Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

# Cost-effective resource segmentation in hierarchical mobile edge clouds*

Ming-shuang JIN[1], Shuai GAO‡[1], Hong-bin LUO[2], Hong-ke ZHANG[1]

[1]*School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China*

[2]*School of Computer Science and Engineering, Beihang University, Beijing 100083, China*

E-mail: mshjin@bjtu.edu.cn; shgao@bjtu.edu.cn; luohb@buaa.edu.cn; hkzhang@bjtu.edu.cn

Received Apr. 2, 2018; Revision accepted Sept. 14, 2018; Crosschecked Aug. 23, 2019

**Abstract:** The fifth-generation (5G) network cloudification enables third parties to deploy their applications (e.g., edge caching and edge computing) at the network edge. Many previous works have focused on specific service strategies (e.g., cache placement strategy and vCPU provision strategy) for edge applications from the perspective of a certain third party by maximizing its benefit. However, there is no literature that focuses on how to efficiently allocate resources from the perspective of a mobile network operator, taking the different deployment requirements of all third parties into consideration. In this paper, we address the problem by formulating an optimization problem, which minimizes the total deployment cost of all third parties. To capture the deployment requirements of the third parties, the applications that they want to deploy are classified into two types, namely, computation-intensive ones and storage-intensive ones, whose requirements are considered as input parameters or constraints in the optimization. Due to the NP-hardness and non-convexity of the formulated problem, we have designed an elitist genetic algorithm that converges to the global optimum to solve it. Extensive simulations have been conducted to illustrate the feasibility and effectiveness of the proposed algorithm.

**Key words:** Edge clouds; Edge computing; Edge caching; Resource segmentation; Virtual machine (VM) allocation

https://doi.org/10.1631/FITEE.1800203 **CLC number:** TP393.1

## 1 Introduction

### 1.1 Background and motivation

With the explosive growth in mobile traffic, it is difficult for current fourth-generation (4G) networks to satisfy the requirements (i.e., high data rate and low delay) of the representative applications in fifth generation (5G) such as virtual reality and automatic driving. The fundamental cause of the problem is that the nodes in the current network are all design-specific devices, leading to high complexity and poor flexibility and scalability of the network (Zhang et al., 2016). In addition, network customization, in-network caching, and computing are all impossible. In recent years, a large number of works have focused on research into software-defined mobile networks, including SoftRAN (Gudipati et al., 2013), virtual evolved packet core (vEPC) (Sama et al., 2015), caching in mobile networks (Wang et al., 2014), and mobile edge computing (Taleb et al., 2017). In addition, standardization organizations such as the Third Generation Partnership Project (3GPP, https://www.3gpp.org/ftp/specs/archive/23_series/23.501/) and Next Generation Mobile Networks (NGMN, https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 61972026)

ORCID: Ming-shuang JIN, http://orcid.org/0000-0002-1139-8355

© Zhejiang University and Springer-Verlag GmbH Germany, part of Springer Nature 2019

2015/NGMN_5G_White_Paper_V1_0.pdf) have all envisioned their 5G architecture based on the technology of software-defined network (SDN), network function virtualization (NFV), and cloud computing.

However, the infrastructure of the current mobile network cannot support the above mentioned development requirements. Therefore, 5G network cloudification has been an inevitable trend in the network realm (Sousa et al., 2016). For example, China Telecom released a white paper clarifying integration of the 5G network and the cloud (China Telecom, 2016). China Unicom claimed that thousands of edge data centers will be implemented in the near future (http://www.lightreading.com/data-center/data-center-infrastructure/china-unicom-to-build-thousands-of-edge-dcs/d/d-id/737151?). In addition, the European Telecommunications Standards Institute (ETSI) has formed a specific group called "Multi-access Edge Computing" (MEC), which has released a series of standards for mobile edge computing (MEC, 2016, 2018).

There has been considerable research in the area of edge clouds (Baktir et al., 2017; Mach and Becvar, 2017; Mao et al., 2017; Wang SQ et al., 2017; Abbas et al., 2018; Roman et al., 2018). Specifically, except for flexible networking for the mobile network operator (MNO), which is out of the scope of this work, cloudification of the 5G network enables third parties to deploy their own applications at the edge of clouds by leasing cloud servers from the MNO, thus providing a better quality of experience (QoE) to their customers. For example, video providers could place caching and transcoding instances in the edge clouds to provide a high-quality video on demand and free viewing of live broadcasting services (Bilal and Erbad, 2017). Transport systems could deploy local caching and image recognition instances for functions such as target tracking and real-time traffic guidance (Enayet et al., 2018; Zhou et al., 2018). Game companies could place edge servers so as to enable players to pre-fetch the pre-rendered environment frames, thus significantly improving the QoE of virtual reality (VR) games (Lai et al., 2017).

All the above-mentioned edge deployments are based on virtual machine (VM) technologies (Manzalini et al., 2013); the efficiency of VM allocation is a key advantage for both MNOs and third parties in the newly formed edge cloud market. In Fig. 1, we briefly show the workflow for application deployment in edge clouds, which conforms to the reference framework in MEC (MEC, 2016). It is obvious that third parties cannot initialize or start running their applications until the MNO has accomplished resource allocation in the mobile edge orchestrator. Nevertheless, the resource allocation algorithm has not been specified and is just what we focus on in this work.

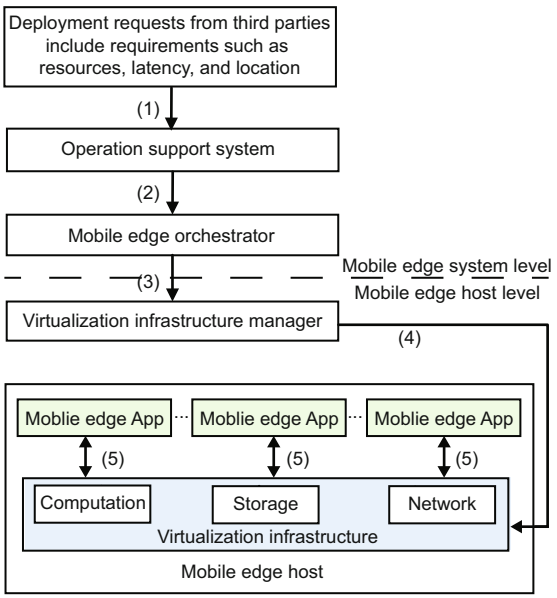


**Fig. 1 Workflow for application deployments in edge clouds**

## 1.2 Challenges and contributions

Compared with VM allocation in traditional cloud data centers (Mann, 2015), VM allocation in edge clouds faces new challenges due to new features in both the cloud architecture and emerging applications. Traditional cloud data centers are geographically distributed over a wide area. Therefore, due to the long distance from end users, the delay performance varies greatly (Alicherry and Lakshman, 2012). In comparison, the edge cloud of a 5G MNO is a hierarchical architecture, which consists of multiple edge data centers even at the base station (BS) level, providing excellent latency performance at the millisecond level. This architectural advancement in the edge cloud has led to booming development in

