Transfer Learning Based Algorithm for Service Deployment Under Microservice Architecture

Wenlin Li[1(✉)], Bei Liu[1,2], Hui Gao[3], and Xin Su[4]

[1] School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing, China
S190101011@stu.cqupt.edu.cn

[2] Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing, China
liubei@mail.tsinghua.edu.cn

[3] Beijing University of Posts and Telecommunications, Beijing, China
huigao@bupt.edu.cn

[4] Tsinghua University, Beijing, China
suxin@tsinghua.edu.cn

Abstract. In recent years, with the large-scale deployment of 5G network, research on 6G networks has gradually begun. In the 6G era, new service scenarios, such as Broad Coverage and High Latency Communication (BCHLC) will be introduced into the network, further increasing the complexity of network management. Furthermore, the development of edge computing and microservice architectures enables services to be deployed in a container on the edge clouds closer to the user side, significantly solving the problems. However, how to deploy services on edge clouds with limited resources is still an unresolved problem. In this paper, we model the problem as a Markov Decision Process (MDP), then propose a Deep Q Learning (DQN) based service deployment algorithm to optimize the delay and deployment cost of the services. Furthermore, a Multi-Category Joint Optimization Transfer Learning (MCJOTL) algorithm is proposed in this paper to address the problem of slow convergence of the DQN algorithm, which can adapt to different service scenarios in future networks faster. The simulation results show that the proposed algorithm can effectively improve training efficiency and service deployment effects.

Keywords: Service deployment · Edge computing · DQN · Transfer learning

1 Introduction

With the rapid development of wireless communication, the number of mobile applications and services continues to grow. The current network is facing considerable difficulties in dealing with the exponentially increasing service demands

Supported by the National Key R&D Program of China under Grant 2020YFB1806702.

H. Gao et al. (Eds.): ChinaCom 2021, LNICST 433, pp. 52–62, 2022.
https://doi.org/10.1007/978-3-030-99200-2_5

of mobile users. In the future, the 6G network will have more stringent requirements on delay, throughput, and other aspects. Taking cloud virtual reality (VR) as an example, it is expected to achieve an end-to-end delay within 10 ms [1]. At the same time, in order to adapt to the richer scenarios in the future 6G era, literature [2] further proposes the fourth service scenario, Broad Coverage and High Latency Communication (BCHLC) based on the three service scenarios of enhanced Mobile Broadband (eMBB), massive Machine Type Communication (mMTC) and Ultra Reliable Low Latency Communication (URLLC) in 5G network.

Recently, with continuous research of mobile edge computing (MEC) technology, edge cloud has shown great potential in deploying services on the edge of the network that requires large amounts of resource consumption and delay-sensitive services. Compared with mobile devices and remote clouds, the edge clouds allow users to use the strong computing power of the cloud platform without causing a high delay in communication with remote clouds [3], thereby significantly reducing the data traffic to and from the core network and meeting the requirements of delay-sensitive services. At the same time, with the rise of microservice architecture, services can be dynamically deployed on the edge clouds in a container, which makes future networks more flexible and more scalable [4].

Compared with the traditional remote cloud, the edge clouds are composed of resource-constrained machine clusters. Therefore, it is sential to deploy the services on them appropriately. However, ensuring the effective use of resources on the edge clouds, and the delay of the services, are challenging research problems [5]. Literature [6] proposes a distributed MEC-based service deployment platform, and it introduces a protocol to help service providers deploy services on MEC nodes to reduce transmission delay. Literature [7] proposes a Deep Reinforcement Learning (DRL) based dynamic service orchestration algorithm, taking into account the dynamic migration of the service, which minimizes the orchestration cost and improves the quality of service deployment.

