Price-aware Service Deployment in Hierarchical Mobile Edge Computing

Jie Huang, Ao Zhou, *Member, IEEE,* and Shangguang Wang, *Senior Member, IEEE*

***Abstract*—Mobile edge computing is considered as a promising solution to release pressure on the core network and reduce service response time. Edge nodes with storage and computation resources are able to cache various services and process tasks rather than offloading to remote clouds. However, it is difficult to make service deployment decisions appropriately since resources are limited in edge nodes and requirements of services are diverse. The hierarchical mobile edge computing structure in 5G network causes extra complication, and the cost of service deployment aggravates the hardness, especially for service providers. In this paper, we focus on the service deployment problem considering service caching, resource allocation, and task scheduling in the hierarchical mobile edge computing network, aiming at minimizing monetary cost. To address the heterogeneous limitations and requirements, we formulate the problem as mixed integer non-linear programming problem and develop an iterative service deployment algorithm by exploiting Gibbs sampling. We make the service caching strategies of edge nodes iteratively. Furthermore, we transform the resource allocation and task scheduling optimization into linear programming problem and employ a typical optimization function. Simulation results show that our algorithm always obtains minimal monetary cost for various number of services and task arrival rates, compared with benchmarks.**

***Index Terms*—Mobile Edge Computing, resource allocation, service caching, service deployment, task scheduling.**

I. Introduction

THE explosive growth of mobile devices and networking technologies greatly promote network cloudification [1], and expedite generation of various mobile applications such as IoT sensor monitoring and mobile games. According to the prediction of Cisco [2], 299.1 billion mobile applications will be downloaded globally by 2023. It is a critical problem that how to deploy these application services with different latency and resource requirements appropriately. Cloud computing provides an idea of storing these services in cloud data center and offloading tasks through the core network. However, the wide area network between central cloud and mobile devices may lead to uncontrolled latency, which is not friendly for latency-sensitive services like augmented reality. Mobile edge computing (MEC) is a convincing solution which brings high-performance power of cloud to network edge [3], [4]. Tasks are able to be processed at edge nodes, which substantially releases the pressure of offloading traffic through the core network and reduce service response time.

Compared with cloud which has elastic resource capacity, resources in the edge node are limited. Therefore, the proper caching of services and resource allocation in MEC is extremely crucial, since an edge node is not able to cache all services. In addition, latency requirements on services are different. For example, services such as virtual reality prefers shorter latency, and constraint is relatively loose of video services. Hence tasks of various services can be processed at different layer in the hierarchical MEC framework [5], [6], which makes task scheduling more important.

The service deployment problem in MEC has attracted attention of many researchers. Poularakis *et al.* jointly optimize service placement and request routing [7], which maximize the number of requests served by edge nodes. However, the request that offloaded to cloud may exceed the latency constraint and the user QoS is not able to be guaranteed. Except that, the heterogeneities of edge nodes or services are simplified, rather than describe the differences on resources or requirements in the existing works [8]–[10]. More importantly, cost (specifically, the monetary cost) is practical for service providers in service deployment, and is under-appreciated [11]–[14]. In fact, we should not only solve the complications caused by varieties of edge nodes and services, but also reduce the service deployment cost.

In this paper, we solve the service deployment problem by answering three questions: which services should be cached in each edge node, how many resources will be used, and how to schedule tasks of corresponding services. Firstly, we cache services (including the application data and related databases) not only consider diversities of edge nodes, but also meet distinct requirements of services. Secondly, given the service caching strategies, proper computation resources should be allocated to match the occupied storage resources. Thirdly, computation tasks should be scheduled correctly among the edge nodes that have cached the corresponding services. Finally, we greatly reduce service deployment cost by taking full use of resources in each edge node.

Challenges for the problem can be summarized into two folds: coupled problem, and heterogeneities of edge nodes and services. Firstly, service caching, resource allocation, and task scheduling are coupled. To be specific, service caching should be supported by resource allocation, or the strategy is meaningless. And resource allocation without caching service induce waste. Besides, though caching strategies restrain the range of task scheduling, the scheduling indicates performance of service caching. The amount of allocated resources de-

J. Huang, A. Zhou, and S. Wang are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China (e-mail: {huangjie, aozhou, sgwang}@bupt.edu.cn).

termines the number of scheduled tasks, and in return, task scheduling results reflect the resource requirement. Considering interactions between the three aggravates challenges of solving the problem. Secondly, each kind of resources and price of edge nodes are diverse, requirements of resources and latency of services are distinct, which makes resource allocation and task scheduling more difficult. As shown in Fig. 1, different categories and number of servers indicates the resources of edge node are various, and hierarchical structure complicates the difference. Communication latency increases with layer since the transmission distance from base stations to edge nodes grows. In view of each service has individual requirement on resources and latency, the balance between resource usage and latency causes complexity. Servers at lower-layer are closer to users, leads to higher expense as they are placed on densely-populated area with expensive rent. For reducing cost, the trade-off between resource usage among edge nodes makes the problem even difficult.

In this paper, we investigate the service deployment problem with the purpose of minimizing monetary cost while meeting service latency and resource requirements. We further propose an iterative service deployment algorithm by exploiting Gibbs sampling, which decouple and simplify the problem. We update service caching strategies of edge nodes iteratively, and jointly optimize resource allocation and task scheduling with the results of service caching. To make full use of limited resources, we consider storage and computation resource allocation respectively. In particular, we divide computation tasks and schedule different portion to diverse edge nodes for improving resource efficiency. We use CPU frequency to represent the computation resource of each edge node, and compensate lager transmission latency by allocating more computation resource of a remote edge node. In other words, the remote edge node with greater resources is able to reduce computation latency and meet the latency requirement of the corresponding service. Therefore, we reduce monetary cost through weighing resource allocation and task scheduling.

Our contributions are summarized as follows:

- We investigate service deployment in hierarchical MEC network, aiming at minimizing monetary cost, within different service latency constraints. We depict the computation latency of each edge node by using queuing theory, and make load balance among edge nodes for reducing service deployment cost. Furthermore, we formulate the problem as mixed integer non-linear program problem and decouple it to release the complexity.
- We propose an iterative algorithm and give the caching strategies under various resource limitation and different latency requirements for services. Besides, we transform the optimal resource allocation and task scheduling subproblem into linear programming problem and take advantage of a typical optimization function.
- We compare our algorithm with three benchmarks by a variety number of services and task arrival rates. Simulation results demonstrate the effectiveness of the proposed algorithm and the superior performance on reducing service deployment cost.