emerging latency-sensitive applications such as interactive VR games, bringing new demands for deployment to the edge cloud market. In summary, a new VM allocation algorithm is needed in the context of edge clouds. In particular, we make three contributions: (1) Based on the hierarchical edge cloud model of a 5G MNO and the detailed application deployment requirements of multiple third parties, we formulated the VM allocation problem, which is called a resource segmentation problem (RSP), as a mixed-integer non-linear programming (MINLP) problem. (2) To solve the NP-hard and non-convex RSP, we designed an elitist genetic algorithm (eGA). To guarantee the feasibility of all offspring, the elitist selection strategy has been executed in every iteration. (3) We conducted extensive simulations to evaluate the feasibility and effectiveness of the proposed eGA. Specifically, the VM allocation performance was analyzed under different deployment requirements of third parties.

## 2 Related work

### 2.1 Resource allocation in the area of edge caching

Ding et al. (2017) discussed the auction mechanism when allocating the cache space in the cache-enabled BS. Peng et al. (2016) solved the cache size allocation problem with the aim of maximizing the user success probability influenced by the wireless channel statistics and file popularity. Similarly, Zhang et al. (2017) determined the cache size deployed at multiple small base stations (SBSs) and macro base stations (MBSs) to maximize the network capacity for a better QoE in file transferring. Other works focused on obtaining the cache provision mechanism by designing a proper cache strategy. For example, Ghoreishi et al. (2016) determined the cache strategy by optimizing the trade-off between the cost of caching and bandwidth savings in the hierarchical in-network caching architecture.

The above-mentioned works all focused on obtaining the cache size provisioning results from the perspective of content providers (CPs) by maximizing the cache performance/benefit. In comparison, we focus on allocating both computation and storage resources from the perspective of an MNO, taking the CPs' cache deployment requirements (obtained from a strategy similar to the above-mentioned cache provisioning approach) as input parameters or constraints.

### 2.2 Resource allocation in the area of edge computing

Wang W et al. (2017) jointly solved the VM allocation problem and the workload assignment problem for the computation-intensive applications (CIAs). Tong et al. (2016) designed a hierarchical cloud architecture and allocated computation capacity among different tiers to efficiently serve the total workloads. However, the above-mentioned works considered only resource allocation for computation-intensive applications. In comparison, we consider the deployment requirements of both computation- and storage-intensive applications (SIAs).

### 2.3 Edge resource provisioning from other perspectives

Some works focused on edge resource provisioning from other perspectives. For example, Yin et al. (2017) proposed a framework that could discover the unforseen edge server location, which makes resource provisioning at the server level (other than the VM level) more efficient. Similarly, Wang S et al. (2017) solved the application placement problem through graph mapping (also at the server level). In comparison, we assume that the number of edge servers is fixed and focus on VM-level cost-effective resource provisioning.

The only work that considered both computation and storage resources in VM-level resource allocation is Tang et al. (2016), which jointly allocates computation-intensive VMs for network transmission of an MNO and storage-intensive VMs for in-network caching of a content provider. In comparison, we do not consider the VMs that the MNO uses for its own networking, but the VMs that are leased to the third parties for their application deployments (according to the process in Fig. 1).

## 3 Problem formulation

In this section, we first describe the system model, in which an MNO provides the hierarchical edge cloud infrastructure and multiple third parties deploy their applications in the edge cloud. Then

we describe the deployment requirements of both computation- and storage-intensive applications. Finally, the resource segmentation problem is formulated based on the system model.

### 3.1 System model

We consider the system model in Fig. 2, where an MNO is the owner of the edge clouds, enabling third parties to deploy their applications on demand at the network edge to provide flexible service capabilities to users in target areas that they want to cover. For simplicity, in our model we consider the situation where multiple third parties are offering services to their users in a certain BS through the edge clouds provided by an MNO.

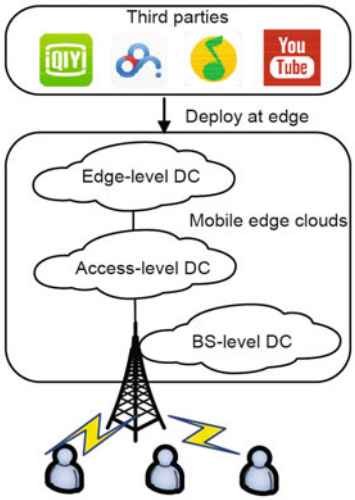


**Fig. 2 System model (DC: data center)**

Without loss of generality, we consider a hierarchical edge cloud architecture, which consists of a BS-level data center (DC), an access-level DC, and an edge-level DC, with each data center equipped with a set of general-purpose cloud servers. Let $P$, $Q$, and $H$ represent the server sets in the above-mentioned data centers, respectively, and $L=P\cup Q\cup H$ represents the total server set. $(C_l,\ S_l)$ is used to capture the capacity of server $l \in L$, where $C_l$ and $S_l$ represent the computation and storage capacities of server $l$, respectively. Let $t_l$ represent the network propagation delay from the BS to server $l$. Note that we focus on the aggregated user demand (at the BS), which is further processed by the VM of different applications in edge clouds, rather than transmissions in the wireless part; thus, the wireless characters are not considered in this study. Based on this consideration, which is the same as that of Wang W et al. (2017), the transmission delay and network propagation delay between user equipments (UEs) and the BS are not taken into account in the latency-related constraints in the problem formulation. The parameters and variables used in this study are summarized in Table 1.

### 3.2 Deployment requirements of third parties

Let $A$ represent the application set where $a \in A$ represents an application that a certain third party wants to deploy in the edge cloud to serve its users in the BS. Specifically, the applications in $A$ have different deployment requirements (i.e., VM configuration, service capability, and latency assurance), so each application is implemented in a certain number of customized VMs instantiated on physical servers located at different cloud data centers owned by the MNO. In this study we classify the applications into two types, computation- and storage-intensive ones, whose deployment requirements are described as follows.

