure latency [10]–[13]. Especially in such typical regional networks as industrial Internet of Things (IIoT) and campus network.

In real IoT system with multiple RANs, the number of IoT devices and kinds of applications are heterogeneous in different RANs, which results in geographical unevenly distribution of computing loads [14]–[17]. Some computation tasks in load-heavy RANs may hardly be satisfied due to the limited capacity of ECSs, while ECSs deployed in RANs with light loads may have abundant unoccupied resources. More applications will be satisfied in edge network if ECSs with unoccupied resources can release part overloaded tasks from other RANs. Then, fewer ECSs are required if computing tasks from RANs with extremely light loads are released by other ECSs with idle resources, which can reduce the cost of network deployment. Besides, service function (SF) with dedicated code and database is indispensable for the task processing of intelligent applications [18], [19]. The running of an installed SF also require considerable computing resources that can be named as base resources [20]. An installed SF can serve multiple same kind applications from different IoT nodes by allocated dedicated resources and processing threads. So, the number of deployed SF can be decreased by distributed placing SF and allow IoT nodes to offload their task to SFs installed on the ECSs deployed in other RAN, which can reduce the consumption of base resources. Then, more resources can be spared to task processing, which can lead to considerable improvement in the system performance and resource-utilization for computation. In addition, during the processing of a computing task, multiple types of computing resources (i.e., CPU and memory) are consumed simultaneously. Apart from the difference in resource consumption between task processing of different kinds of applications, the task processing of a certain application may also expose obvious consumption-preference for a certain type of resource, which may lead to low resource utilization and system performance. For example, if tasks' processing of an application are rather sophisticated but little temp data generated, a large amount of CPU is occupied while a little memory is required [21]. Due to limited resources of the ECS, CPU will quickly be depleted when existing too many similar applications in the RAN, while unoccupied memory is still ample. Therefore, the resource efficiency and system performance can be improved by establishing collaboration among ECSs and assigning application tasks according to their resources-consumption properties, which also provide the potential of reducing the cost of network deployment.

Although collaboration among ECSs indicates potential advantages, network deployment cost and system performance are significantly influenced by edge service deployment policy which includes the placement of ECSs and SFs, as well as the assignment of application tasks to ECSs [22]. Similar to cloudlet placement in previous mobile cloud computing [23], [24], current ECS placement works consider deploying ECS for large area network like wireless metropolitan area networks (WMAN), in which, computing load composed by all applications in a RAN is treated as an entirety and can only be mapped to be processed by one server [25]–[28]. Besides, the capability of a server is assumed sufficient to handle the computation requirements of multiple RANs. These settings are reasonable in WMAN-like networks, while intelligent applications in future IIoT-like regional systems are rather sophisticated and densely deployed but may be prior known and working continuously. The resources of an ECS may not be always sufficient to handle the total computing load from one RAN. The SF placement (SFP) refers to placing SFs on pre-deployed ECSs and scheduling of the transmission or assignment of tasks from applications to corresponding SFs [19], [20], which can provide flexible edge service management to obtain optimal system performance like service latency, the number of applications processed in edge network, and data volume further offloaded to cloud server [29]–[32]. However, many of them consider every RAN is equipped with an ECS, which may not be economical. Besides, the simultaneous consumption of multiple kinds of computing resources has not been investigated, which may restrict the feasibility and efficiency of their mechanisms.

Therefore, given the above motivation, this paper investigates the edge service deployment in regional IoT like IIoT where facilities running sophisticated applications are pre-deployed and working continuously. A collaborative service deployment and application assignment (ColSDA) algorithm is proposed to find a service deployment and application assignment policy for a given regional IoT. Required ECSs for guaranteeing the quality of service (QoS) of all applications are obtained and placed by exploring the collaboration among ECSs with considering the SFP and resources-consumption properties of applications. The contributions of our work are concluded as follows:

- We investigate the problem of edge service deployment for required ECS minimization in regional edge computing enabled IoT (EdgeIoT). Particularly, the collaboration among ECSs is explored by considering the properties of task processing and SFP on the simultaneous consumption of different computing resources.
- A collaborative service deployment and application assignment (ColSDA) algorithm is proposed. A minimum resource ratio increase (MinRI) greedy algorithm is designed for the required ECS minimization problem. Local search based algorithms are proposed to further increase the tasks processed by locally deployed ECSs.
- Simulations are conducted and results demonstrate that collaboration among ECSs can significantly reduce the number of required ECS and the ColSDA method can provide service deployment tactics as expected.

The rest of this paper is organized as follows. Related works are discussed in section II. Section III introduces the collaboration in regional IoT. Section IV depicts the system model and service deployment problem. The proposed ColSDA algorithm is illustrated in section V. We evaluate the proposed ColSDA method in section VI. The paper is concluded in section VII.

II. RELATED WORKS

The deployment of ECS provides physical platform and resources for task processing, which plays a significant role in service capacity of EdgeIoT and guaranteeing QoS of applications. In recent years, few works have contributed to ECS placement. In [33], the problem of minimizing total energy consumption is studied and a particle swarm optimization based energy-aware ECS placement algorithm is proposed to find locations for ECSs and map of the workload from base stations (BS) to ECSs. In [25], ECS placement and BS to ECS association is studied to promise a certain end-to-end service latency with the minimum ECSs. Yang *et.al* [34] studied how to place ECSs and allocate each requested task to ECSs and the public cloud with the minimum total energy consumption without violating each task's latency requirement. Wang *et al.* [28] investigated the ECS placement problem in large-scale edge computing systems to achieve workload balance between ECSs and access delay minimization. In [26], a greedy low-cost ECS placement algorithm is proposed for WMAN to minimize the number of required ECS while ensuring access delay. A server placement and application configuration in WMAN is studied in [35] to minimize the system cost, however, the capacity of ECSs is simplified as the number of applications that can be processed, and the differences in QoS and resource requirement of applications are not considered. Cao *et al.* [27] studied the deployment of heterogeneous ECSs to optimize the expected response time of both the whole and individual BSs.

Most of the above works consider WMANs, in which, the capacity of an ECS is assumed sufficient to process computing loads from multiple RANs. Besides, RAN is considered as the minimum generator of computing load. Computing load from one RAN can only be assigned to one ECS as a whole. However, regional IoTs like IIoT is one of the main application scenarios of edge computing in the future. In such systems, computing tasks are rather sophisticated. And applications in a RAN are heterogeneous in types, QoS, and resources' requirements. Besides, due to economical or environmental limitations, the computing resources of an ECS are usually limited and may be insufficient to handle all applications in a RAN. Moreover, SFs are necessary for the task processing of future intelligent applications, and SFP plays an impose significant influence on QoS of applications and system performance.

Recent works have investigated the SFP and application task assignment in edge computing systems to improve system performances. Fan and Ansari [10] studied the workload allocation problem to ensure the service latency of application tasks and balance computation loads among ECSs. Guo *et al.* [36] proposed an energy-efficient workload allocation in an IoT-edge-cloud computing system for minimizing the total energy consumption while guaranteeing latency constraints of users. Chen *et al.* [29] investigated SFP in MEC-enabled dense small-cell networks and proposed a collaborative service placement method to optimize the utility of MEC operators. Every RAN in this study is assumed to be equipped with can be associated with each reachable BS. Poularakis *et al.* [30] studied multi-cell edge computing networks where every RAN is equipped with an ECS, and a joint service placement and request routing policy is designed to maximize the number of tasks served by ECSs. However, the base resources of SFs are not considered and required resources of applications are randomly generated rather than estimated based on their QoS and network state. In [32], SFP is investigated to maximize the total reward in a heterogeneous MEC system with a fixed number of ECS. However, base resources and latency caused by network transmission are not considered. Yu *et al.* [20] investigated the collaborative service placement in edge computing systems with the objective to minimize the traffic load caused by computing task forwarding. Each BS is installed with an ECS and only one computing resource is considered for task processing. Besides, the increase in the required computing resource of task processing that may be introduced by network latency is neglected. In [19], a machine learning-based SFP is proposed based on the prediction of traffic demand to satisfy the user with end-to-end latency and data rate requirements.

Different from previous works, this paper aims at providing edge computing service for a given region IoT by placing both ECSs and SFs. Besides, the simultaneous consumption of multiple kinds of computing resources is considered for establishing collaboration among ECSs to improve resource-efficiency and reduce the number of ECS to be placed.

III. COLLABORATION IN REGIONAL IoT

Considering a regional IoT consisting of multiple RANs as shown in figure 1, in which the number of IoT nodes and type of applications in each RAN are different from each other. So, the computing loads in RANs are heterogeneous. Assuming that every RAN is deployed with an ECS installed with SFs to provide edge computing service and covered IoT nodes can offload their incapable tasks to the ECS [37]. Then ECSs deployed in RANs with intensive loads undertake heavier computing burden than that deployed in RANs with light loads. Particularly, the computing load generated within one RANs may exceed the capacity of one ECS, and the QoS of some applications may not be satisfied. In contrast, ECSs deployed in RANs with a light load may have considerable idle resources left.