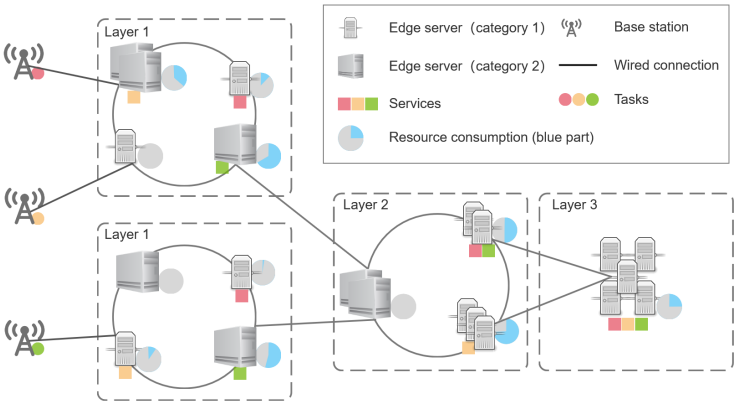


Fig. 1: Price-aware service deployment structure in hierarchical MEC network.

The rest of the paper is organized as follows. Section II reviews the related work. Section III describes the system model and defines the problem formally. And we develop the algorithms in Section IV. Section V performs simulations, followed by the conclusion in Section VI.

II. Related Work

Mobile edge computing has been anticipated as a promising solution to reduce service response time and release stress on massive offloading traffic of the core network. However, resource limitation in edge nodes is the inherent deficiency which confines the performance. There has been extensive studies on service caching or service deployment, to optimize QoS and improve resource utilization.

It is an effective approach to cache contents or services in edge nodes previously. In order to maximize cache hitting rate, Feng *et al.* [15] propose a content caching framework with multiple cloudlets at the edge of network. Zhang *et al.* [16] further take popular contents into account, for minimizing average number of hops across caching tiers and maximizing the overall cache hitting rate within the same tier. To process computation tasks, Xu *et al.* [17] jointly optimize dynamic service caching and task offloading. Besides, deadline-aware task scheduling maximizes the number of deadlines and is benefit for latency-sensitive tasks [18], [19]. Additionally, some researchers provide the idea of edge cooperation [20], [21], for maximizing the amount of tasks which processed at edge nodes. Nevertheless, on the one hand, maximization on cache hitting rate or tasks means that not all requests from users will be responded within service latency constraint. It can hardly guarantee user experience in practical. On the other hand, the limited resources in edge nodes are not able to be fully utilized without reasonable allocation.

Further, a number of researchers focus on joint task offloading and resource allocation. In particular, Yang *et al.* [22] consider storage and computation resource usage, as well as computation cost as for placing services and dispatch requests. The work [23] minimizes monetary cost for application service providers, and Pasteris *et al.* [24] try to maximize the total reward when placing multiple services in a heterogeneous MEC system. Zhou *et al.* [25] solve the problem of dynamic

service deployment expect for minimal response latency under a given budget constraint. Additionally, Ouyang *et al.* [26] propose a novel adaptive service placement mechanism which jointly optimizes latency and minimize total cost by the online algorithm without future information. Different from the existing work, we regard each edge node as an unique in terms of capacities and price of resources. We assume that storage and computation resources are diverse between edge nodes, and price is varying on each kind of resources. Moreover, the resources consumption, latency requirements and tasks of each service are individual.

In brief, we study service deployment by jointly consider service caching strategy, resource allocation, and task scheduling. We simplify the complex correlations in the coupled problem without losing heterogeneities of edge nodes and services. We reduce total monetary cost with reasonable resource allocation and avoid wasting. Except that, we meet service latency constraint by scheduling tasks to the valid destination. The user experience is guaranteed since each task can be processed within latency requirement.

III. System Model and Problem Formulation

In this section, we present the hierarchical MEC network, diverse services, and latency. We provide the model in detail and formulate the service deployment problem.

A. Hierarchical Mobile Edge Computing Network

We consider the hierarchical MEC consists a series of base stations and edge nodes. The lowest part of the network is a set of $\mathcal{B} = \{1, 2, ..., B\}$ base stations (on the far left in Fig. 1). They are wired connected to edge nodes and have no computation capability. Each base station $j \in \mathcal{B}$ receives user requests for all kinds of services and transmits corresponding tasks to the appropriate edge node.

In addition, the other layers of the network are composed of different number of edge nodes respectively. The partition of different layer in the network can be comprehended by the covered area. The number of edge nodes decreases relatively since nodes at upper layer serve larger coverage. It is worth noting that an edge node stands for a data center consists of servers. Accordingly, capacity implies sum of available resources in servers which form the edge node.

Specifically, suppose that there is a list of $\mathcal{E} = \{1, 2, ..., E\}$ edge nodes, and each edge node $k \in \mathcal{E}$ has a maximal storage capacity A_k, which is used to store candidate services. Assuming that price of all available storage resources A_k in edge node *k* is P_k^{a}, then cost caused by storage resource consumption can be calculated proportionally. Similarly, each edge node *k* has a maximal computation capacity F_k, denoted by CPU frequency (cycles per second). Suppose price of F_k is P_k^{f}. The cost of computation resource would be calculated proportionally as well.

B. Service

We will deploy multiple services of service provider in the same time. Formally, there is a set of $\mathcal{N} = \{1, 2, ..., N\}$ services ought to be deployed on *E* edge nodes simultaneously. In addition, each service can be replicated and cached in a group of edge nodes, but the same service is exclusive in a certain edge node. We do not impose restrictions on the number of duplications of the service, that is, we are allowed to cache the service on each edge node if feasible.

Let α_i express storage requirement of each service *i*. Furthermore, $x_k^i \in \{0, 1\}$ is a binary decision variable to show whether service *i* is cached in edge node *k* or not. For ensuring any kind of task can be responded in the network, each service should be cached at least once:

$$\sum_{k \in \mathcal{E}} x_k^i \geq 1, \quad i \in \mathcal{N}. \tag{1}$$

Let $\boldsymbol{x}_k = \{x_k^i \mid i \in \mathcal{N}\}$ denote service caching decision of an edge node *k*, and $\boldsymbol{X}_k$ is the action space.