CIAs are applications that need more computation resources than storage resources (i.e., big data analytics and video transcoding). We use $A_{\mathrm{c}}$ $(A_{\mathrm{c}} \subseteq A)$ to represent a CIA. Let $(C_i,\ S_i)$ represent the resource requirements of one VM of application $i$ $(i \in A_{\mathrm{c}})$, where $C_i$ and $S_i$ are quantities of computation and storage resources needed by the VM, respectively. We assume that the request arrival rate of $i$ $(i \in A_{\mathrm{c}})$ at the BS is $r_i$. In addition, let $T_i$ represent the average service latency threshold of $i$ $(i \in A_{\mathrm{c}})$.

SIAs are applications that need more storage resources than computation resources. Note that some SIAs have to be implemented in the same data center with a certain CIA for efficiency or QoS requirements. For example, in some scenarios of intelligent surveillance applied in banks or transportation, a video storage application has to be implemented in a specific location, while a data analytics application (computation-intensive) is implemented in the same data center. By comparison, to deploy a video on-demand service, a video storage application could be implemented alone, without the requirement of other applications being in the same data center. Therefore, SIAs are classified into two types below.

We use $A_{\mathrm{s}_1}$ $(A_{\mathrm{s}_1} \subseteq A)$ to represent the SIA

**Table 1 Summary of parameters and variables used in this paper**

| Parameter | Description |
|---|---|
| $P,\ Q,\ H\ (L=P\cup Q\cup H)$ | Cloud server sets in the BS-level DC, access-level DC, and edge-level DC, respectively; $L$ is the total server set |
| $A_{\mathrm{c}},\ A_{\mathrm{s}_1},\ A_{\mathrm{s}_2}\ (A=A_{\mathrm{c}}\cup A_{\mathrm{s}_1}\cup A_{\mathrm{s}_2})$ | CIA and two types of SIA; $A$ is the total application set |
| $(C_a, S_a)$ | (Computation resource requirement, storage resource requirement) of a VM of $a\in A$ |
| $t_l$ | Network propagation delay from the BS to server $l$ |
| $D_l$ | Static cost of server $l$ ($l\in L$) when it is active |
| $D_l^a$ | Cost of deploying a VM of application $a$ at server $l$ |
| $u_a$ | Service capability of a VM of application $a$ |
| $\phi_{i,j}$ | Quantitative relation between the number of VMs of applications $i$ and $j$ |
| $r_i$ | Request arrival rate of $i$ ($i\in A_{\mathrm{c}}$) at the BS |
| $\Pi_k$ | Storage capacity that application $k$ ($k\in A_{\mathrm{s}_2}$) needs |
| $T_i$ | Service latency threshold of application $i$ ($i\in A_{\mathrm{c}}$) |
| $T_k$ | Latency threshold that a VM of application $k$ ($k\in A_{\mathrm{s}_2}$) should satisfy |
| **Variable** | **Description** |
| $n_{la}$ | The number of VMs of application $a$ deployed at server $l$ |
| $r_{il}$ | The amount of application $i$'s request load that is served by server $l$ |

DC: data center; CIA: computation-intensive application; SIA: storage-intensive application

where $j\in A_{\mathrm{s}_1}$ has to be implemented in the same data center with a certain application in $A_{\mathrm{c}}$. Let $(C_j,\ S_j)$ denote the resource requirements of a VM of application $j$, where $C_j$ and $S_j$ are the quantities of computation and storage resources needed by the VM, respectively. In addition, $\phi_{i,j}>0$ means that the number of VMs of application $j$ ($j\in A_{\mathrm{s}_1}$) is $\phi_{i,j}$ times that of $i$ ($i\in A_{\mathrm{c}}$) in the same data center; $\phi_{i,j}=0$ means that the numbers of VMs of applications $i$ and $j$ are not relative in any data center.

We use $A_{\mathrm{s}_2}$ ($A_{\mathrm{s}_2}\subseteq A$) to represent the SIA that could be implemented independently. $(C_k,\ S_k)$ is used to denote the resource requirements of one VM of application $k$ ($k\in A_{\mathrm{s}_2}$). $\Pi_k$ is the storage capacity that application $k$ ($k\in A_{\mathrm{s}_2}$) needs. Note that the service latency of SIAs is generally not a hard constraint in the cache allocation problem (Ghoreishi et al., 2016; Peng et al., 2016; Tong et al., 2016), so only a one-way delay of the location of a certain VM is given as the latency constraint for application $k$ in the later formulation of the optimal problem. Let $T_k$ represent the one-way delay of the location of a VM of $k\in A_{\mathrm{s}_2}$ that should be characterized.

### 3.3 Resource segmentation problem

Based on the edge cloud infrastructure of the MNO and the deployment requirements of third parties we described above, in this subsection we formulate the resource segmentation problem (RSP) to determine the deployment details of both the MNO and third parties.

Assume that the base cost of server $l$ ($l\in L$) is $D_l$ and $D_l^a$ is the cost of deploying a VM of application $a$ at server $l$. With the parameters in Table 1 as input, the RSP is formulated as in Eqs. (1a)–(1j). The objective is to minimize the total deployment cost of all applications, which influences both the resource efficiency of the MNO and the leasing prices of the third parties. As Eq. (1a) shows, the first part of the total cost is the summation of the static cost of total active servers, and the second part is the total deployment cost of all VMs at the relative active servers. $\|\boldsymbol{x}\|_0$ represents the $l_0$-norm of $\boldsymbol{x}$, which stands for the number of nonzero elements of $\boldsymbol{x}$.

Constraint (1b) means that the computation resources of all VMs cannot exceed the servers' computation capacity. Constraint (1c) means that the storage resources of all VMs cannot exceed the servers' storage capacity. Constraint (1d) means that all VMs that are allocated to application $i$ ($i\in A_{\mathrm{c}}$) should serve no less than the request load. Constraint (1e) guarantees that the sum of the queueing and processing latency at server $l$ and the network propagation delay from the BS to server $l$ cannot exceed the service latency of application $i$ ($i\in A_{\mathrm{c}}$). Constraints (1f)–(1h) guarantee the quantitative relation between the VMs of application $j$ ($j\in A_{\mathrm{s}_1}$) and the VMs of any application in $A_{\mathrm{c}}$ in $P$, $Q$, $H$,

respectively. Constraint (1i) guarantees that the VMs allocated to application $k$ ($k \in A_{\mathrm{s}_2}$) satisfy the capacity needs of the third party. Constraint (1j) guarantees that for any $k$ ($k \in A_{\mathrm{s}_2}$), its VM(s) should be deployed at the location from which the propagation delay to the BS is less than its latency requirement.

$$\min \quad \sum_{l\in L}\left\|\sum_{a\in A} n_{la}\right\|_0 D_l + \sum_{l\in L}\sum_{a\in A} n_{la} D_l^a \tag{1a}$$

s.t.