Besides, different computing resources are consumed simultaneously during task processing, and the amount of resources consumed by different applications are diverse. Especially, the task processing of some applications may have an obvious preference for a certain kind of resource. For example, application tasks whose computations are relatively simple, but require database support.

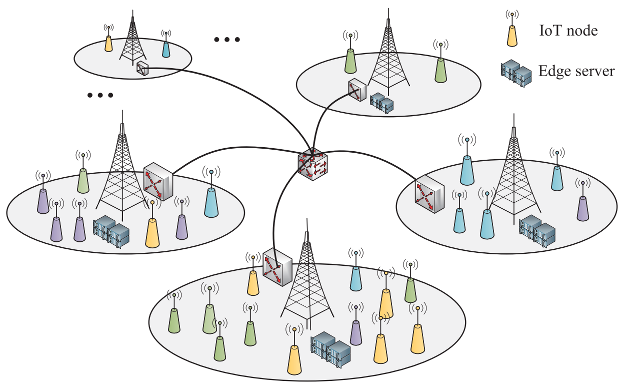


FIGURE 1. Regional edge computing enable IoT.

Meanwhile, tremendous temp data are generated during the computation. A small amount of CPU and huge memory are consumed. These kinds of applications can be named as memory-hungry applications in this paper. On the other hand, CPU-hungry application requires massive CPU as compared to memory. The CPU of an ECS may quickly be exhausted, while a proportion of memory is still unoccupied if assigning too many CPU-hungry applications to the ECS. Then the ECS is unable to provide computation service for more applications, although memory is still abundant. Such conditions may often occur in IIoT, since the same kind of facilities are usually placed in a concentrated area.

Moreover, SFs are software platforms consisting of dedicated codes and data to accomplish the task processing of corresponding intelligent IoT applications. Applications for industrial manufacturing are usually working continuously, so ECSs are always able to receive tasks offloaded from IoT nodes. Therefore, SFs should be pre-installed and launching, and private resources should be pre-allocated to guarantee the QoS of task processing. Otherwise, the SF needs to perform time-consumed data reloading and resources requesting operations each task arrives. An ECS is usually installed with different SFs to provide service for relevant applications simultaneously, and installed SFs are usually designed with the ability to handle multiple concurrent task requests. Considerable amounts of base resources are required for the working of a SF. For a certain application, the private data are stored in allocated dedicated memory space, and processing operations are accomplished by the allocated CPU. Then resources consumed by SFs can be reduced if tasks are assigned to SFs that have already been installed on other ECSs without instancing a new SF.

Inspired by the above backgrounds, we illustrate the feasibility and benefits of collaboration among ECSs by considering the properties of applications, and introduce three conceivable collaboration methods here, as shown in figure 2. Firstly, the ECS with idle resources can collaborate to release the computing load of the overloaded ECSs. We define such collaboration as the load-release collaboration, as shown in figure 2(a). Secondly, as exhibited

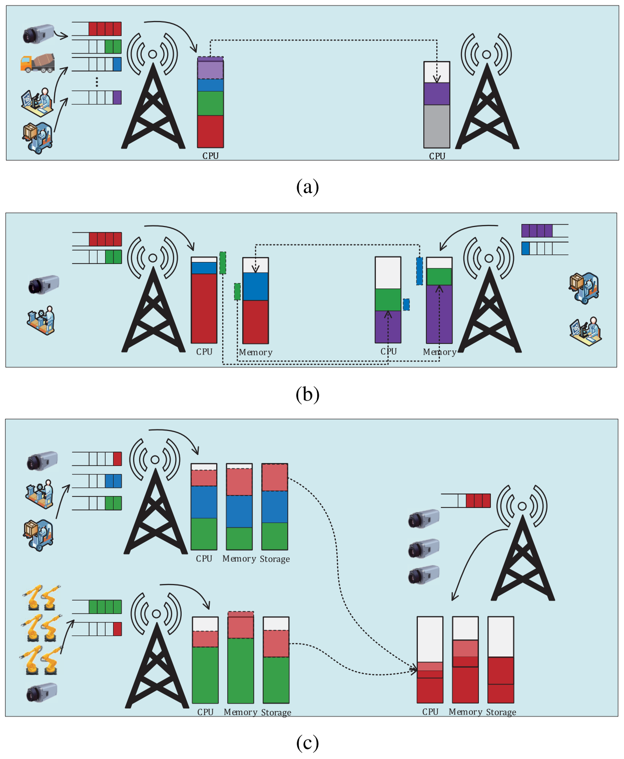


FIGURE 2. Collaboration among ECSs. (a) load releasing collaboration. (b) resources-complementary collaboration. (c) integration processing collaboration.

in figure 2(b), collaboration can also be established among ECSs by taking the resources-consumption properties of applications into account, which is named in this paper as the resources-complementary collaboration. As shown in figure 2(b), the CPU required by the application indicated by green is more than the residual CPU. Meanwhile, if the application represented by blue is processed at the right-side ECS, the memory will be exhausted while left a lot of free memory. However, both of these two applications can be satisfied if exchanging their processing location. Moreover, the resources of each edge server are utilized more balance, and both of these two ECSs may have idle resources for processing other tasks. Figure 2(c) displays the integration processing of congeneric applications, in which the same kind of applications can be processed by one deployed SF to reduce the resource occupation of instancing a new SF.

IV. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we first introduce the EdgeIoT system model and the computing service model in our work. Then, we formulate the service deployment and application assignment problem investigated in this paper.

A. REGIONAL EDGE COMPUTING IoT

In this paper, we investigate a regional EdgeIoT network, as revealed in figure 1, in which, IoT nodes $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ operating intelligent applications are distributed in the network to provide various services. A set of access points (APs) $\mathcal{P} = \{p_1, p_2, \dots, p_k\}$ are deployed to provide communication support for IoT nodes in their coverage, and IoT nodes are associated with their corresponding APs. Besides, there are j kinds of applications $\mathcal{A} = \{a_1, a_2, \dots, a_j\}$ are installed on the IoT nodes. In this paper, we assume that only one application can be installed on an IoT node. An IoT node with multiple applications installed in the real world can be represented by multiple nodes installing a single application. Therefore, each IoT node in figure 1 also represents an application. Then, considering the geographical distribution of computing loads and collaboration among ECSs, a set of ECSs $\mathcal{G} = \{e_1, e_2, \dots, e_i\}$ can be deployed to some of RANs to provide computing services. Besides, the data transmission is managed by a soft-defined network (SDN) switch [38], [39] since the data are usually required to be gathered to the data center in regional networks such as campus networks, networks for industrial plants.

Due to limited computing capacity and resources, IoT nodes will offload their incapable tasks to ECSs for efficient processing. SFs are installed on ECSs, and only one SF for each type of application is installed on each ECS. The edge computing service for an application can be represented by a tuple (u, p, a, e), which indicates the IoT node u with application a installed is associated with AP p. And the computing tasks are offloaded and forwarded to ECS e for execution. We define a binary indicator $x_{u,p} = 1$ to represent u is connect to p, $x_u^a = 1$ means the type of application installed on u is a, and $y_{u,e} = 1$ indicates task of application installed on u is assigned to be processed by ECS e. Besides, there are two kinds of relationships between AP p and ECS e, which is indicated by a binary indicator $\psi_{p,e}$. $\psi_{p,e} = 1$ means that ECS e is placed and associated with AP p. $\psi_{p,e} = 0$ means ECS e is not placed to the RAN covered by AP p.

Applications generate and offload computing tasks from IoT nodes to associated AP continuously. Then tasks are forwarded to assigned ECSs and be processed by corresponding SFs. Following a computation, the result is generated and replied to the IoT node along the reverse path. All related symbols are listed in table 1.

B. EDGE COMPUTING SERVICE

To guarantee QoS of continuously working applications, the task stream for each application is handled by a separate thread with dedicated resources. To simplify our work, we assumed that the offloaded tasks from IoT u arrival into the system follows a Poisson process with an average rate of λ_u. The data size of computing tasks and results for application a is uniformly distributed with means l_a^t and l_a^r. The required computing intensity for tasks of application a is exponentially distributed with means P_a^c. For a certain computing group (u, p, a, e), ECS e allocate $c_{u,e}$ computing capacity to the processing of tasks from u. Then, the task processing of an application can be modeled as an M/M/1 queueing system. The average service rate under given CPU resource and

TABLE 1. Table of notation.