As mentioned before, the amount of various resources in each edge node increases layer by layer, since lower-layer means closer to users and it is more expensive to manage servers. Thus MEC provider is unwilling to place lots of servers in densely-populated area. The fewer number of servers not only causes less available resources, but also leads to higher price of resources in edge nodes at lower-layer. Nonetheless, the adjustment between resource usage and task scheduling leads to compensation for the difficulty. As shown in Fig. 1, different kinds of tasks (shown as red, yellow, and green circles) can be processed in edge node at Layer 1 for taking advantage of shorter communication latency. We can also schedule tasks to edge node at Layer 2 or even Layer 3 with longer communication latency yet more computation resources. Thus the reduction on computation latency is able to meet total response time constraint, and also may bring a lower monetary cost. Let $y_{jk}^i \in [0, 1]$ represent the proportion of tasks transmit to edge node *k* among the overall requests for service *i* in a base station *j*. All the tasks should be processed:

$$\sum_{k \in \mathcal{E}} y_{jk}^i = 1, \quad i \in \mathcal{N}, j \in \mathcal{B}. \tag{2}$$

Moreover, the arrived tasks for service *i* is a Poisson process with expected rate λ_j^i at base station *j*, which is a general assumption [17]. Note that tasks received by edge nodes are all from base stations, then the arrived tasks for service *i* at edge node *k* is also Poisson process [27]. In other words, the arrival of tasks for service *i* at edge node *k* is subject to Poisson process with λ_k^i, which calculated as:

$$\lambda_k^i = \sum_{j \in \mathcal{B}} y_{jk}^i \cdot \lambda_j^i. \tag{3}$$

Besides, assume that computation task requirement (denoted by CPU cycles) for service *i* follows an exponential distribution with expectation μ_i. Similarly, it can be deduced that task processing time of service *i* in edge node *k* follows exponential distribution with expectation μ_i / z_k^i , where z_k^i is the amount of allocated computation resources to service *i*. It deserves to be explained that resources are actually shared by all the services cached in the same edge node, thus computation resources have to be separated to different services. Let $0 \leq z_k^i \leq F_k$ confine the allocated computation resources of service *i* could

not beyond maximal computation resources in edge node k, and $z_k^i = 0$ while $x_k^i = 0$. Intuitively, $\sum_{i\in\mathcal{N}} z_k^i$ draws the entire consumed computation resouces of cached services in edge node k, and $0 \leq \sum_{i\in\mathcal{N}} z_k^i \leq F_k$ imposes the computation capacity restriction (we use the gray circle in Fig. 1 to denote available computation resources and blue part shows the consumption in practice). It is possible that some edge nodes are not used (shown as the gray circle without blue part). Therefore, $\sum_{i\in\mathcal{N}} z_k^i = 0$ means no piece of resource is occupied in edge node k, and also means we do not cache any service on it. Consequently, remove extra overlapped constraints, we know that:

$$\begin{cases} 0 \leq z_k^i, \\ \sum_{i\in\mathcal{N}} z_k^i \leq F_k, & i \in \mathcal{N}, k \in \mathcal{E}. \end{cases} \tag{4}$$

C. Latency

Service response time is absolutely vital in service deployment problem. Tasks without meeting latency requirement not only damage quality of user experience, but also possibly incur irretrievable mistake in some case. For instance, objective recognition task of autonomous driving service should definitely obey the latency constraint, or it may trigger accidence. Accordingly, except for different resource requirements, each service has different latency requirement. Let Φ_i denote the tolerant upper bound latency of service i, that is, response time of tasks for service i is not allowed to exceed Φ_i.

The latency we described in this model consists of two parts: communication latency and computation latency. Some researchers present different methods to describe wireless latency between users and base stations [28]–[31]. Based on these presentation, and in view of statistical information of coverage, the communication latency in this paper is calculated from base stations. Therefore, the latency only refers to total time between a base station transmits user requests and computation task completed. Connections among edge nodes are wired, which means congestion of data transmission is not an obstruction. We assume that the state of network is stable in the given time interval. Let T_{jk} be the communication latency from initial base station j to target edge node k and communication latency can be calculated previously.

We use queuing method for depicting computation latency, and it mainly includes two parts: queuing and processing. According to the M/M/1 queue, the computation latency of tasks to service i which processed in edge node k is:

$$D_k^i = \frac{1}{\frac{z_k^i}{\mu_i} - \lambda_k^i}, \qquad x_k^i = 1, \tag{5}$$

where $x_k^i = 1$ states that D_k^i is valid only if service i has already been cached in edge node k. Otherwise, we evaluate D_k^i with a very large number to declare computation latency is unacceptable long once service i is non-existent in edge node k. This constraint can vastly eliminate meaningless task scheduling as well.

For ensuring the stability of the queue, or to avoid infinite queue length, there should be:

$$\frac{z_k^i}{\mu_i} > \lambda_k^i, \quad i \in \mathcal{N}, k \in \mathcal{E}. \tag{6}$$

In brief, the total response time L_{jk}^i for tasks processed in edge node k is sum of computation latency of corresponding service i and transmission latency from base station j. Specifically, we have:

$$L_{jk}^i = D_k^i + T_{jk}. \tag{7}$$

D. Problem Formulation

This paper studies service deployment including service caching, resource allocation and task scheduling, aiming at minimizing monetary cost in hierarchical MEC. In summary, the price-aware service deployment problem is formulated as:

$$\mathbf{P1}: \min_{x_k^i, z_k^i} \sum_{k\in\mathcal{E}} \sum_{i\in\mathcal{N}} x_k^i \cdot \left(\frac{\alpha_i}{A_k} \cdot P_k^{\mathrm{a}} + \frac{z_k^i}{F_k} \cdot P_k^{\mathrm{f}}\right)$$

$$\begin{aligned}
&\textit{s.t.} && \sum_{k\in\mathcal{E}} y_{jk}^i = 1, && i \in \mathcal{N}, j \in \mathcal{B} \\
&\mathrm{C1}: && \sum_{i\in\mathcal{N}} z_k^i \leq F_k, && k \in \mathcal{E} \\
&\mathrm{C2}: && \frac{z_k^i}{\mu_i} > \lambda_k^i, && i \in \mathcal{N}, k \in \mathcal{E} \\
&\mathrm{C3}: && L_{jk}^i \leq \Phi_i, && i \in \mathcal{N}, j \in \mathcal{B}, k \in \mathcal{E} \\
&\mathrm{C4}: && \sum_{i\in\mathcal{N}} x_k^i \cdot \alpha_i \leq A_k, && k \in \mathcal{E} \\
&\mathrm{C5}: && z_k^i \leq x_k^i \cdot F_k, && i \in \mathcal{N}, k \in \mathcal{E} \\
&\mathrm{C6}: && 0 \leq y_{jk}^i \leq x_k^i, && i \in \mathcal{N}, j \in \mathcal{B}, k \in \mathcal{E} \\
&\mathrm{C7}: && x_k^i \in \{0, 1\}, && i \in \mathcal{N}, k \in \mathcal{E} \\
&\mathrm{C8}: && y_{jk}^i \in [0, 1]. && i \in \mathcal{N}, j \in \mathcal{B}, k \in \mathcal{E}
\end{aligned}$$

Here C3 means no matter which edge node is chosen for processing tasks, the total latency is unable to exceed constraint of corresponding service. C4 ensures storage resource usage in each edge node could not beyond the maximal capacity. Constraint C5 shows that computation resource allocation to a service i in edge node k makes sense only if the service has already been cached. Similarly, C6 assures that only edge nodes equipped with service i are capable to receive corresponding tasks.