$$\sum_{a\in A} n_{la}C_a \le C_l,\ \forall l \in L, \tag{1b}$$

$$\sum_{a\in A} n_{la}S_a \le S_l,\ \forall l \in L, \tag{1c}$$

$$\sum_{l\in n_{li}} r_{il} = r_i,\ \forall i \in A_{\mathrm{c}}, \tag{1d}$$

$$\frac{1}{u_a - r_{il}/n_{la}} + t_l \le T_i,\ \forall i \in A_{\mathrm{c}},\ \forall l \in L, \tag{1e}$$

$$\sum_{p\in P} n_{pj} = \phi_{ij}\sum_{p\in P} n_{pi},\ \forall i \in A_{\mathrm{c}},\ \forall j \in A_{\mathrm{s}_1}, \tag{1f}$$

$$\sum_{q\in Q} n_{qj} = \phi_{ij}\sum_{q\in Q} n_{qi},\ \forall i \in A_{\mathrm{c}},\ \forall j \in A_{\mathrm{s}_1}, \tag{1g}$$

$$\sum_{h\in H} n_{hj} = \phi_{ij}\sum_{h\in H} n_{hi},\ \forall i \in A_{\mathrm{c}},\ \forall j \in A_{\mathrm{s}_1}, \tag{1h}$$

$$\sum_{l\in L} n_{lk} = \left\lceil \frac{\Pi_k}{S_k} \right\rceil,\ \forall k \in A_{\mathrm{s}_2}, \tag{1i}$$

$$(T_k - t_l)n_{la} \ge 0,\ \forall k \in A_{\mathrm{s}_2},\ \forall l \in L; \tag{1j}$$

var :

$$n_{la} \in \mathbb{N},\ \forall a \in A,\ \forall l \in L,$$

$$r_{il},\ \forall i \in A_{\mathrm{c}},\ \forall l \in L.$$

The RSP is a mixed-integer non-linear programming (MINLP) problem, which is NP-hard in general (Burer and Letchford, 2012). Furthermore, the non-smooth and non-convex $l_0$-norm in the objective function makes the RSP even more difficult to solve. Inspired by Chu and Beasley (1997), in the next section, we propose a genetic algorithm to solve the RSP.

## 4 An eGA-based algorithm to solve the resource segmentation problem

Generally speaking, a genetic algorithm is an intelligent algorithm which stimulates the process of evolution to search for the best offspring (solution) by applying genetic operators on the latest population in each reproduction (Tan et al., 2016). For the RSP, we demonstrate the specific genetic operators used in this study as follows.

The selection operator is used to select individuals in the current population for later crossover and mutation operations. We adopt the elitist selection strategy (referred to as $S(\cdot)$ in the algorithm) in this work. It works according to the following rules: (1) Use Eq. (1a) as the fitness function to calculate every individual's fitness, where the smaller the fitness is, the better the individual is. (2) Individuals that do not satisfy constraints Eqs. (1b)–(1j) should be rendered obsolete in every iteration (Michalewicz and Schoenauer, 1996). (3) The best individual (the most feasible individual with the smallest fitness) in the current population will be directly reserved in the next population, without being operated by crossover and mutation operators. (4) The probability of selecting individual $r$ is calculated as $p_r = \mathrm{fitness}_r / \sum_{i=1}^{S} \mathrm{fitness}_i$ ($S$ is the size of the population).

The crossover operator is used to generate new offspring with the individuals in the current population, without introducing new genes. We adopt arithmetic crossover in this work (referred to as $C(\cdot)$ in the algorithm), which generates the new offspring by linearly combining two parents.

The mutation operator is used to generate a new offspring by introducing a new piece of gene pool to an individual in the current population. We adopt non-uniform mutation in this work (referred to as $M(\cdot)$ in the algorithm), which generates a new offspring by introducing a stochastic disturbance to a parent.

Based on the genetic operators above, we design the eGA as Algorithm 1, which considers the population size $S$, the probability of mutation $P_{\mathrm{mu}}$, the maximum iteration number $I$, and the convergence threshold $\delta$ as input, and output the optimal solution of the RSP.

First, an initial population is generated (line 1). In the $i^{\mathrm{th}}$ iteration of the evolution, a new population is initialized as an empty set (line 3). Then the best individual of the last population is chosen as $u_{i-1}^*$ and assigned to $u^*$, which is used to record the best individual until now (lines 4 and 5). If the solution converges, the evolution terminates (lines 6

and 7). Otherwise, take $u^*_{i-1}$ as the first individual of $U_i$ (line 9) and generate the new offspring through selection, crossover, and mutation (lines 11–13), and the newly generated offspring $\tilde{u}_a, \tilde{u}_b$ are included in $U_i$. Lines 11–14 are executed repeatedly until there are $S$ individuals in $U_i$.

**Algorithm 1** An eGA-based algorithm to solve the RSP

1: Generate an initial population $U_0$ with $S$ feasible solutions of RSP
2: **for** $i \leftarrow 1$ to $I$ **do**
3: $U_i = \varnothing$
4: Evaluate the fitness and feasibility of solutions in $U_{i-1}$ and obtain the best individual $u^*_{i-1}$ ($u^*_{i-1} \in U_{i-1}$)
5: $u^* = u^*_{i-1}$
6: **if** $u^*$ has converged (determined according to $\delta$) **then**
7: Break
8: **end if**
9: $U_i(1) = u^*_{i-1}$
10: **while** $|U_i| < S$ **do**
11: $u_a, u_b \Leftarrow S(U_i)$ // selection
12: $u_a^{'}, u_b^{'} \Leftarrow C(u_a, u_b)$ // crossover
13: $\tilde{u}_a \Leftarrow M(u_a^{'}, P_{\mathrm{mu}})$, $\tilde{u}_b \Leftarrow M(u_b^{'}, P_{\mathrm{mu}})$ // mutation
14: $U_i \Leftarrow U_i \cup \{\tilde{u}_a, \tilde{u}_b\}$
15: **end while**
16: **end for**
17: Return $u^*$ as the optimal solution of RSP

## 5 Evaluation

In this section, we conduct simulations in Matlab R2017b to verify the feasibility and effectiveness of the proposed algorithm and analyze the numerical results.