Symbol	Definition
$\mathcal{U}$	Set of IoT nodes.
$\mathcal{P}$	Set of APs.
$\mathcal{G}$	Set of deployed ECSs.
$\mathcal{A}$	Set of applications.
x_u^a	Binary indicator that indicate if application a is installed on IoT node u.
$x_{u,p}$	Binary indicator that indicate if IoT node u is connect to AP p.
$y_{u,e}$	Binary indicator that indicate if application of u is assigned to ECS e.
$\psi_{p,e}$	Binary indicator that indicate if ECS e is placed to RAN covered by p.
S_e^a	Binary indicator that indicate if SF for a is deployed to ECS e.
P_a^c	Average computing intensity of tasks offloaded from application a.
P_a^m	Memory for processing tasks from application a.
ε_a^c	Required CPU for running an SF for a.
ε_a^m	Required memory resource for running an SF for a.
l_a^t	Average task size of application a.
l_a^r	Average computing result size of application a.
τ_a	Service latency constraint of application a.
V_a	Tolerable percentage of tasks from application a exceeding latency constraint.
λ_u	Average task arrival rate from IoT node u.
μ_u	Average service rate for tasks from IoT node u.
$c_{u,e}$	CPU that ECS e allocate to process tasks from u.
$m_{u,e}$	Memory that server e allocate to process tasks from u.
C_e	CPU occupied on the ECS e.
M_e	Memory occupied on the ECS e.
C	CPU capacity of an ECS.
M	Memory capacity of an ECS.
R	Network transmission rate.

computing intensity condition can be expressed as

$$\mu_u = \frac{c_{u,e}}{P_a^c}. \tag{1}$$

Although mean response time can reflect the performance, long-term service reliability is more important for continuously working applications and average response time cannot promise QoS of each task. Therefore, in this paper, the QoS of an application is set as the percentage of tasks that satisfy the latency constraint. For application a, latency constraint of task service is represented by τ_a, and V_a is the tolerable proportion of tasks whose computing period exceeds τ_a.

The distribution of computing service time (t) for a computing task stream offloaded from u follows [40]

$$f_D(t) = (\mu_u - \lambda_u)e^{-(\mu_u - \lambda_u)t}. \tag{2}$$

The percentile of tasks whose computing duration (D) exceed the computing latency constraint (t_d) can be expressed as [41]

$$\begin{aligned} P(D > t_d) &= 1 - F_D(t_d) \\ &= e^{(\lambda_u - \mu_u)t_d}. \end{aligned} \tag{3}$$

Then, if application a installed on IoT node u is assigned to be processed by ECS e, the required service rate is

$$\mu_u = \lambda_u - \frac{\ln V_a}{\tau_a - d_{u,e}}, \tag{4}$$

where $d_{u,e}$ represent the average transmission delay of the computing tasks from IoT nodes u to ECS e and computing responses from e to u.

$$\begin{aligned} d_{u,e} &= d_{u,p} + d_{p,e} + d_{e,p} + d_{p,u} \\ &= \frac{l_a^t + l_a^r}{r_{u,p}} + \frac{l_a^t + l_a^r}{r_{p,e}} \\ &= \frac{l_a^t + l_a^r}{r_{u,p}} + 2(1-\psi_{p,e})\frac{l_a^t + l_a^r}{R}, \end{aligned} \tag{5}$$

where $r_{u,p}$ is the transmission rate from the IoT node to AP, and R is the network transmission rate. If executed by a remotely deployed ECS, computing tasks of an application will experience two-hop transmission from AP to the SDN switch and then from the SDN switch to the ECS. Otherwise, there is no network transmission delay since ECS is directly associated with AP. It should be noted that the average transmission delay can be obtained by tracing and statistics in real networks.

C. COMPUTING RESOURCES OCCUPATION

To guarantee the QoS, ECSs should provide sufficient computing service rate required by task processing of applications. If application a installed on u is assigned to be executed at ECS e, according to (1) and (4), the CPU allocated to task stream offloaded from it should be

$$\begin{aligned} c_{u,e} &= \sum_{a\in\mathcal{A}} x_u^a y_{u,e} \mu_u P_a^c \\ &= \sum_{a\in\mathcal{A}} x_u^a y_{u,e} P_a^c (\lambda_u - \frac{\ln V_a}{\tau_a - d_{u,e}}). \end{aligned} \tag{6}$$

The memory resource required for the processing of tasks from a installed on IoT node u is

$$m_{u,e} = \sum_{a\in\mathcal{A}} x_u^a y_{u,e} p_a^m. \tag{7}$$

Then, under a specific application assignment condition, computing resources occupied on ECS e are

$$C_e = \sum_{u\in\mathcal{U}} c_{u,e} + \sum_{a\in\mathcal{A}} S_e^a \varepsilon_a^c, \tag{8}$$

$$M_e = \sum_{u\in\mathcal{U}} m_{u,e} + \sum_{a\in\mathcal{A}} S_e^a \varepsilon_a^m, \tag{9}$$

where ε_a^c and ε_a^m are the base CPU and memory resources required for running a SF for application a. S_e^a represents whether a SF for application a is installed on ECS e.

$$S_e^a = \begin{cases} 1, & \text{if } \sum_{u\in\mathcal{U}} x_u^a y_{u,e} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

D. SERVICE DEPLOYMENT PROBLEM

In this paper, we investigate the deployment of service and assignment of application with considering the collaboration among ECSs in regional IoT where edge computing technology is to be implemented. The service deployment includes the placement of ECSs and SFs. Considering a given regional IoT, as shown in figure 1, where intelligent applications are installed. The deployed ECSs and SFs should promise task processing of all applications in the system. The primary objective of this paper is set to minimize the number of required ECSs to be placed so that the network deployment cost can be reduced. In this paper, we consider two common computing resources(i.e., CPU and memory), then, the primary objective can be formulated as follows.

$$P1: \quad \min \sum_{p\in\mathcal{P}} N_p \tag{11}$$

$$s.t. \quad C_e \le C, \ \forall e, \tag{11a}$$

$$M_e \le M, \ \forall e, \tag{11b}$$

$$\sum_{a\in\mathcal{A}} x_u^a = 1, \ \forall u, \tag{11c}$$

$$\sum_{p\in\mathcal{P}} x_{u,p} = 1, \ \forall u, \tag{11d}$$

$$\sum_{e\in\mathcal{G}} y_{u,e} = 1, \ \forall u, \tag{11e}$$

$$\sum_{e\in\mathcal{G}} \psi_{p,e} = 1, \ \forall p, \tag{11f}$$

$$\sum_{p\in\mathcal{P}} \psi_{p,e} = 1, \ \forall e, \tag{11g}$$

$$N_p \in \{0,1\}, \tag{11h}$$

in which N_p indicates whether an ECS is deployed to the RAN covered by AP p. Constraints (11a) and (11b) are the resource constraints. Expression (11c) represents that only one application is installed on an IoT node. Constraints (11d) and (11e) represent that an IoT node can only be covered by one AP and assigned to one ECS. Constraints (11f)∼(11h) indicates that only one ECS can be deployed to one RAN. Besides, one ECS can only be mounted to one RAN.

After obtaining the minimum number of ECS to be placed, we can also get an initial application assignment result. However, some applications may be assigned to ECS remotely placed in other RAN, which increases the network forwarding load. By moving those application tasks to ECSs locally placed in RANs where they are oriented if required computing resources can be satisfied, the network transmission load can be reduced [42], [43]. Therefore, after determining the minimum number of required ECSs, we further decide the ascription of both ECSs to RANs and applications to ECSs to maximize the number of application tasks served by locally deployed ECSs.[1] This objective can be written as

$$P2: \quad \max \sum_{u\in\mathcal{U}} \sum_{p\in\mathcal{P}} \sum_{e\in\mathcal{G}} x_{u,p} y_{u,e} \psi_{p,e} \lambda_u \tag{12}$$

[1] other objectives like reducing service latency and QoS violations can also be investigated.

V. COLLABORATIVE SERVICE DEPLOYMENT AND APPLICATION ASSIGNMENT

This section describes our proposed collaborative service deployment and application assignment (ColSDA) heuristic algorithm. The purpose of edge service deployment is to find ECSs placement, SFP, and application assignment policy for a given regional IoT. Therefore, the ColSDA algorithm consists of four main steps: (1) obtain the minimum number of ECS to be placed. (2) balancing the computing load of ECSs and prepare for further operation. (3) placing ECSs to RANs. (4) moving and swapping some applications to improve the number of tasks assigned to locally deployed ECSs. The whole work procedure is revealed in algorithm. 1.

Algorithm 1 ColSDA Algorthim

```
Input: $\mathcal{N}, \mathcal{A}_{pro}, C, M, \mathcal{U}_{inf}$
Output: $\mathcal{U}_a^{opt}$
1: $\mathcal{U}_a^{ini}$ =MinRI($\mathcal{U}_{inf}, \mathcal{A}_{pro}, \mathcal{U}_{inf}$)  /** Obtaining the minimum number of ECSs to be placed and the initial assignment of applications to ECSs.
2: $\mathcal{U}_a^{bal}$ =LBRA($\mathcal{U}_a^{ini}, \mathcal{A}_{pro}, \mathcal{U}_{inf}$)  /** Balancing the computing load of ECSs and spare resources for further operations.
3: Determining the placement locations of ECSs by employing branch and bound algorithm with the objective of maximizing the number of tasks allocated to locally deployed servers.
   Updating the resource occupations information on ECSs and add the location information into $\mathcal{N}$.
4: $\mathcal{U}_a^{opt}$ =SeSw($\mathcal{U}_a^{bal}, \mathcal{A}_{pro}, \mathcal{N}, \mathcal{U}_{inf}$)  /**Maximizing the number of tasks processed by locally deployed ECSs.
```

in which, $\mathcal{N}$ is the network information including topology and application installing information. $\mathcal{A}_{pro}$ is the properties of applications including the average computing resources required for task processing (P_a^c, P_a^m), the base resources of SFs (ε_a^c, ε_a^m), average size of computing tasks and results (l_a^t, l_a^r), and the QoS (τ_a, V_a) of applications. $\mathcal{U}_{inf}$ is the resource requirement information of task processing that can be estimated according to (5) and (6). $\mathcal{U}_a^{ini}$, $\mathcal{U}_a^{bal}$ and $\mathcal{U}_a^{opt}$ are application assignment results. The details of each procedure are specified below.