It could be deduced that for any service i, there exists no less than one edge node k makes $x_k^i = 1$. Otherwise, the computation latency D_k^i is always infinite and never meet constraint C3. It ensures that every service i will be cached at least once, so that (1) is able to be eliminated. Additionally, constraint C4 can be rewrote as $z_k^i > \mu_i \cdot \lambda_k^i$ since $\mu_i > 0$ is valid for any service i. Note that $\lambda_k^i \geq 0$, then $\mu_i \cdot \lambda_k^i \geq 0$. Therefore, the inequation $z_k^i \geq 0$ in (4) is can be removed.

IV. Algorithm

In this section, we describe our proposed algorithm which jointly optimizes service caching strategy of edge nodes, computation resource allocation, and task scheduling. We

formulate service deployment as mixed integer non-linear programming problem which is NP-hard and difficult to solve directly. Therefore, we further design an Iterative Service Deployment Algorithm by exploiting Gibbs sampling. Specifically, we update service caching strategy iteratively, then **P1** is reduced to the resource allocation and task scheduling subproblem. We further transform the subproblem into linear programming, and solve it by a typical optimization function.

A. Iterative Service Deployment Algorithm

Gibbs sampling is one of the Markov Chain Monte Carlo methods which applies for random variables with at least two dimensions. It can be concluded that the sampling distribution converges to the joint distribution by the properties of Markov chain and transition probability matrix. In brief, we can obtain objective joint distribution by Gibbs sampling method from conditional distributions [32]. Gibbs sampling is a heuristic algorithm which changes value of one individual variable in an iteration while keeping the rest variables unchanged. After sweeping each variable, it simulates conditional distribution samples to infer the joint distribution of all variables. As shown in Algorithm 1, we find out optimal service caching strategy with the objective value of **P1** (Step 7) by exploiting the idea of Gibbs sampling. To be specific, we unite the conditional probability distribution of service caching strategy, then the inferred joint distribution converges to the optimal results with high probability.

In each iteration, we select an edge node k and a valid service caching strategy $\boldsymbol{x}_k$ randomly while keeping choices of rest edge nodes unchanged. Therefore, problem **P1** will be reduced to the resource allocation and task scheduling subproblem with the given strategies of all edge nodes:

$$
\begin{aligned}
\mathbf{P2}: \min_{z_k^i} \ & \sum_{k\in\mathcal{E}}\sum_{i\in\mathcal{N}} x_k^i \cdot \left(\frac{\alpha_i}{A_k}\cdot P_k^{\mathrm{a}} + \frac{z_k^i}{F_k}\cdot P_k^{\mathrm{f}}\right) \\
\text{s.t.} \ & \sum_{k\in\mathcal{E}} y_{jk}^i = 1, && i\in\mathcal{N}, j\in\mathcal{B} \\
& \sum_{i\in\mathcal{N}} z_k^i \le F_k, && k\in\mathcal{E} \\
& \frac{z_k^i}{\mu_i} > \lambda_k^i, && i\in\mathcal{N}, k\in\mathcal{E} \\
& L_{jk}^i \le \Phi_i, && i\in\mathcal{N}, j\in\mathcal{B}, k\in\mathcal{E} \\
& z_k^i \le x_k^i \cdot F_k, && i\in\mathcal{N}, k\in\mathcal{E} \\
& 0 \le y_{jk}^i \le x_k^i, && i\in\mathcal{N}, j\in\mathcal{B}, k\in\mathcal{E} \\
& y_{jk}^i \in [0,1]. && i\in\mathcal{N}, j\in\mathcal{B}, k\in\mathcal{E}
\end{aligned}
$$

With results of problem **P2**, we obtain the optimal objective value g defined as $g = \min\limits_{z_k^i} \sum\limits_{k\in\mathcal{E}}\sum\limits_{i\in\mathcal{N}} x_k^i\cdot\left(\frac{\alpha_i}{A_k}\cdot P_k^{\mathrm{a}} + \frac{z_k^i}{F_k}\cdot P_k^{\mathrm{f}}\right)$. The g changes to g^* when service caching strategy of edge node k varies from $\boldsymbol{x}_k$ to $\boldsymbol{x}_k^*$. We further have $\rho = \frac{1}{1+e^{(g^*-g)/\omega}}$ (where ω is a smooth parameter and $\omega > 0$) represents the probability of the change from $\boldsymbol{x}_k$ to $\boldsymbol{x}_k^*$. Hence edge node k remains $\boldsymbol{x}_k$ unchanged with probability $1-\rho$. The iteration ends with the stop criteria is satisfied.

Algorithm 1 Iterative Service Deployment Algorithm

Input:

A_k, P_k^{a}, F_k, P_k^{f}, α_i, Φ_i, λ_j^i, μ_i $(i\in\mathcal{N}, j\in\mathcal{B}, k\in\mathcal{E})$

Output:

The optimal service caching strategy $\boldsymbol{X}$, computation resource allocation $\boldsymbol{Z}$ and task scheduling $\boldsymbol{Y}$.