### 5.1 Simulation scenario and setup

In the simulations, we consider a scenario where an MNO owns 4, 3, 3 available cloud servers in different hierarchies in its edge clouds, respectively. The server capacity and VM capacity are set according to the parameters in Tang et al. (2016). The delay performance (reflected by $t_l$) that the edge cloud based network can provide is set according to the end-to-end delay supported by the 5G edge cloud architecture of China Unicom (China Unicom, 2018). Moreover, there are 10 requests for deploying 10 different applications with different latency requirements, which are set according to Wang W et al. (2017). In summary, all parameters are listed in Table 2. In addition, considering that the VM cost (i.e., energy consumption) in the cloud data center is usually given as a linear approximation of the CPU load (usage of computation resources) (Mann, 2015), $D^a_l$ is set according to $D^a_l = 10\,000 C_a / C_l$.

**Table 2 Parameters used in the simulation**

| Parameter | Value |
|---|---|
| $\lvert L\rvert$ | 10 ($\lvert P\rvert = 4,\ \lvert Q\rvert = 3,\ \lvert H\rvert = 3$) |
| $\lvert A\rvert$ | 10 ($\lvert A_{\mathrm{c}}\rvert = 4,\ \lvert A_{\mathrm{s}_1}\rvert = 2,\ \lvert A_{\mathrm{s}_2}\rvert = 4$) |
| $(C_l,\ S_l)$ | (20 000, 100 000) |
| $D_l$ | 1000 |
| $t_l$ (ms) | 0.5 ($l \in P$), 9 ($l \in Q$), 18 ($l \in H$) |
| $(C_a,\ S_a)$ | (500,150), (520,100), (550,180), (585,175), (120,450), (135,435), (180,495), (155,475), (175,530), (190,585) |
| $r_i$ ($i \in A_{\mathrm{c}}$) | 450, 600, 300, 450 |
| $\phi$ | $\phi_{3,1} = \phi_{4,2} = 1$ |
| $\Pi_k$ ($k \in A_{\mathrm{s}_2}$) | 5400, 2500, 5400, 3600 |
| $T_i$ ($i \in A_{\mathrm{c}}$) (ms) | 1, 5, 20, 20 |
| $T_k$ ($k \in A_{\mathrm{s}_2}$) (ms) | 20, 25, 30, 50 |

The parameters of eGA are set as follows: $S = 500,\ I = 2000,\ P_{\mathrm{mu}} = 0.3,\ \delta = 10^{-2}$.

As discussed in Section 2, the related articles are different from ours in two aspects, namely, the optimized objective (including the granularity of the allocated resources) and the key observed metric. Take Wang W et al. (2017) as an example. This work considers only the edge deployment of CIAs. The total computation resources consumed are minimized and observed as a key metric (in evaluation, they specify the total computation units as the capacity of each edge data center). In comparison, we consider the deployment requirements of both CIAs and SIAs. In system model and evaluation, we give the configuration (the need for computation and storage resources) of one VM for every type of application. In addition, the capacity of every edge data center is given as the number of physical servers with the specific configuration. The observed metric is the total cost, which consists of the basic cost of the active physical servers and the accumulated cost of the VMs running in the active servers. The above discussion implies that it is inappropriate and unfair to conduct the comparisons.

For a better understanding of the resource efficiency that could be obtained, we demonstrate the simulation results under different situations when the load requirements and/or latency requirements of the third parties vary. We use the load ratio and the latency ratio defined in the following to describe the variation in the third parties' deployment

requirements.

Using the request arrival rate of applications in $A_c$ and the capacity requirements of applications in $A_{s_2}$ as the base request load, we observe the trend in the related metrics as the request load increases to $\alpha$ (called the load ratio in the rest of the paper) times the baseline request load.

Considering that latency is one of the most important parameters in an application deployment in mobile edge clouds, we use the latency ratio $\beta$ to represent the case in which the latency requirement of the related applications becomes $\beta$ $(0 < \beta < 1)$ times the baseline latency (defined as $\beta = 1$).

### 5.2 Performance metrics

First of all, we observe the running time of the implemented algorithm. Simulations are conducted in a machine with an Intel Core i3 processor and 6 GB RAM. For the baseline case, with both the load ratio and latency ratio equaling one (these two ratios have no influence on the time complexity of the algorithm), it takes 7.2144 s to obtain the optimum. We believe that the running time will be shorter if the algorithm is implemented in high-performance commercial systems.

Furthermore, to evaluate the performance of the resource segmentation algorithm, we observe the following metrics during the simulation: (1) Total cost: a smaller total cost brings a better resource segmentation scheme. (2) Total VM: this assists in the operation and further optimization of the MNO. (3) Quantity of VMs of different types in every data center: this indicates the detailed utilization of every data center. (4) Average computation/storage resource utilization: this indicates the average resource utilization of all the active servers.

### 5.3 Simulation results

#### 5.3.1 Feasibility and convergence of eGA

Fig. 3 shows the convergence of eGA in different cases where the load ratio, the number of servers, and the applications vary. We find that the algorithm always converges to the minimum fitness (cost) within 1000 iterations.

Fig. 4 shows the optimal $n_{la}$ when the load ratio is three and the latency ratio is one. The three axes are the sequence number of the servers in $L$, the sequence number of applications in $A$, and the optimal $n_{la}$ $(l \in L,\ a \in A)$. Generally speaking, the CIAs ($a_1$–$a_4$) are allocated with more VMs at the servers that are nearer to the BS (servers that have smaller sequence numbers), while the SIAs ($a_5$–$a_{10}$) are allocated with more VMs at the servers that are further from the BS (servers that have larger sequence numbers). In particular, we find that $n_{l1} = 0,\ \forall l \in [5, 10]$, which meets the latency requirement of $a_1$ (the latency threshold of $a_1$ is 1 ms, which is smaller than the network propagation delay of the servers in $P$ and $Q$); thus, VMs for $a_1$ could be deployed only in the BS-level DC that owns $l_1$–$l_4$. Similarly, $a_2$ could be deployed only in $l_1$–$l_7$, so $n_{l2} = 0,\ \forall l \in [8, 10]$.

In addition, it can be verified that the numerical relationships of $n_{l5}$ and $n_{l3}$ when $l \in [1, 4]$, $l \in [5, 7]$, $l \in [8, 10]$ satisfy constraints (1f)–(1h), respectively, and so on for $n_{l6}$ and $n_{l4}$.