In the work procedures of ColSDA algorithm, a min-resource ratio increase (MinRI) algorithm is proposed to determine the minimum number of ECSs to be placed. Then, in step 2, a load-balancing application block reassignment (LBRA) is designed to balance loads and get an initial application assignment result. After that, the branch and bound algorithm [44] is employed to find locations for placement of ECSs to maximize the number of application tasks assigned to their locally deployed ECSs. At last, a search and swap (SeSw) algorithm is designed to further increase the number of tasks assigned to locally deployed ECSs by considering the locations of both ECSs and applications. The proposed algorithms are illustrated in the following part.

The primary objective of this paper is to determine the minimum number of ECSs to be placed considering the simultaneous consumption of CPU and memory. Since the required resources for task processing of all applications can be pre-estimated, problem **P1** is a normal vector bin packing problem [45] without considering base resources for running SFs. The vector bin packing problem that dimension more than two is APX-hard [46]. Early first-fit decreasing (FFD) greedy algorithm is one suggested method for this problem. However, it hard to determine which element is the descending reference. Dot-product (DP) and norm-based greedy (EL2) algorithms were proposed in [45] to obtain superior performance by considering the relationship between residual space and items to be packaged. However, the weight factors of different required resources are hard to determine when considering the influence of SFs. Considering the diversity and preference property of resources' consumption by task processing of different applications, we propose a simple minimum resource ratio increase (MinRI) greedy algorithm to find an initial application assignment policy that can obtain the minimum number of ECSs to be placed. The MinRI algorithm can help to realize resources-complementary collaboration among ECS to improve resource utilization. Besides, by employing the normalized resource occupancy ratio, it is unnecessary to consider the weight factors for different computing resources.

Algorithm 2 MinRI Algorthim

```
Input: $\mathcal{U}_{inf}, \mathcal{A}_{pro}, C, M$
Output: $\mathcal{U}_a^{ini}$
1: $i = 1$  /**Initializing the number of ECS ($i$).
2: while $\mathcal{U} \neq \phi$ do
3:     for $u \in \mathcal{U}$ do
4:         $(c_u, m_u)$←Required resources for assigning $u$ to current ECS.
5:     end for
6:     $\mathcal{CU}$ ← IoT nodes whose required resources can be satisfied by current ECS.
7:     if $\mathcal{CU} = \phi$ then
8:         $i = i + 1$  /**Creating a new ECS.
9:     else
10:        $u^*$: if $\max(\frac{e_c+c_{u^*}}{C}, \frac{e_m+m_{u^*}}{M}) = \min\left\{\frac{e_c+c_{\mathcal{CU}}}{C}, \frac{e_m+m_{\mathcal{CU}}}{M}\right\}$ Selecting the application in $\mathcal{CU}$ which can obtain the minimum resource occupation ratio value after being packaged to current ECS.
11:        Assigning $u^*$ to the current ECS.
12:        $\mathcal{U} = \mathcal{U} \backslash u^*$.
13:    end if
14:    Updating $\mathcal{U}_a^{ini}$ and resources occupation information of current ECS.
15: end while
16: return $\mathcal{U}_a^{ini}$
```

Algorithm 2 is the pseudo-code of MinRI. The MinRI algorithm iteratively selects the application that can minimize the maximum resource occupation ratio of all resources after being packaged to currently opened ECSs. The MinRI firstly finds all applications whose required computing resources can be satisfied by the current ECS from unallocated applications. Assuming that the resources required by application from u is (c_u, m_u) and resource occupation of the current opened ECS is (e_c, e_m). Then the maximum resource ration after packaging u to ECS is $\max((e_c + c_u)/C, (e_m + m_u)/M)$. When there is no left application that can be packaged to the current ECS due to insufficient residual resources, MinRI will create and open a new ECS. Otherwise, MinRI selects the applications that can obtain the minimum resource ratio increase (u^*) and assign u^* to the current ECS. Then, the resource occupation information is updated. The MinRI algorithm keeps running until every application is assigned to an ECS.

Applications in the network will be initially assigned to ECSs after executing the MinRI algorithm. However, the location of applications is not considered in this step. Besides, a new ECS is created only when resources of the previous ECS is exhausted, which may lead to unbalanced resource occupation [47]. After executing MinRI, the last created ECS is usually unsaturated while other ECSs created earlier are all saturated that cannot package any other applications. Therefore, a load-balancing application block reassignment (LBRA) algorithm is proposed to balancing loads among ECSs and spare space for further application reassignment.

LBRA is detailed in algorithm 3. LBRA firstly groups the same kind applications assigned to the same ECS into an application blocks $\mathcal{A}_{block}$ based on the initial assignment result of MinRI (i.e., $\mathcal{U}_a^{ini}$). Then, LBRA adopts search and move operations to move application blocks among ECSs to balance computing load. Application blocks in saturated ECSs can be moved to unsaturated ECSs and resources can be spared for other movement operations. LBRA is continuously running to find an application assignment result with a smaller variance (σ) of resource occupation until no better result can be obtained within pre-defined search counts (T). In our simulations, the number of application blocks gets from MinRI is not too much since the same kind applications are usually allocated to one ECS. Therefore, we set $T = 10$ in our simulation is enough to get an acceptable result since this step is just an intermediate step to balance load and spare space for further operation. Besides, whether corresponding SF has been placed in the target ECS should be considered when estimating the required resource for an application or application block. For example (c_u, m_u) in step 4 of MinRI. Algorithms will firstly check if a SF for application installed on u has been deployed on e and indicate by a temporary binary indicator (α). Then, when trying to assign a installed on u to e the required resources are estimated by

$$c_{u,e} = c_u + (1-\alpha)\varepsilon_a^c \tag{13}$$
$$m_{u,e} = m_u + (1-\alpha)\varepsilon_a^m \tag{14}$$

Algorithm 3 LBRA Algorthim

```
Input: $\mathcal{U}_a^{ini}$, $\mathcal{A}_{pro}$, $C$, $M$
Output: $\mathcal{U}_a^{bal}$
 1: $\mathcal{A}_{block}$ ← Set of grouped application blocks.
 2: $\mathcal{G}$ ← Set of ECSs.
 3: Sorting $\mathcal{A}_{block}$ according to occupied resource ratio.
 4: $\mathcal{U}_a^{bal} = \mathcal{U}_a^{ini}$,  $\mathcal{U}_a^{tem} = \mathcal{U}_a^{ini}$.
 5: $\sigma^o$ ← Variance of current occupied resource ratio.
 6: $t = 0$ ← /**Initializing search count.
 7: while $t < T$ do
 8:   for $A \in \mathcal{A}_{block}$ do
 9:     $e_A$ ← ECS that $A$ is assigned to.
10:     $\mathcal{G}_o = \mathcal{G} \backslash e_A$ ← other ECSs
11:     for $e \in \mathcal{G}_o$ do
12:       $(c_A^e, m_A^e)$ ← Resources required by $A$ if processed
          at $e$.
13:       $(f_e^c, f_e^m)$ ← Residual resources of $e$.
14:       if $c_A^e \leq f_e^c$ && $m_A^e \leq f_e^c$ then
15:         Moving applications in $A$ from $e_A$ to $e$.
16:         Update $\mathcal{U}^{tem}$.
17:         break;
18:       end if
19:     end for
20:   end for
21:   Calculating current $\sigma$.
22:   if $\sigma < \sigma^o$ then
23:     $\sigma^o = \sigma$.
24:     $\mathcal{U}_a^{bal} = \mathcal{U}_a^{tem}$.
25:     $t = 0$.
26:   end if
27:   $t = t + 1$.
28: end while
29: return $\mathcal{U}_a^{bal}$
```

Following the above steps, the computing loads of ECSs can be balanced, and every ECS may have some idle resources. Then, following step 3 in CloSDA, we will place ECSs to appropriate RANs and aim at maximizing the number of computing tasks processed by the locally deployed ECSs. The problem is an obvious assignment problem and can be solved by branch and bound algorithm [44].

After placing ECSs to RANs in the system, SFP and application assignment imposed a significant influence on system performance. Although similar objectives (i.e., **P2**) have been investigated in previous work, multiple computing resources and the increased resource requirements caused by network latency have not been considered. These two facts increase the difficulty of tackling this problem. Since applications have been initially assigned by performing the above steps, SeSw algorithm is designed to move and swap applications to their locally deployed ECSs.