1: Initialize $\boldsymbol{X}^0$.
2: **for** iteration r = 1, 2, ... **do**
3: Randomly select an edge node $k\in\mathcal{E}$ and a service caching strategy $\boldsymbol{x}_k^*\in\boldsymbol{X}_k$.
4: **if** $\boldsymbol{x}_k^*$ is feasible **then**
5: Find out the computation resource allocation $\boldsymbol{Z}$ and task scheduling $\boldsymbol{Y}$ in **P2**, as well as the corresponding optimal objective value g, based on the service caching strategies $(\boldsymbol{x}_1^{r-1},...,\boldsymbol{x}_k^{r-1},...,\boldsymbol{x}_E^{r-1})$.
6: Find out the computation resource allocation $\boldsymbol{Z}^*$ and task scheduling $\boldsymbol{Y}^*$ in **P2**, as well as the corresponding optimal objective value g^*, based on the service caching strategies $(\boldsymbol{x}_1^{r-1},...,\boldsymbol{x}_k^*,...,\boldsymbol{x}_E^{r-1})$.
7: Let $\boldsymbol{x}_k^r = \boldsymbol{x}_k^*$ with the probability $\rho = \frac{1}{1+e^{(g^*-g)/\omega}}$, and remain $\boldsymbol{x}_k^r = \boldsymbol{x}_k^{r-1}$ with the probability $1-\rho$.
8: **end if**
9: **if** the stopping criteria is satisfied **then**
10: End the iteration and return $\boldsymbol{X}^r$, $\boldsymbol{Z}^r$, $\boldsymbol{Y}^r$.
11: **end if**
12: **end for**

Theorem 1: *The proposed algorithm converges to the global optimal of problem **P1** with a high probability when ω decreases. The algorithm converges to global optimal with probability of 1 when $\omega\to 0$.*

Proof: Let $\boldsymbol{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_C\}$ be the decision space of service caching, where C is the number of all possible alternative choices. In each iteration, we stochastically choose a caching decision in $\boldsymbol{S}$ to the randomly selected edge node k. Following iterations over each edge node and decision, the service caching strategies $\boldsymbol{X}$ evolves in an E-dimension Markov chain with k-th dimension shows service caching decision of edge node k. We start with a simple case which consists of 2 edge nodes, and $\langle \boldsymbol{x}_1, \boldsymbol{x}_2\rangle$ represent the Markov chain. In view of only one randomly selected edge node k updates the caching decision from $\boldsymbol{x}_k$ with arbitrary at each iteration, we suppose the decision $\boldsymbol{x}_1$ changes to $\boldsymbol{x}_1^*$ firstly:

$$
\begin{aligned}
&\Pr\left(\langle x_1^*, x_2\rangle \mid \langle x_1, x_2\rangle\right) = \\
&\frac{1}{2C(1+e^{\frac{g(\langle x_1^*,x_2\rangle)-g(\langle x_1,x_2\rangle)}{\omega}})} = \frac{e^{\frac{g(\langle x_1,x_2\rangle)}{\omega}}}{2C(e^{\frac{g(\langle x_1^*,x_2\rangle)}{\omega}}+e^{\frac{g(\langle x_1,x_2\rangle)}{\omega}})},
\end{aligned}
\tag{8}
$$

where $g(\langle x_1, x_2\rangle)$ is the objective value at caching strategy $\langle x_1, x_2\rangle$. Once again, we desire a minimal cost, which means the smaller $g(\boldsymbol{x}_k)$, the higher probability to choose service caching decision $\boldsymbol{x}_k$. Thus we can rewrite (8) to draw the

above statement more suitable:

$$\Pr\left(\langle x_1^*, x_2\rangle \mid \langle x_1, x_2\rangle\right) = \frac{e^{-\frac{g(\langle x_1^*, x_2\rangle)}{\omega}}}{2C(e^{-\frac{g(\langle x_1^*, x_2\rangle)}{\omega}} + e^{-\frac{g(\langle x_1, x_2\rangle)}{\omega}})}. \tag{9}$$

Similarly, we can conclude:

$$\Pr\left(\langle x_1, x_2^*\rangle \mid \langle x_1, x_2\rangle\right) = \frac{e^{-\frac{g(\langle x_1, x_2^*\rangle)}{\omega}}}{2C(e^{-\frac{g(\langle x_1, x_2^*\rangle)}{\omega}} + e^{-\frac{g(\langle x_1, x_2\rangle)}{\omega}})}. \tag{10}$$

In addition, let $\theta(\langle x_1, x_2\rangle)$ be the stationary probability distribution at caching strategy $\langle x_1, x_2\rangle$. Therefore, we have a balanced equation derived from stationary condition of the Markov chain as:

$$\begin{aligned}&\theta(\langle s_1, s_1\rangle)\ \Pr\left(\langle s_1, s_c\rangle|\langle s_1, s_1\rangle\right)\\ &= \theta(\langle s_1, s_c\rangle)\ \Pr\left(\langle s_1, s_1\rangle|\langle s_1, s_c\rangle\right).\end{aligned} \tag{11}$$

Substituting (11) with (9) and (10), we have:

$$\begin{aligned}&\theta(\langle s_1, s_1\rangle) \times \frac{e^{-\frac{g(\langle s_1, s_c\rangle)}{\omega}}}{2C(e^{-\frac{g(\langle s_1, s_c\rangle)}{\omega}} + e^{-\frac{g(\langle s_1, s_1\rangle)}{\omega}})}\\ &= \theta(\langle s_1, s_c\rangle) \times \frac{e^{-\frac{g(\langle s_1, s_1\rangle)}{\omega}}}{2C(e^{-\frac{g(\langle s_1, s_c\rangle)}{\omega}} + e^{-\frac{g(\langle s_1, s_1\rangle)}{\omega}})}.\end{aligned} \tag{12}$$

We find that (12) is symmetric and the set of equations are balanced if $\theta(\langle x_1, x_1\rangle) = \tau e^{-\frac{g(\langle x_1, x_2\rangle)}{\omega}}$, where τ is a constant. Let Π be the strategy space and we make the stationary probability distribution as:

$$\begin{aligned}&\theta(\langle x_1, x_2\rangle) =\\ &\frac{e^{-\frac{g(\langle x_1, x_2\rangle)}{\omega}}}{\sum\limits_{\langle x_1^u, x_2^u\rangle \in \Pi} e^{-\frac{g(\langle x_1^u, x_2^u\rangle)}{\omega}}} = \frac{1}{\sum\limits_{\langle x_1^u, x_2^u\rangle \in \Pi} e^{\frac{g(\langle x_1, x_2\rangle) - g(\langle x_1^u, x_2^u\rangle)}{\omega}}},\end{aligned} \tag{13}$$

which ensures $\sum\limits_{\langle x_1, x_2\rangle \in \Pi} \theta(\langle x_1, x_2\rangle) = 1$.