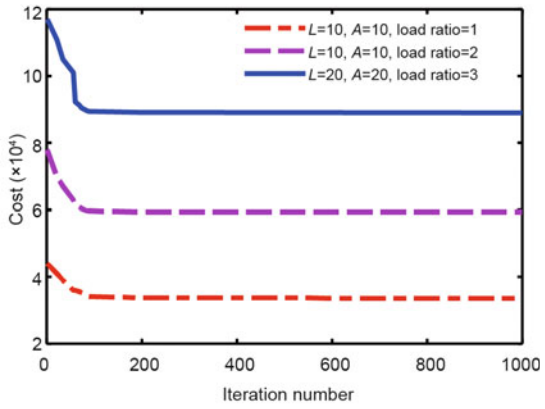


**Fig. 3 Convergence of the elitist genetic algorithm (eGA)**

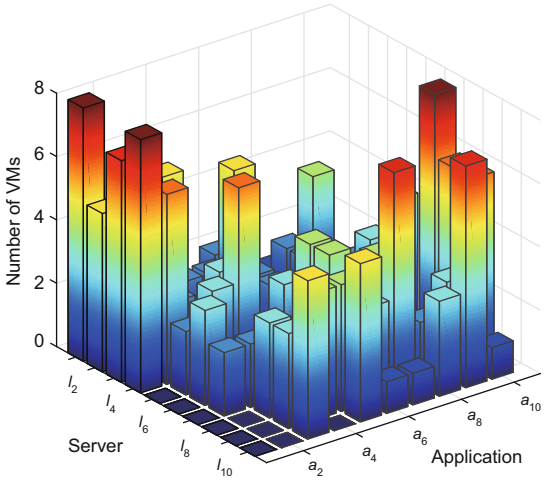


**Fig. 4 Optimal solution when the load ratio is three and latency is one**

5.3.2 Resource segmentation performance

Figs. 5 and 6 present the trend in total deployment cost and total number of VMs when the load ratio increases from 1 to 8 (the latency ratio is equal to 0.8, 0.4, or 0.2). As expected, more VMs are needed by the third parties to satisfy their increasing user demand, which makes the total deployment cost also increase (leading to more leasing payments). Specifically, when the load ratio varies from 1 to 3, the three curves in Fig. 5 increase at nearly the same rate. Nevertheless, as the load ratio continuously increases after reaching 3, the curve with $\beta = 0.2$ obviously increases faster than the other two curves. In Fig. 6, the curve with $\beta = 0.2$ always has a higher growth rate than the other two curves. The results indicate that if the deployed applications are latency-sensitive when the total demand is relatively high, there are fewer available resources that can be used for disaster recovery; thus, the MNO should provision backup resources at neighbor data centers and might consider raising the prices for third parties.

Furthermore, Figs. 7–9 present the total number of VMs for $A_c$, $A_{s_1}$, $A_{s_2}$ in $P$, $Q$, $H$, respectively, when the load ratio increases from 1 to 4. We observe that the number of VMs for every type of application increases when the load ratio increases. Specifically, the number of VMs for $A_c$ obviously increases more in $P$ and $Q$, which is reasonable because some applications in $A_c$ can be deployed only in $P$ and $Q$ due to their latency requirements. In addition, we find that the VMs of applications in $A_{s_1}$ and $A_{s_2}$ account for a higher proportion in $H$, which is reasonable because the latency requirements for the SIAs can always be satisfied by the edge-level DC.

Fig. 10 shows the total number of VMs when the latency ratio decreases from 1 to 0.1 with the load ratio equaling 1. We observe that as the latency requirements of applications become more rigorous, an increasing number of VMs need to be deployed. Specifically, when the latency ratio decreases from