Figure 3 shows the sketch of move and swap operation, in which, A and B are two deployed ECSs. L_A and L_B represent application blocks generated from RANs where ECSs A

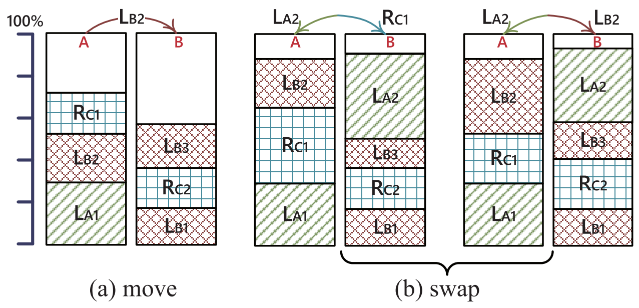


FIGURE 3. Sketch of move and swap operation.

and B are placed respectively. R_c represents the application block generated from a RAN without a deployed ECS. The move operation is performed when a favorable reward (e.g., more locally processed tasks, fewer occupied resources.) can be obtained by moving an application block and the required resources can be satisfied by the target ECS. The swap operation is triggered under conditions when a favorable reward can be obtained by moving the application block to a target ECS but residual resources on the target ECS are insufficient to satisfy the resources required by the application block to be moved. Then the algorithm will search and check if an application block assigning to the target server can be swap with the application to be moved without causing an unfavorable reward (e.g., more resources' occupation.). Moreover, when swapping the two application blocks, the required resources can both be satisfied.

```
Algorithm 4 SeSw Algorthim
Input: $\mathcal{U}_a^{bal}$, $\mathcal{A}_{pro}$, $\mathcal{N}$, $\mathcal{U}_{bal}$
Output: $\mathcal{U}_a^{opt}$
 1: $\mathcal{A}_{block} \leftarrow$ Set of grouped application blocks.
 2: $\mathcal{G} \leftarrow$ Set of ECSs.
 3: $\mathcal{U}_a^{tem} = \mathcal{U}_a^{bal}$
 4: repeat
 5:     for $A \in \mathcal{A}_{block}$ do
 6:         Performing move and swap operations.
 7:         Updating $\mathcal{U}_a^{tem}$.
 8:     end for
 9: until no block can be moved or swapped
10: repeat
11:     $\mathcal{U}_a^{tem}$.
12:     for $a \in \mathcal{A}$ do
13:         Performing move and swap operations.
14:         Updating $\mathcal{U}_a^{tem}$.
15:     end for
16: until no application can be moved or swapped
17: return $\mathcal{U}_a^{opt}$.
```

The SeSw algorithm is illustrated in algorithm 4, in which, move and swap operations are performed twice for considering application blocks and applications. In the first sub-step, on each ECS, application blocks are constructed according to the following principles based on their types and locations. 1) The same kind applications from one same RAN with an ECS deployed are grouped into one block. 2) The same kind applications from RANs without a deployed ECS are grouped into one block. SeSw can move an application block to its locally deployed ECS if an application block requires less CPU since network latency introduce a little more CPU requirement. This sub-step can provide more swap opportunities since application blocks grouped by applications from RANs without a deployed ECS usually occupy enough resources for the swap of application blocks. In the second step, the move and swap operations are conducted to move and swap applications, which can further assign applications to their locally deployed ECS. In real operation, SeSw will firstly check if an application (block) is from a RAN with an ECS and assigned to a remotely deployed ECS. Then, the algorithm attempt to move an application (block) assigned to a remotely deployed ECS to its locally deployed server. Otherwise, the algorithm will skip the current application (block) and check the next one (step 6-9 and step 13-15).

It should be noted that, we only consider if the required resources can be satisfied for moving a remotely processed application to its locally deployed ECS. Whether fewer resources are required is not considered since our aim is maximize the tasks processed by locally deployed ECS rather than minimizing resource occupation.

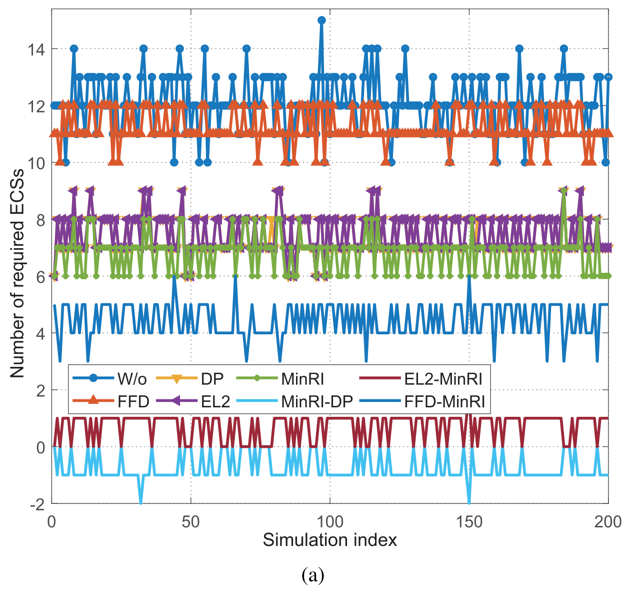
VI. PERFORMANCE EVALUATION

This section illustrates and discusses the simulations conducted in matlab 2019 to evaluate our proposed algorithms. Simulation parameters are detailed in table 2. In this paper, the resource of CPU capacity is indicated by the number of instructions that can be processed per-second (in *MIPS*). For each simulation, 20 kinds of applications are constructed. The properties of the application are randomly generated according to the values given in the table. Once an IoT node is created, the type of application installed on it and the RAN to which it belongs are randomly selected. In addition, we set the average task generation rate for IoT nodes to fluctuate slightly around the mean value.

TABLE 2. Simulation parameters.

parameter	value	parameter	value
C	200 *KMIPS*	l_a^t	[5 20] *Kbyte*
M	128 *GB*	l_a^r	[10 80] *Byte*
p_a^c	[10, 50] *MIPS*	τ_a	[20 100] *ms*
p_a^m	[600, 1000] *MByte*	V_a	1% ∼ 10%
ε_a^c	[800 1800] *MIPS*	λ_u	[2 10] / second
ε_a^m	[800 1500] *Mbyte*	R	54 Mbps

Figure 4 shows the minimum number of ECSs required for satisfying all applications obtained by different methods. Figure. 4(a) shows the statistical result of each simulation, in which 10 RANs and 800 IoT nodes are created. We can find that the number of required ECSs can be greatly reduced by establishing collaboration among ECSs.



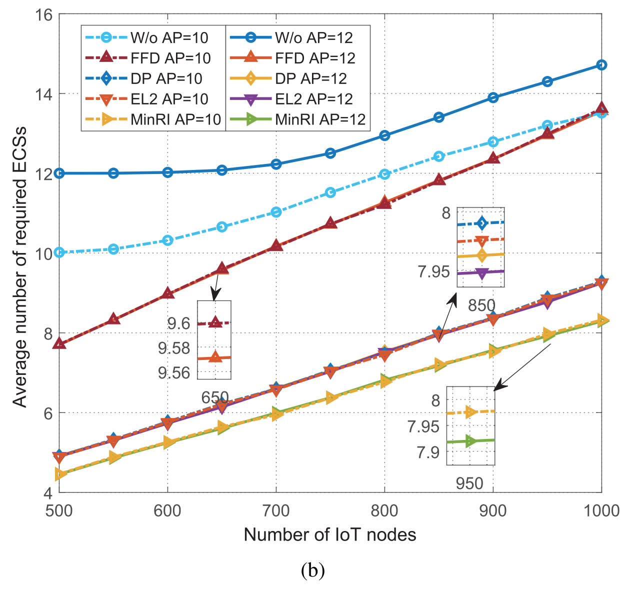


FIGURE 4. Average number of required ECSs.

Without collaboration, the required ECSs more than RANs since the computing loads of some RANs exceed the capacity of an ECS. However, the computing loads of overloaded ECSs can be released by collaborative ECSs, which reduce the number of ECS to be placed. Besides, comparing with other algorithms, our proposed MinRI algorithm can package all applications with minimum ECSs, which can be observed from the differences between MinRI and DP, EL2 as well as FFD [45]. Figure 4(b) is the relationship between the average result of minimum number of required ECSs and IoT nodes. During this simulation, we create 800 network scenarios for each IoT number condition and 500 times simulations with different application properties for every network scenario. We can also observe that the number of ECSs required for the original network is directly related to the number of RANs. And since the computational load of some RANs is easily overloaded, the number of ECSs required is easily over the number of RANs in the network. When collaboration established, the number of required ECSs is decided by the number of IoT nodes rather than the RANs in the network. The number of ECSs to be placed obtained by FFD is far more than others, and results get by DP and EL2 are almost the same which in compliance with results declared in [45]. Since SFs are required and affect the total amount of required resources, the weight for different resources in DP and EL2 is hard to decide. So, this paper sets the weights of the different resources in the comparison algorithm DP and EL2 to be equal. The minimum number of ECSs to be placed can be obtained by our proposed MinRI algorithm as compared with FFD, DP, and EL2 algorithms.