Let $\langle x_1^*, x_2^*\rangle$ be the globally optimal service caching strategy which minimizes the objective value. In other words, the inequation $g(\langle x_1^*, x_2^*\rangle) \leq g(\langle x_1^u, x_2^u\rangle)$ is always valid for any $\langle x_1^u, x_2^u\rangle \in \Pi$. From (13), we know that $\theta(\langle x_1^*, x_2^*\rangle)$ increases with ω decreases. Further, $\lim\limits_{\omega \to 0} \theta(\langle x_1^*, x_2^*\rangle) = 1$ proves the algorithm can converge to the globally optimal objective value with the probability of 1 when $\omega \to 0$. The analysis can be extended to the E-dimensional Markov chain analogously, which ultimately fulfill the proof. ■

B. Resource Allocation and Task Scheduling

With the results of service caching strategy on each edge node, we further solve the problem of how many resources to use and how to schedule tasks among edge nodes. In this subsection, we work out **P2** by transforming it into linear programming problem.

First of all, the allocation of storage resources is confirmed as long as $\boldsymbol{x}_k$ has been determined. Specifically, storage resource requirement is related to application data and databases of the service. The requirement is relatively constant in the given time interval, so that only computation resources affect the deployment cost in this subproblem. The cost of resource in edge node k is then intensively relevant to the amount of consumed computation resources $\sum\limits_{i \in \mathcal{N}} z_k^i$. In addition, with the number of tasks transmitted from base stations to a certain edge node increases, the allocated computation resources ought to be increased accordingly, and vise versa. In other words, it is visible to claim that $\sum\limits_{j \in \mathcal{B}} y_{jk}^i$ and z_k^i are strongly correlative. In order to illustrate the indirect impact of task scheduling for cost, as well as simplify the problem, we introduce a vector $\boldsymbol{v}$ to merge these two variables.

To be intuitive, we integrate z_k^i and y_{jk}^i into a variable $\boldsymbol{v}$ to turn **P2** into a typical linear programming problem. Let $m = N \times E + N \times E \times B$ denote the length of $\boldsymbol{v}$. Let $\boldsymbol{v}_q$ indicate the original z_k^i and the range of q is $\{1, 2, ..., N \times E\}$. We use $\boldsymbol{v}_h$ to depict the original y_{jk}^i and the range of h is $\{(N \times E + 1), (N \times E + 2), ..., m\}$ likewise.

We can rewrite the objective in **P2** as:

$$\mathbf{P2}^*: \min_{v_q} \sum_{k \in \mathcal{E}} \sum_{i \in \mathcal{N}} x_k^i \cdot \left(\frac{\alpha_i}{A_k} \cdot P_k^{\mathrm{a}} + \frac{v_q}{F_k} \cdot P_k^{\mathrm{f}}\right),$$

where $q = (i-1) \times E + k$ and $h = N \times E + (i-1) \times N \times E + (j-1) \times E + k$ $(i \in \mathcal{N}, j \in \mathcal{B}, k \in \mathcal{E})$.

We can obtain $\sum\limits_{k \in \mathcal{E}} v_h = 1$ from (2), and the value range is $0 \leq v_h \leq x_k^i, v_h \in [0, 1]$. Besides, restrictions of computation resources are $\sum\limits_{i \in \mathcal{N}} v_q \leq F_k$ and $v_q \leq x_k^i \cdot F_k$. With the fact that $\mu_i > 0$, we convert the queuing method constraint to a more typical form in linear programming: $-v_q < -\mu_i \cdot (\sum\limits_{j \in \mathcal{B}} v_h \cdot \lambda_j^i)$, and latency requirement $(T_{jk} - \Phi_i) \cdot (\frac{v_q}{\mu_i} - \sum\limits_{j \in \mathcal{B}} v_h \cdot \lambda_j^i) \leq -1$.

We have transformed the resource allocation and task scheduling subproblem into a linear programming problem $\mathbf{P2}^*$ with $\boldsymbol{v}$. In the context of the simplification, we are able to gain the optimal objective value easily. We are willing to take advantage of primal-dual method with polynomial complexity in the class of interior-point methods. Therefore, we employ an optimization function module equipped with the method named linprog [33], [34] from SciPy libraries[1].

V. Simulation Results

In this section, we compare the performance of service deployment cost between our proposed algorithm and three benchmarks. We simulate the hierarchical MEC as a three-layer network with 10 different edge nodes. The edge nodes are equipped with diverse but limited storage resources and computation resources. We organize five edge nodes in Layer 1, four edge nodes in Layer 2, and one edge node in Layer 3. Besides, we set the values of parameters basically according to [4], [5], [27], and Ali Cloud servers[2] (the price of resource is calculated hourly).

Furthermore, we form the hierarchical network structure by making a gap in value of edge node parameters over different layers. To be specific, edge nodes in Layer 1 are empowered by

[1]https://docs.scipy.org/doc/scipy

[2]https://www.aliyun.com/price

fewest resources and highest price, although communication latency to base stations is the shortest. On the contrary, the edge node in Layer 3 has the most sufficient resources and lowest price, but occurs the longest communication latency. In other words, the evaluation of resources and price has a separation between different layers, yet the difference is smaller within the same layer. The selections in value range of any parameter follow uniform distribution. In summary, all the parameters are listed in Table I.

We compare the performance on deployment cost between our proposed algorithm and following three benchmarks.

Random: Services are cached on edge nodes randomly, that is, we only optimize resource allocation and task scheduling in this method.

Coarse: Service caching strategies are addressed on the basis of Gibbs Sampling, but all the available computation resources in the edge node are consumed without allocation once service cached.

Greedy: Services are sorted according to latency requirement respectively. Latency-sensitive services have higher priority to be cached. For each base station, we calculate the preference on edge nodes by product of latency and price.

A. *Performance Comparison on Service Deployment Cost*

We compare the the performance of deployment cost with 8 kinds of services and task arrival rates in the range of [50, 80]. As shown in Fig. 2, the proposed algorithm obtains the minimal cost after iterations. Results of Coarse is visibly larger than others, which caused by the indiscriminate usage of computation resources. Additionally, deployment cost of Coarse decreases with iteration but the change is relative small, compared with our algorithm. This is owing to tiny effect on storage resource cost since all the computation resources are occupied in the caching edge node. We can observe that objective value of Random suddenly decreases sharply in couple of iterations. The results are drawn as the stepped shape in Fig. 2, which illustrates one of the main weaknesses, that is, a feasible solution is difficult to reach. It will take a long time to iterate and seek out a better objective value since alternative solutions are massive yet selections are random. Therefore, changes are infrequent in Random and another valid solution is hard to explore. Furthermore, the results of Greedy can be obtained without iteration, which leads to the straight line in Fig. 2. We multiply price of the edge node and communication latency to each base station, the products reflect the preference substantially. We calculate the preference on edge nodes for each base station and transmit tasks according to the lists. However, the result is not ideal as Greedy can only choose one strategy in a greedy manner. Different from the benchmarks, the proposed algorithm achieve minimal cost by jointly considering service caching, resource allocation and task scheduling.