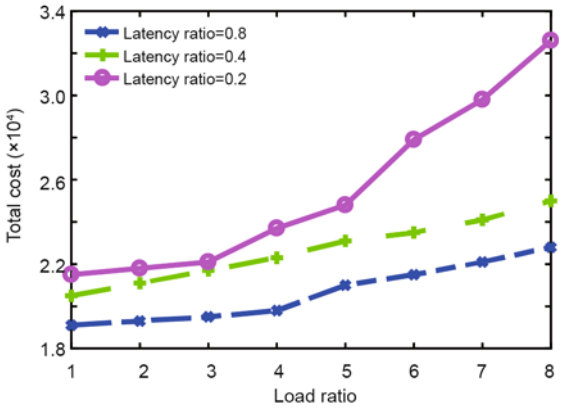


**Fig. 5 Total cost when the load ratio increases**

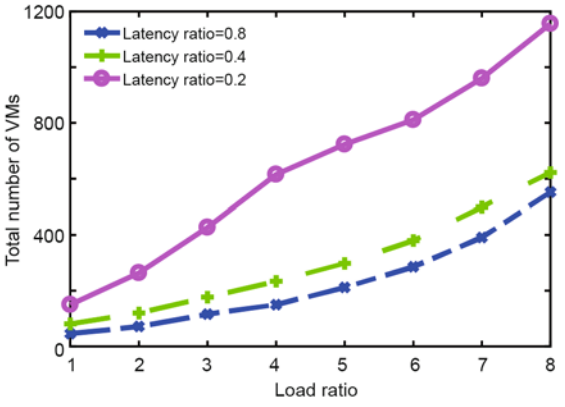


**Fig. 6 Total number of VMs when the load ratio increases**

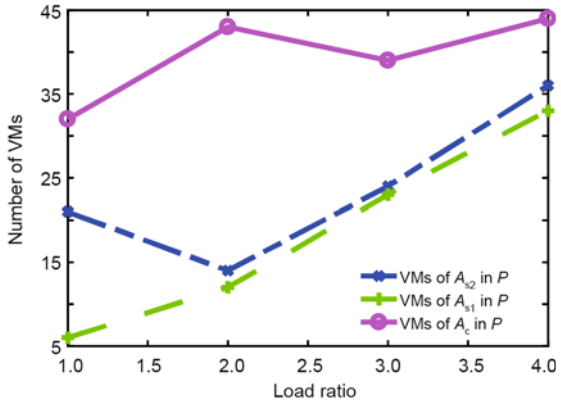


**Fig. 7 Number of VMs for each type of application in $P$ when the load ratio increases**

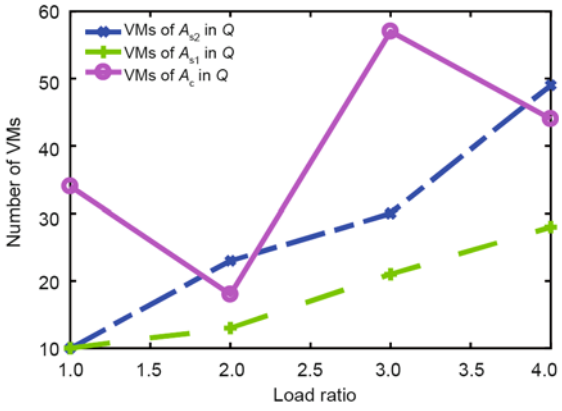


**Fig. 8 Number of VMs for each type of application in $Q$ when the load ratio increases**

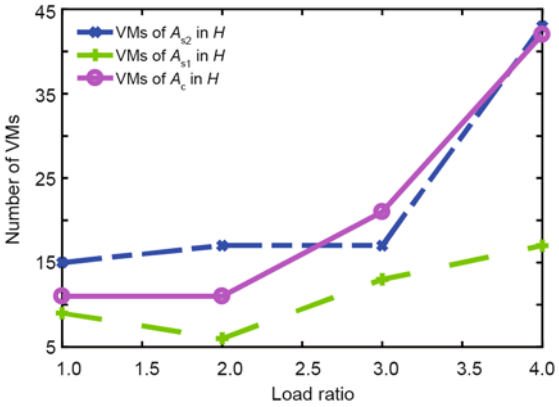


**Fig. 9 Number of VMs for each type of application in $H$ when the load ratio increases**

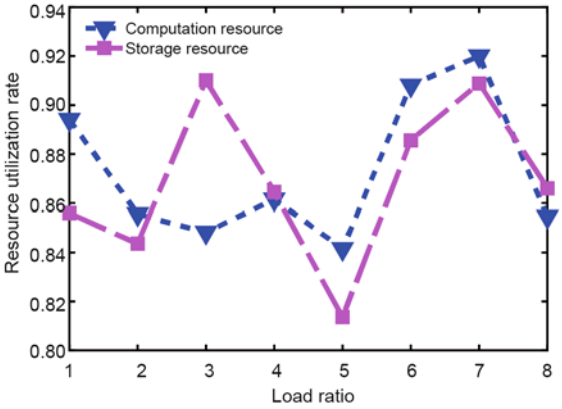


**Fig. 11 Resource utilization rate when the load ratio increases**

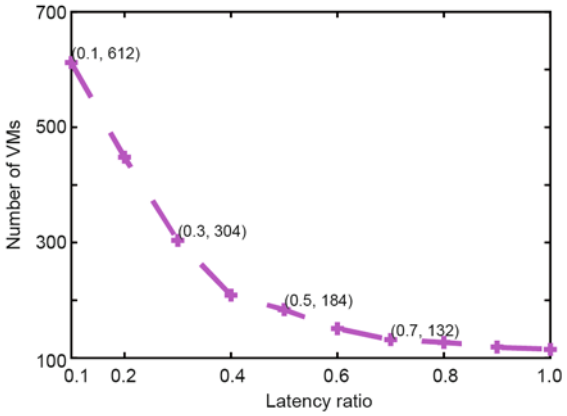


**Fig. 10 Number of VMs when the latency ratio decreases**

0.4 to 0.1, the growth rate for the needed VMs becomes extremely high. This result could be used by MNOs to apply different pricing plans to each type of VM according to its particular latency guarantee.

Fig. 11 shows the average computation/storage resource utilization of all the servers when the load ratio increases from 1 to 8. We find that the resource utilization can always be maintained at more than 80%, which proves that our eGA is cost-effective for resource segmentation. Note that the result here is the highest ratio that could be obtained by employing our algorithm. In practice, however, an MNO might set a threshold for the utilization ratio for various operational goals, e.g., energy savings.

### 5.4 Limitations and insights

Although problem formulation and simulation analysis have been conducted, there is still a lack of validation in practical deployments, which is essential and should be pursued in future work. Moreover, recall that for simplicity we considered the scenario where the user demand in an area covered by only one BS is satisfied. In practice, third parties would want to serve the user demand in different areas covered by different BSs. In that case we could modify the current formulation so as to adapt to these practical needs, that is, giving every application a specific cloud server set (i.e., use $L_a$ to replace $L$ in the formulation) to satisfy its requirement of covering different areas. Furthermore, the metrics tested in this work could be taken into account when planning for expansion or clustering of edge cloud resources (Bouet and Conan, 2018).

## 6 Conclusions

In this paper, we have addressed the problem of VM-level resource allocation, called "resource segmentation" in the context of 5G hierarchical clouds. Specifically, the various applications that third parties want to deploy have been classified into two types, computation- and storage-intensive ones, with each type of application having specific deployment requirements. Then we formulated the resource segmentation problem from the perspective of an MNO, taking the deployment requirements of all its third parties into consideration. An elitist genetic algorithm has been designed to solve the problem. Finally, the feasibility and effectiveness of the algorithm were evaluated under different levels of deployment requirements.

In future work, we plan to evaluate the proposed algorithm in our ongoing system implementation,

with practical deployment requirements and demands that cover multiple BSs.

## Compliance with ethics guidelines

Ming-shuang JIN, Shuai GAO, Hong-bin LUO, and Hong-ke ZHANG declare that they have no conflict of interest.

## References


Abbas N, Zhang Y, Taherkordi A, et al., 2018. Mobile edge computing: a survey. *IEEE Int Things J*, 5(1):450-465. https://doi.org/10.1109/JIOT.2017.2750180

Alicherry M, Lakshman TV, 2012. Network aware resource allocation in distributed clouds. Proc IEEE INFOCOM, p.963-971. https://doi.org/10.1109/INFCOM.2012.6195847

Baktir AC, Ozgovde A, Ersoy C, 2017. How can edge computing benefit from software-defined networking: a survey, use cases, and future directions. *IEEE Commun Surv Tutor*, 19(4):2359-2391. https://doi.org/10.1109/COMST.2017.2717482

Bilal K, Erbad A, 2017. Edge computing for interactive media and video streaming. Proc 2nd Int Conf on Fog and Mobile Edge Computing, p.68-73. https://doi.org/10.1109/FMEC.2017.7946410

Bouet M, Conan V, 2018. Mobile edge computing resources optimization: a geo-clustering approach. *IEEE Trans Netw Serv Manag*, 15(2):787-796. https://doi.org/10.1109/TNSM.2018.2816263

Burer S, Letchford AN, 2012. Non-convex mixed-integer nonlinear programming: a survey. *Surv Oper Res Manag Sci*, 17(2):97-106. https://doi.org/10.1016/j.sorms.2012.08.001

China Telecom, 2016. China Telecom CTNet2025 Network Architecture White Paper (in Chinese).

China Unicom, 2018. White Paper for China Unicom's Edge-Cloud Service Platform Architecture and Industrial Eco-system. China Unicom Network Technology Research Institute.