Apart from the minimum number of required ECS, the resource occupation of saturated ECSs can also reflect whether the heterogeneous resource-consuming applications on the ECS are placed appropriately to fully utilize the resources. The saturated ECS defined in this paper is ECSs whose residual resources are insufficient to satisfy any other application in the network. We define the resource occupation as the average ratio of occupied resources to total resources of an ECS. Figure. 5 shows the average resource occupation of

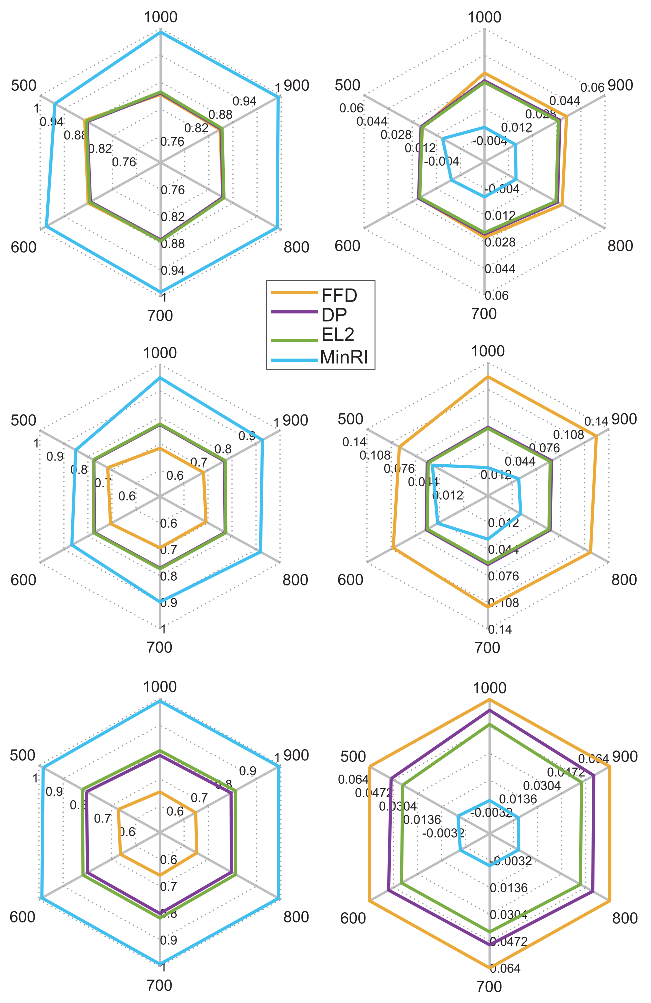


FIGURE 5. Average resource occupation of saturated ECSs.

saturated ECSs under different IoT node numbers (500-1000). Simulation is conducted with 100 groups of generated application properties for each node number, and 300 network scenarios under each application properties. We conduct three experiments with different range of P_a^c and P_a^m. In figure 5, the left side are the average resource occupation rations and right side are the corresponding average variance of resource occupation ratio. Since SFs require base resources, more SFs may be installed by algorithms and more ECSs are created. To avoid the effect of resources occupied by SFs, and demonstrated that MinRI could adapt to the discrepancy of resource preferences between applications. We select the first three created ECSs since simulations always require more than four ECSs in our simulations. We can learn that our proposed MinRI algorithm can achieve higher resource utilization on saturated ECSs as compared to FFD, DP, and EL2. The reason is that resource-complementarity is considered in MinRI, which can make resources are utilized more balanced and reduce the possibility that one kind of resource of edge servers is exhausted quickly. The result indicates that the allocation of applications finished by MinRI is more appropriate to fully utilize resources, which is beneficial to package all applications with fewer ECSs.

In addition, to evaluate the effectiveness of the operation after placing the ECS, simulations are conducted to compare the ratio of computational tasks assigned to locally deployed ECSs and the number of installed SFs. Since we consider multiple kind of computing resources that have not been considered in previous works, we only compare the results of three operations: before SeSw, SeSw with the goal of maximizing the number of tasks handled by locally deployed ECS (MaxLocal), and SeSw seeking minimum resources occupation (w/o MaxLocal).

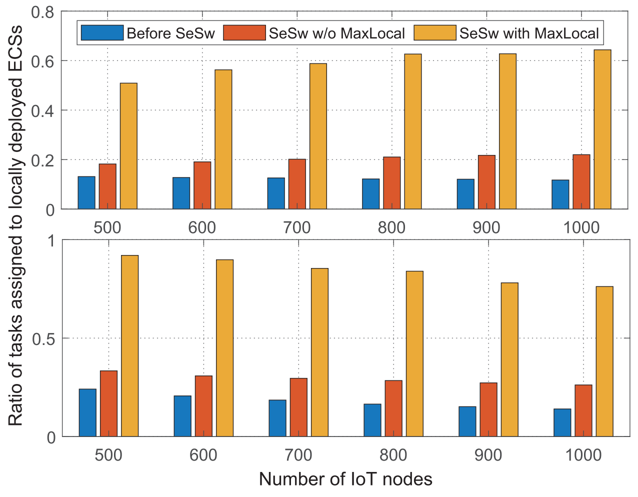


FIGURE 6. Ratio of application tasks assigned to be processed by locally deployed ECSs.

Figure 6 shows the ratio of tasks assigned to be processed by locally deployed ECSs. The upper sub-figure is the result considering all applications in the system. The lower sub-figure is the result that only considering applications from RANs with an ECS deployed. For simple presentation, we use global-local (GL) to indicate the ratio of tasks assigned to locally deployed ECSs when considering the whole system, and local-local (LL) to indicate that when only considering applications from RANs with an ECS deployed. We generate 300 groups of application properties for every point and 150 random network topology under each application properties condition. The number of RANs is set to be 12. We can find that a small number of tasks are assigned to locally deployed ECSs before performing SeSw. Lots of applications are set to be integrate processed to reduce resource occupation by MinRI, and resources-complementary collaborations are established to reduce the number of ECSs. Before performing SeSw, LL decreases as IoT nodes increase. Because more applications lead to more ECSs deployed, which results in a great increase in the total number of related tasks while less increase in tasks assigned to local servers. The GL experiences a slight decrease as IoT nodes increase since some more overloaded applications have to be assigned to remote ECSs. After performing SeSw, the number of tasks assigned to local ECSs is significantly improved, especially when employing Maxlocal. When only considering to reduce resources-consumption (w/o MaxLocal), an application block or application assigned to a remote ECS can only be moved to the local ECS when requiring less resources. So, some applications can not be moved to local ECSs when corresponding SFs are not installed since considerable base resources are required. Both GL and LL are greatly improved after executing SeSw with MaxLocal. While GL tends to be stable when IoT nodes more than 800 while the LL still experiences a decrease. ECSs are usually placed in RANs with heavier loads to maximize the number of tasks assigned to locally deployed ECSs. Then, the later created ECSs are placed to RANs with lighter loads. Applications in these RANs contribute more to the total number of related tasks as compared with tasks assigned to local ECSs. Besides, RANs with overloaded applications increases as IoT nodes increase, which contributes more to the ration of tasks being processed by remote ECSs. In addition, more applications will result in more saturated ECSs, which decreases the feasibility of performing move and swap operations. Finally, GL will be equal to LL when every RAN with an ECS placed.

Figure 7 shows the number of installed SFs, in which, the simulation settings are same as that in figure 6. We can find that collaboration among ECSs can reduce the number of installed SFs since required ECS is reduced and part congeneric applications are integrated assigned to one SF. In our simulations, applications are randomly installed so almost all types of applications are installed in a RAN. Therefore, in the systems without collaboration and the systems adopt collaboration with MaxLocal, the numbers of installed SFs are significantly higher than that before performing SeSw, and present obvious increases as the IoT nodes increases. The reason is that corresponding SFs is required by an ECS for providing services for all applications assigned to it.

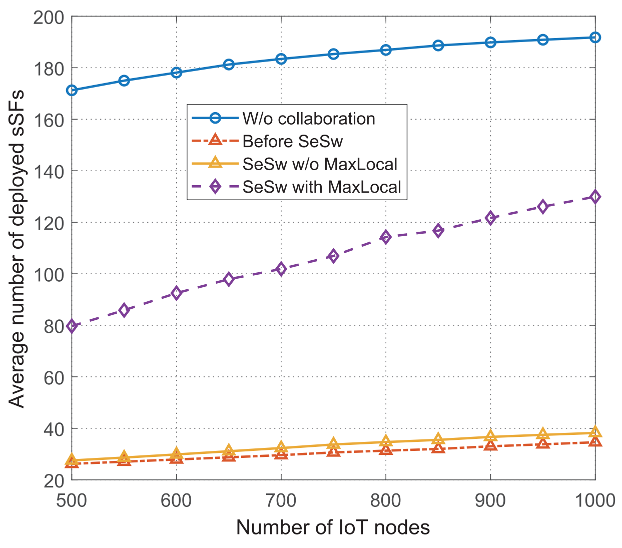


FIGURE 7. Average number of deployed SFs.