Moreover, our proposed algorithm always achieves the minimal cost with various number of services and task arrival rates. As shown in Fig. 3(a), with the increasing service number varies from 5 to 12, the cost tends to be higher in general. Note that the cost tends to decrease on couple of points in each method. It can be explained that we randomly evaluate task arrival rates from [50, 80], thus computation consumption unlikely increases with accelerate number of services. Nevertheless, the cost increases with task arrival rates change from 45 to 100 (the amount of arriving tasks at base stations per second) respectively. In addition, the trend of cost is relatively unconspicuous in Coarse with both different number of services and task arrival rates. It also confirms the analysis in the previous that cost of storage resource consumption causes little difference in Coarse.

TABLE I: Simulation Parameters

Parameter	**Value**
Edge node storage resource A_k	[50, 200] GB
Edge node storage resource price P_k^{a}	[10, 40] CNY
Edge node computation resource F_k	[50, 150] Giga CPU cycles/s
Edge node computation resource price P_k^{f}	[10, 50] CNY
Service storage requirement α_i	[10, 40] GB
Service computation requirement μ_i	[0.1, 0.5] Giga CPU cycles/task
Service latency requirement Φ_i	[10, 50] ms
Communication latency T_{jk}	[1, 15] ms
Smooth parameter ω	10^{-6}

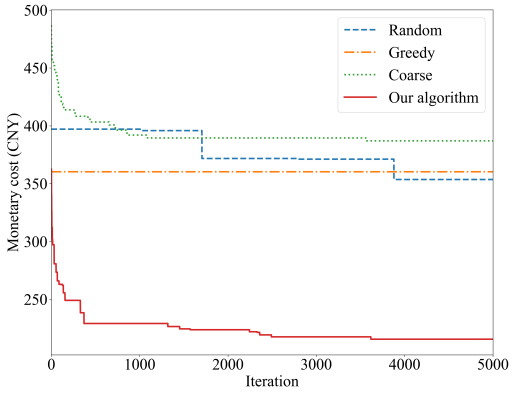
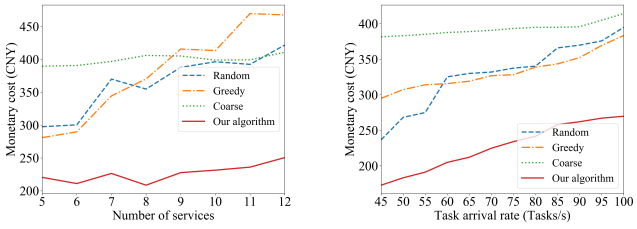


Fig. 2: Service deployment cost.



(a) (b)

Fig. 3: Performance comparison on service deployment cost.

B. *Convergence of the proposed algorithm*

According to Theorem 1, the proposed algorithm which exploiting Gibbs sampling converges to the global optimal with probability of 1 when ω is close to 0. We illustrate the influence of ω in Fig. 4.

The total deployment cost can converge with iteration at each ω, but the objective value is smaller as ω decreases in general. The converging rate is faster when ω decreases. These observations in Fig. 4 can be explained by Step 7 of Algorithm 1 and (13). The probability of an edge node changes service caching strategy increases in each iteration as ω decreases. Thus the smaller ω, the faster that objective value converges, and the probability approaches to 1 when ω closes to 0. In other words, the smaller ω, the more probable that our proposed algorithm converges to globally optimal results.

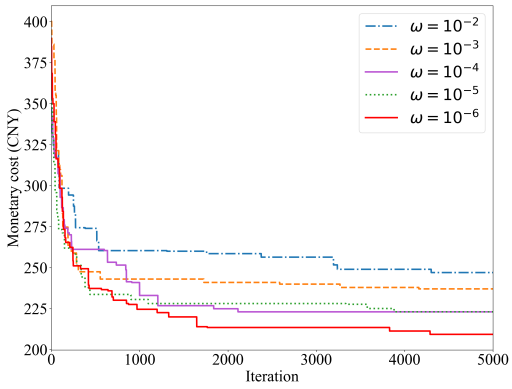


Fig. 4: Impact of ω on the convergence of our algorithm.

VI. Conclusion

In this paper, we have investigated service deployment in hierarchical mobile edge computing with three aspects: service caching, resource allocation and task scheduling, aiming at minimizing monetary cost. We have formulated the problem as mixed integer non-linear programming problem and proposed an iterative service deployment algorithm by exploiting Gibbs sampling. We have decoupled the problem and obtain optimal service caching strategies of edge nodes firstly. Furthermore, we have transformed the resource allocation and task scheduling subproblem into linear programming, and solve the equal linear problem by a typical optimization function. Finally, extensive simulations show the effectiveness and advantages on minimizing deployment cost of our algorithm.

References

[1] Q. Duan, S. Wang, and N. Ansari, "Convergence of networking and cloud/edge computing: Status, challenges, and opportunities," *IEEE Network*, vol. 34, no. 6, pp. 148 – 155, 2020.

[2] Cisco, "Cisco annual internet report (2018–2023)," *White Paper*, 2020.

[3] K. Ha, P. Pillai, W. Richter, Y. Abe, and M. Satyanarayanan, "Just-in-time provisioning for cyber foraging," in *Proc. 11th Annual International Conference on Mobile Systems, Applications, and Services, (MobiSys'13), Taipei, Taiwan, June 25-28, 2013*, pp. 153 – 166.

[4] ETSI, MECISG, "Mobile edge computing (mec); framework and reference architecture," *ETSI, DGS MEC*, 2016.

[5] IMT-2020 (5G) Promotion Group, "5G network architecture design white paper," *White Paper*, 2016.

[6] China Unicom, "China unicom edge computing technology white paper," *White Paper*, 2017.

[7] K. Poularakis, J. Llorca, A. M. Tulino, I. J. Taylor, and L. Tassiulas, "Joint service placement and request routing in multi-cell mobile edge computing networks," in *Proc. IEEE Conference on Computer Communications, (INFOCOM'19), Paris, France, April 29 - May 2, 2019*, pp. 10 – 18.

[8] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *Proc. IEEE Conference on Computer Communications, (INFOCOM'19), Paris, France, April 29 - May 2, 2019*, pp. 1459 – 1467.