Chu PC, Beasley JE, 1997. A genetic algorithm for the generalised assignment problem. *Comput Oper Res*, 24(1): 17-23. https://doi.org/10.1016/S0305-0548(96)00032-9

Ding QH, Pang HT, Sun LF, 2017. SAM: cache space allocation in collaborative edge-caching network. Proc IEEE Int Conf on Communications, p.1-6. https://doi.org/10.1109/ICC.2017.7996701

Enayet A, Razzaque MA, Hassan MM, et al., 2018. A mobility-aware optimal resource allocation architecture for big data task execution on mobile cloud in smart cities. *IEEE Commun Mag*, 56(2):110-117. https://doi.org/10.1109/MCOM.2018.1700293

Ghoreishi SE, Friderikos V, Karamshuk D, et al., 2016. Provisioning cost-effective mobile video caching. Proc IEEE Int Conf on Communications, p.1-7. https://doi.org/10.1109/ICC.2016.7511549

Gudipati A, Perry D, Li LE, et al., 2013. SoftRAN: software defined radio access network. Proc 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, p.25-30. https://doi.org/10.1145/2491185.2491207

Lai ZQ, Hu YC, Cui Y, et al., 2017. Furion: engineering high-quality immersive virtual reality on today's mobile devices. Proc 23rd Annual Int Conf on Mobile Computing and Networking, p.409-421. https://doi.org/10.1145/3117811.3117815

Mach P, Becvar Z, 2017. Mobile edge computing: a survey on architecture and computation offloading. *IEEE Commun Surv Tutor*, 19(3):1628-1656. https://doi.org/10.1109/COMST.2017.2682318

Mann ZÁ, 2015. Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms. *ACM Comput Surv*, 48(1), Article 11. https://doi.org/10.1145/2797211

Manzalini A, Minerva R, Callegati F, et al., 2013. Clouds of virtual machines in edge networks. *IEEE Commun Mag*, 51(7):63-70. https://doi.org/10.1109/MCOM.2013.6553679

Mao YY, You CS, Zhang J, et al., 2017. A survey on mobile edge computing: the communication perspective. *IEEE Commun Surv Tutor*, 19(4):2322-2358. https://doi.org/10.1109/COMST.2017.2745201

MEC, 2016. Multi-access Edge Computing (MEC); Framework and Reference Architecture. ETSI GS MEC 003 v2.1.1. https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/02.01.01_60/gs_MEC003v020101p.pdf

MEC, 2018. Mobile Edge Computing (MEC); Deployment of Mobile Edge Computing in an NFV Environment. ETSI GR MEC 017 v1.1.1. https://www.etsi.org/deliver/etsi_gr/MEC/001_099/017/01.01.01_60/gr_MEC017v010101p.pdf

Michalewicz Z, Schoenauer M, 1996. Evolutionary algorithms for constrained parameter optimization problems. *Evol Comput*, 4(1):1-32. https://doi.org/10.1162/evco.1996.4.1.1

Peng X, Zhang J, Song SH, et al., 2016. Cache size allocation in backhaul limited wireless networks. Proc IEEE Int Conf on Communications, p.1-6. https://doi.org/10.1109/ICC.2016.7511288

Roman R, Lopez J, Mambo M, 2018. Mobile edge computing, Fog et al.: a survey and analysis of security threats and challenges. *Fut Gener Comput Syst*, 78:680-698. https://doi.org/10.1016/j.future.2016.11.009

Sama MR, Contreras LM, Kaippallimalil J, et al., 2015. Software-defined control of the virtualized mobile packet core. *IEEE Commun Mag*, 53(2):107-115. https://doi.org/10.1109/MCOM.2015.7045398

Sousa B, Cordeiro L, Simões P, et al., 2016. Toward a fully cloudified mobile network infrastructure. *IEEE Trans Netw Serv Manag*, 13(3):547-563. https://doi.org/10.1109/TNSM.2016.2598354

Taleb T, Dutta S, Ksentini A, et al., 2017. Mobile edge computing potential in making cities smarter. *IEEE Commun Mag*, 55(3):38-43. https://doi.org/10.1109/MCOM.2017.1600249CM

Tan LZ, Tan YY, Yun GX, et al., 2016. Genetic algorithms based on clustering for traveling salesman problems. Proc 12th Int Conf on Natural Computation, Fuzzy Systems and Knowledge Discovery, p.103-108. https://doi.org/10.1109/FSKD.2016.7603158

Tang JH, Quek TQS, Tay WP, 2016. Joint resource segmentation and transmission rate adaptation in Cloud RAN with Caching as a Service. Proc IEEE 17th Int Workshop on Signal Processing Advances in Wireless Communications, p.1-6. https://doi.org/10.1109/SPAWC.2016.7536886

Tong L, Li Y, Gao W, 2016. A hierarchical edge cloud architecture for mobile computing. Proc 35th Annual IEEE Int Conf on Computer Communications, p.1-9. https://doi.org/10.1109/INFOCOM.2016.7524340

Wang S, Zhang X, Zhang Y, et al., 2017. A survey on mobile edge networks: convergence of computing, caching and communications. *IEEE Access*, 5:6757-6779. https://doi.org/10.1109/ACCESS.2017.2685434

Wang SQ, Zafer M, Leung KK, 2017. Online placement of multi-component applications in edge computing environments. *IEEE Access*, 5:2514-2533. https://doi.org/10.1109/ACCESS.2017.2665971

Wang W, Zhao YL, Tornatore M, et al., 2017. Virtual machine placement and workload assignment for mobile edge computing. Proc IEEE 6th Int Conf on Cloud Networking, p.1-6. https://doi.org/10.1109/CloudNet.2017.8071527

Wang XF, Chen M, Taleb T, et al., 2014. Cache in the air: exploiting content caching and delivery techniques for 5G systems. *IEEE Commun Mag*, 52(2):131-139. https://doi.org/10.1109/MCOM.2014.6736753

Yin H, Zhang X, Liu HH, et al., 2017. Edge provisioning with flexible server placement. *IEEE Trans Parall Distrib Syst*, 28(4):1031-1045. https://doi.org/10.1109/TPDS.2016.2604803

Zhang HK, Quan W, Chao HC, et al., 2016. Smart identifier network: a collaborative architecture for the future Internet. *IEEE Netw*, 30(3):46-51. https://doi.org/10.1109/MNET.2016.7474343

Zhang S, Zhang N, Yang P, et al., 2017. Cost-effective cache deployment in mobile heterogeneous networks. *IEEE Trans Veh Technol*, 66(12):11264-11276. https://doi.org/10.1109/TVT.2017.2724547

Zhou H, Wang H, Li XH, et al., 2018. A survey on mobile data offloading technologies. *IEEE Access*, 6:5101-5111. https://doi.org/10.1109/ACCESS.2018.2799546