In original network and collaboration with Maxlocal, most applications are assigned to locally placed ECSs. When collaboration employed, as compared to SwSw with MaxLocal SeSw without MaxLocal can obtain far less installed SFs and installed SFs only increase slightly as IoT nodes increase. These two results benefit from integration processing collaboration in which the congeneric applications are integrally processed by one SF. The results demonstrate that collaboration can provide the potential to reduce deployment and management expenditure. Besides, better results can be expected to be acquired in real networks where the distribution of intelligent applications are not randomly deployed.

Figure 8 shows the average resource utilization ratio of ECSs. We can observe that collaboration can significantly improve the resource occupation rate of ECSs, which reflect the resources of ECSs are more fully utilized because computing tasks are gathered to be processed by fewer ECSs. The resource utilization in systems without collaboration increased obviously as application increase. The reason is that unsaturated ECS in systems undertake more computation work as applications increase. The resource utilization when employing collaboration without MaxLocal is slightly higher than that before SeSw, while resource utilization when employing collaboration with MaxLocal is obviously higher than that before SeSw. The reason is that SeSw with MaxLocal move and exchange applications with the objective of maximizing the number of computing tasks, which makes it reassign applications to their locally deployed ECSs without considering whether required resources are more than currently occupied resources. The amount of occupied resource may increase if moving an application from a remote ECS to its locally deployed ECS, especially when a new SF is required to be installed. In comparison, SeSw without MaxLocal only reassigns an application to its locally deployed ECS when its required CPU resource is less or equal

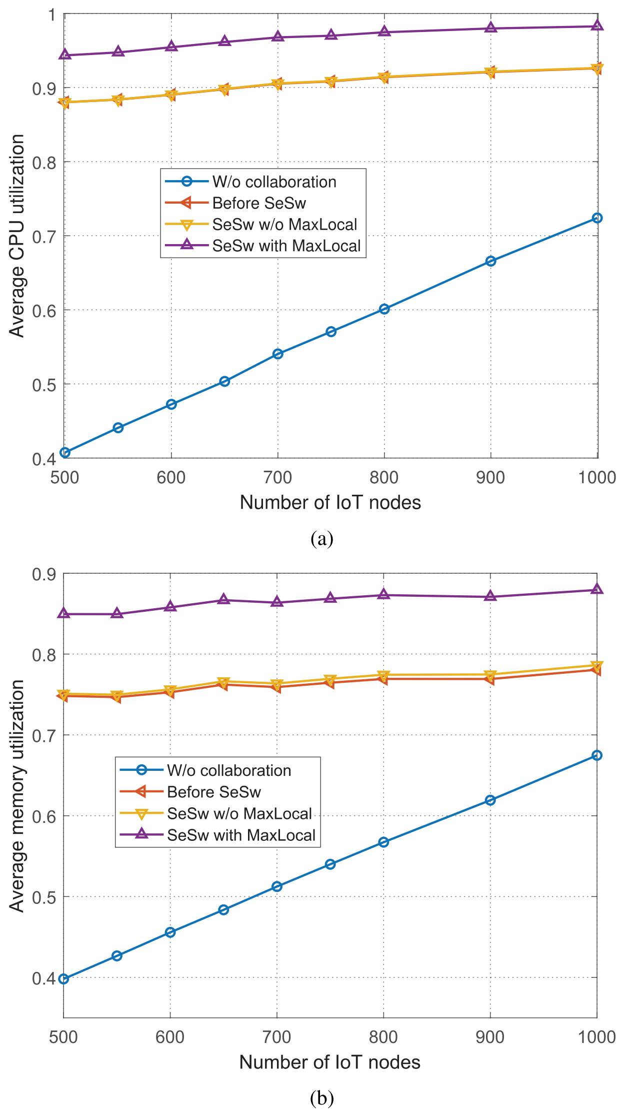


FIGURE 8. Average resource utilization. (a) CPU. (b) memory.

than currently occupied resource, which greatly reduces the number of applications that can be moved. Thus, the resource occupation after executing SeSw without MaxLocal is almost equal to that before SeSw, and the memory occupation ratio presents a little improvement.

Figure 9 displays an instance of resource occupation results during the procedure of finding the service deployment and application assignment result, where figure 9(a) shows the resources required by every RAN in the original system. Figure 9(b)$\sim$ 9(e) are the resource occupation results obtained by performing FFD, EL2, DP, and MinRI to determine the number of required ECSs. Compared with the original method of deploying ECSs for every RAN, the collaboration among edge servers can satisfy QoS of all applications in regional IoT with fewer ECS. Besides, comparing with FFD, DP, and EL2, our proposed MinRI can obtain minimum ECSs to be placed, which has been demonstrated

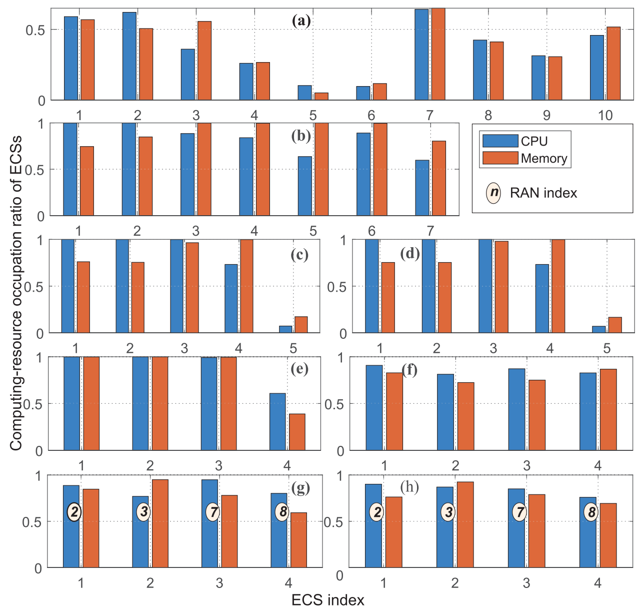


FIGURE 9. A resource occupation instance of one simulation. (a) original IoT without collaboration. (b)∼(e) the resource occupation obtained by employing FFD, DP, EL2, and MinRI in the step of determining the number of required ECSs. (f) resource occupation after the LBRA and placement. (g) SeSW without MaxLocal. (h) SeSw considering MaxLocal.

in figure 4. Also, the resource occupation of saturated ECSs obtained by MinRI is more balanced and fulfilled, which has been demonstrated in figure 5. Figure 9(f) is the balanced resource occupation after performing LBRA and placing ECSs, in which data of figure 9(e) are the input information. We can find that resource occupation are balanced and space for the following SeSw operation are obtained. Figure 9(g) and figure 9(h) are the final deployment results obtained by SeSw without MaxLocal and with MaxLocal, respectively. RAN index represents the RAN where the ECS is placed to.

VII. CONCLUSION

In this paper, we investigated the service deployment in regional EdgeIoT by exploring the collaboration among ECSs. Three feasible collaboration methods were introduced by considering the resources-limitation of ECSs, the heterogeneous resources' consumption of applications, and base resources' consumption of SFs. Then, we formulated the service deployment problem, and the ColSDA algorithm was proposed to obtain the service deployment policy for a give EdgeIoT, including the placement of ECSs and SFs, as well as the assignment of applications to ECSs. We formulated the primary objective of minimizing the number of ECSs to be placed as a vector bin packing problem, and the MinRI algorithm was proposed to obtain the minimum ECSs. Then, computing loads of ECSs were balanced by the LBRA algorithm to prepare for further operation. After that, ECSs were placed to RANs by employing the branch and bound algorithm. At last, the SeSw algorithm was proposed to further increase tasks assigned to locally deployed ECSs. Simulations results demonstrated that collaboration among ECSs can significantly reduce the number of required ECSs. Comparing with previous algorithms, the proposed MinRI algorithm in the ColSDA solution can obtain minimum ECSs and higher resource utilization. The forwarding load of network switch can be greatly released by performing SeSw with Maxlocal to assign most of the tasks to locally deployed ECSs. Therefore, ColSDA can be employed to deploy edge computing services for regional IoT.

For future work, network transmission capacity that simplified in this paper and more general network can be investigated. Besides, more real applications that require different edge computing service procedure can be investigated. Apart from maximizing the number of tasks processed by locally deployed ECSs that studied in this paper, more optimization objectives can be explored for ECSs and SFs placement after obtaining the minimum ECSs to be placed, such as minimizing system delay and maximizing the average QoS.

REFERENCES

[1] 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.

[2] G. Gui, M. Liu, F. Tang, N. Kato, and F. Adachi, "6G: Opening new horizons for integration of comfort, security and intelligence," *IEEE Wireless Commun.*, early access, Mar. 3, 2020, doi: 10.1109/MWC.001.1900516.

[3] G. Avino, P. Bande, P. A. Frangoudis, C. Vitale, C. Casetti, C. F. Chiasserini, K. Gebru, A. Ksentini, and G. Zennaro, "A MEC-based extended virtual sensing for automotive services," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 4, pp. 1450–1463, Dec. 2019.

[4] W. Tu, F. Pop, W. Jia, J. Wu, and M. Iacono, "High-performance computing in edge computing networks," *J. Parallel Distrib. Comput.*, vol. 123, p. 230, Jan. 2019.