[9] X. Guo, T. Wang, and S. Wang, "Joint optimization of caching and routing strategies in content delivery networks: A big data case," in *Proc. IEEE International Conference on Communications, (ICC'19), Shanghai, China, May 20-24, 2019*, pp. 1 – 6.

[10] Z. Xu, L. Zhou, S. C. Chau, W. Liang, Q. Xia, and P. Zhou, "Collaborate or separate? distributed service caching in mobile edge clouds," in *Proc. IEEE Conference on Computer Communications, (INFOCOM'20), Toronto, ON, Canada, July 6-9, 2020*, pp. 2066 – 2075.

[11] Z. Zhong, J. Qin, Z. Zhong, and Z. Li, "Fog radio access networks with hierarchical content delivery," *IEEE Access*, vol. 7, pp. 20 950 – 20 960, 2019.

[12] W. Jiang, G. Feng, and S. Qin, "Optimal cooperative content caching and delivery policy for heterogeneous cellular networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 5, pp. 1382 – 1393, 2017.

[13] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, and F. Tian, "Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud ran," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3282 – 3299, 2020.

[14] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 360 – 374, 2021.

[15] B. Feng, L. Jiang, G. Feng, S. Qin, and Y. Guo, "Network coding based content caching in hierarchical cloud service network for 5g," in *Proc. IEEE Global Communications Conference, (GLOBECOM'16), Washington, DC, USA, December 4-8, 2016*, pp. 1–6.

[16] X. Zhang and Q. Zhu, "Collaborative hierarchical caching over 5g edge computing mobile wireless networks," in *Proc. IEEE International Conference on Communications, (ICC'18), Kansas City, MO United States, May 20-24, 2018*, pp. 1 – 6.

[17] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE Conference on Computer Communications, (INFOCOM'18), Honolulu, HI, USA, April 15-19, 2018*, pp. 207 – 215.

[18] J. Meng, H. Tan, X. Li, Z. Han, and B. Li, "Online deadline-aware task dispatching and scheduling in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1270 – 1286, 2020.

[19] T. Zhu, T. Shi, J. Li, Z. Cai, and X. Zhou, "Task scheduling in deadline-aware mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4854 – 4866, 2019.

[20] T. He, H. Khamfroush, S. Wang, T. L. 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 International Conference on Distributed Computing Systems, (ICDCS'18), Vienna, Austria, July 2-6, 2018*, pp. 365 – 375.

[21] V. Farhadi, F. Mehmeti, T. He, T. L. Porta, H. Khamfroush, S. Wang, and K. S. Chan, "Service placement and request scheduling for data-intensive applications in edge clouds," in *Proc. IEEE Conference on Computer Communications, (INFOCOM'19), Paris, France, April 29 - May 2, 2019*, pp. 1279 – 1287.

[22] L. Yang, J. Cao, G. Liang, and X. Han, "Cost aware service placement and load dispatching in mobile cloud systems," *IEEE Transactions on Computing*, vol. 65, no. 5, pp. 1440 – 1452, 2016.

[23] Y. Chen, S. Deng, H. Zhao, Q. He, Y. Li, and H. Gao, "Data-intensive application deployment at edge: A deep reinforcement learning approach," in *Proc. IEEE International Conference on Web Services, (ICWS'19), Milan, Italy, July 8-13, 2019*, pp. 355 – 359.

[24] S. Pasteris, S. Wang, M. Herbster, and T. He, "Service placement with provable guarantees in heterogeneous edge computing systems," in *Proc. IEEE Conference on Computer Communications, (INFOCOM'19), Paris, France, April 29 - May 2, 2019*, pp. 514 – 522.

[25] J. Zhou, J. Fan, J. Wang, and J. Jia, "Dynamic service deployment for budget-constrained mobile edge computing," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 24, pp. 5436 – 5452, 2019.

[26] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, "Adaptive user-managed service placement for mobile edge computing: An online learning approach," in *Proc. IEEE Conference on Computer Communications, (INFOCOM'19), Paris, France, April 29 - May 2, 2019*, pp. 1468 – 1476.

[27] X. Ma, A. Zhou, S. Zhang, and S. Wang, "Cooperative service caching and workload scheduling in mobile edge computing," in *Proc. IEEE Conference on Computer Communications, (INFOCOM'20), Toronto, ON, Canada, July 6-9, 2020*, pp. 2076 – 2085.
[28] T. Nguyen, E. Huh, and M. Jo, "Decentralized and revised content-centric networking-based service deployment and discovery platform in mobile edge computing for iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4162 – 4175, 2019.
[29] Y. Liang, J. Ge, S. Zhang, J. Wu, L. Pan, T. Zhang, and B. Luo, "Interaction-oriented service entity placement in edge computing," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 1064 – 1075, 2021.
[30] Z. Xiaojian and Z. M. Chu, "Multi-objective optimized cloudlet deployment and task offloading for mobile edge computing," *IEEE Internet of Things Journal*, pp. 1–1, 2021.
[31] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 939 – 951, 2021.
[32] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, *Markov Chain Monte Carlo in Practice (Interdisciplinary Statistics)*. Chapman and Hall/CRC, 1996.
[33] E. D. Andersen and K. D. Andersen, "Presolving in linear programming," *Mathematical Programming*, vol. 71, no. 2, pp. 221 – 245, 1995.
[34] E. D. Andersen and K. D. Andersen, "The mosek interior point optimizer for linear programming: An implementation of the homogeneous algorithm," *High performance optimization. Springer US*, pp. 197 – 232, 1999.

Jie Huang is currently a Ph.D. candidate at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications (BUPT). Her research interests include mobile edge computing and service computing. Contact her at huangjie@bupt.edu.cn.

Ao Zhou received Ph.D. degree in Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2015. She is currently an Associate Professor with State Key Laboratory of Networking and Switching Technology, BUPT. She has published 50+ research papers. She played a key role at many international conferences. Her research interests include cloud computing and mobile edge computing.

Shangguang Wang is a Professor at the School of Computer Science and Engineering, Beijing University of Posts and Telecommunications, China. He received his Ph.D. degree at Beijing University of Posts and Telecommunications in 2011. He has published more than 150 papers. His research interests include service computing, mobile edge computing, and satellite computing. He is currently serving as Chair of IEEE Technical Committee on Services Computing (2022-2013), and Vice-Chair of IEEE Technical Committee on Cloud Computing (2020-). He also served as General Chairs or Program Chairs of 10+ IEEE conferences. He is a Fellow of the IET, and Senior Member of the IEEE. For further information on Dr. Wang, please visit: http://www.sguangwang.com