[5] D. Wu, J. Yan, H. Wang, and R. Wang, "User-centric edge sharing mechanism in software-defined ultra-dense networks," *IEEE J. Sel. Areas Commun.*, early access, Apr. 16, 2020, doi: 10.1109/JSAC.2020.2986871.

[6] 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.

[7] J. Ren, G. Yu, Y. Cai, and Y. He, "Latency optimization for resource allocation in mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5506–5519, Aug. 2018.

[8] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Trans. Commun.*, vol. 66, no. 4, pp. 1594–1608, Apr. 2018.

[9] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2633–2645, Aug. 2018.

[10] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based IoT," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2146–2153, Jun. 2018.

[11] A. Ndikumana, N. H. Tran, T. M. Ho, Z. Han, W. Saad, D. Niyato, and C. S. Hong, "Joint communication, computation, caching, and control in big data multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1359–1374, Jun. 2020.

[12] A. Yousefpour, A. Patil, G. Ishigaki, I. Kim, X. Wang, H. C. Cankaya, Q. Zhang, W. Xie, and J. P. Jue, "FOGPLAN: A lightweight QoS-aware dynamic fog service provisioning framework," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5080–5096, Jun. 2019.

[13] D. Wu, Q. Liu, H. Wang, D. Wu, and R. Wang, "Socially aware energy-efficient mobile edge collaboration for video distribution," *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2197–2209, Oct. 2017.

[14] R. Yu, J. Ding, S. Maharjan, S. Gjessing, Y. Zhang, and D. H. K. Tsang, "Decentralized and optimal resource cooperation in geo-distributed mobile cloud computing," *IEEE Trans. Emerg. Topics Comput.*, vol. 6, no. 1, pp. 72–84, Jan. 2018.

[15] X. Yuan, G. Min, L. T. Yang, Y. Ding, and Q. Fang, "A game theory-based dynamic resource allocation strategy in geo-distributed datacenter clouds," *Future Gener. Comput. Syst.*, vol. 76, pp. 63–72, Nov. 2017.

[16] G. Gui, Z. Zhou, J. Wang, F. Liu, and J. S. Sun, "Machine learning aided air traffic flow analysis based on aviation big data," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 4817–4826, May 2020.

[17] C. Chen, L. Liu, T. Qiu, K. Yang, F. Gong, and H. Song, "ASGR: An artificial spider-web-based geographic routing in heterogeneous vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 5, pp. 1604–1620, May 2019.

[18] Z. Xu, W. Gong, Q. Xia, W. Liang, O. Rana, and G. Wu, "NFV-enabled IoT service provisioning in mobile edge clouds," *IEEE Trans. Mobile Comput.*, early access, Feb. 10, 2020, doi: 10.1109/TMC.2020.2972530.

[19] T. Subramanya, D. Harutyunyan, and R. Riggio, "Machine learning-driven service function chain placement and scaling in MEC-enabled 5G networks," *Comput. Netw.*, vol. 166, Jan. 2020, Art. no. 106980. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128619310254

[20] N. Yu, Q. Xie, Q. Wang, H. Du, H. Huang, and X. Jia, "Collaborative service placement for mobile edge computing applications," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

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

[22] K. Wang, H. Yin, W. Quan, and G. Min, "Enabling collaborative edge computing for software defined vehicular networks," *IEEE Netw.*, vol. 32, no. 5, pp. 112–117, Sep. 2018.

[23] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, Oct. 2017.

[24] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 10, pp. 2866–2880, Oct. 2016.

[25] S. Lee, S. Lee, and M.-K. Shin, "Low cost MEC server placement and association in 5G networks," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2019, pp. 879–882.

[26] Y. Ren, F. Zeng, W. Li, and L. Meng, "A low-cost edge server placement strategy in wireless metropolitan area networks," in *Proc. 27th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2018, pp. 1–6.

[27] K. Cao, L. Li, Y. Cui, T. Wei, and S. Hu, "Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing," *IEEE Trans. Ind. Informat.*, early access, Feb. 24, 2020, doi: 10.1109/TII.2020.2975897.

[28] 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.

[29] L. Chen, C. Shen, P. Zhou, and J. Xu, "Collaborative service placement for edge computing in dense small cell networks," *IEEE Trans. Mobile Comput.*, early access, Oct. 7, 2019, doi: 10.1109/TMC.2019.2945956.

[30] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Joint service placement and request routing in multi-cell mobile edge computing networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2019, pp. 10–18.

[31] T. He, H. Khamfroush, S. Wang, T. La Porta, and S. Stein, "It's hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018, pp. 365–375.

[32] S. Pasteris, S. Wang, M. Herbster, and T. He, "Service placement with provable guarantees in heterogeneous edge computing systems," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2019, pp. 514–522.

[33] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jul. 2018, pp. 66–73.

[34] S. Yang, F. Li, M. Shen, X. Chen, X. Fu, and Y. Wang, "Cloudlet placement and task allocation in mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5853–5863, Jun. 2019.

[35] J. Meng, C. Zeng, H. Tan, Z. Li, B. Li, and X.-Y. Li, "Joint heterogeneous server placement and application configuration in edge computing," in *Proc. IEEE 25th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2019, pp. 488–497.

[36] M. Guo, L. Li, and Q. Guan, "Energy-efficient and delay-guaranteed workload allocation in IoT-edge-cloud computing systems," *IEEE Access*, vol. 7, pp. 78685–78697, 2019.

[37] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[38] G. Zhao, H. Xu, S. Chen, L. Huang, and P. Wang, "Joint optimization of flow table and group table for default paths in SDNs," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1837–1850, Aug. 2018.

[39] X. Li, D. Li, J. Wan, C. Liu, and M. Imran, "Adaptive transmission optimization in SDN-based industrial Internet of Things with edge computing," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1351–1360, Jun. 2018.

[40] U. N. Bhat, *An Introduction to Queueing Theory: Modeling and Analysis in Applications*, 2nd ed. Boston, MA, USA: Birkhäuser, 2015.

[41] G. Giambene, *Queuing Theory and Telecommunications: Networks and Applications*, 2nd ed. Cham, Switzerland: Springer, 2014. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4614-4084-0

[42] P. Wang, H. Xu, L. Huang, C. Qian, S. Wang, and Y. Sun, "Minimizing controller response time through flow redirecting in SDNs," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 562–575, Feb. 2018.

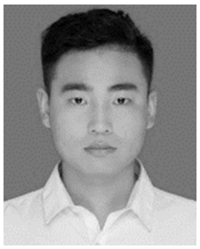
[43] C. Chen, J. Hu, T. Qiu, M. Atiquzzaman, and Z. Ren, "CVCG: Cooperative V2 V-aided transmission scheme based on coalitional game for popular content distribution in vehicular ad-hoc networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 12, pp. 2811–2828, Dec. 2019.

[44] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, vol. 21, 5th ed. Cham, Switzerland: Springer, 2012.

[45] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder. (2011). *Heuristics for Vector Bin Packing*. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2011/01/VBPackingESA11.pdf

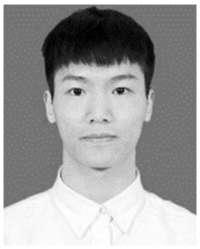
[46] G. J. Woeginger, "There is no asymptotic PTAS for two-dimensional vector packing," *Inf. Process. Lett.*, vol. 64, no. 6, pp. 293–297, Dec. 1997.

[47] L. Lv, Y. Zhang, Y. Li, K. Xu, D. Wang, W. Wang, M. Li, X. Cao, and Q. Liang, "Communication-aware container placement and reassignment in large-scale Internet data centers," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 540–555, Mar. 2019.



YAN CHEN received the B.S. degree in information engineering from the China University of Mining and Technology, in 2016, where he is currently pursuing the Ph.D. degree in information and communication engineering with the School of Information and Control Engineering. His research interests include edge computing, the IoT, and wireless networks.

YANJING SUN (Member, IEEE) received the Ph.D. degree in information and communication engineering from the China University of Mining and Technology, in 2008. He has been a Professor with the School of Information and Control Engineering, China University of Mining and Technology, since July 2012. His current research interests include wireless communication, embedded real-time systems, wireless sensor networks, cyber-physical systems, and so on. He is a Council Member of the Jiangsu Institute of Electronics and a member of the Information Technology Working Committee of the China Safety Production Association.



TIANXIN FENG received the B.S. degree in computer science and technology from the China University of Mining and Technology, in 2018, where he is currently pursuing the M.S. degree in information and communication engineering with the School of Information and Control Engineering. His research interests include the Internet of Things and wireless networks.

SONG LI (Member, IEEE) received the B.S. and M.S. degrees in communication and information system from Tianjin University, Tianjin, China, in 2007 and 2009, respectively, and the Ph.D. degree in signal and information processing from the Beijing University of Posts and Telecommunications, Beijing, China, in 2012. He is currently an Associate Professor with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, China. His current research interests include 5G wireless networks, cyber-physical systems, and the industrial Internet of Things.