However, these studies do not fully consider the transferability of the models. Therefore, they are difficult to realize the rapid changing of the dynamic network environment. Compared with the above researches, we study the problem of deploying services on edge clouds under microserice architecture. We consider delay and deployment cost to achieve better service deployment effect. First, we model the problem as a Markov Decision Process (MDP), and propose a Deep Q Learning (DQN) based service deployment algorithm. At the same time, to adapt to the ever-changing communication scenarios in the 6G network, we introduce a Multi-Category Joint Optimization Transfer Learning (MCJOTL) algorithm to speed up the model and can adapt to different service scenarios faster. The simulation results show that the algorithm can faster adapt to different network environments and obtain better service deployment effects.

The rest of the paper is organized as follows. First, we describe the system model in Sect. 2. Section 3 introduces the proposed MCJOTL algorithm of service deployment based on DQN and Transfer Learning. Simulation results are presented in Sect. 4. Finally, we conclude this paper in Sect. 5.

2 System Model

We consider a service development scenario in Fig. 1, which contains a central cloud, N edge clouds and K users, i.e., $\mathcal{N} = \{1, 2, \ldots, N\}$, $\mathcal{K} = \{1, 2, \ldots, K\}$. At the same time, each edge cloud has limited amounts of multi-dimensional resources and belongs to different operators. Suppose there are O operators in total, i.e., $\mathcal{O} = \{1, 2, \ldots, O\}$, and the resource prices provided by different operators are different. Then, the set of service requests generated by K users is denoted as $\mathcal{J} = \{1, 2, \ldots, J\}$, and the services reach the network in a uniformly distributed form. Finally, the global controller located on the central cloud determines the deployment location of the services.

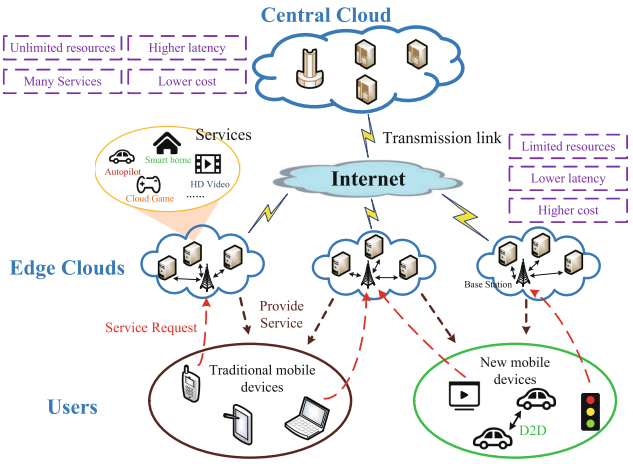

Fig. 1. 6G service deployment scenarios

2.1 Cloud Model

We assume that there are unlimited traffic, storage, and computing resources on the central cloud. While on the edge clouds, multi-dimensional resources are limited, and when new services reach or the services deployed on edge clouds end their lives, they will report their remaining resources to the global controller. Therefore, we use a four-tuple $E(f_{e,i}, s_{e,i}, h_{e,i}, o_i)$, $i \in \mathcal{N}$, $o_i \in \mathcal{O}$ to describe the edge clouds, which are the remaining traffic, storage, and computing resources on the edge cloud i, and the operator to which it belongs.

At the same time, there are differences in resource prices provided by operators, which can be expressed as $\{l_{f,i}, l_{s,i}, l_{h,i}\}$, $i \in \mathcal{O}$, representing the prices of traffic, storage, and computing resources provided by an operator i. Especially, it represents the price of resources provided on the central cloud at the time $i = 0$.

2.2 Service Model

In this scenario, we consider the following four categories of services: mMTC, eMBB, URLLC, and BCHLC [2]. Each service category has large different requirements in terms of traffic, capacity, and computing resource, and there are W types of service in the network that belong to the above four categories. Therefore, we use a five-tuple $B(t_{s,i}, t_{e,i}, f_{s,i}, s_{s,i}, h_{s,i})$, $i \in \mathcal{J}$ to describe the services, which are the time when the service i starts and ends, and the required traffic, storage, and computing resources of service i under normal circumstances. Considering the dynamic nature of the service process, we further assume the maximum instantaneous computing resource demand of the service i as $h_{max,i}$. At the same time, each service needs to correspond to a user, and also needs to be deployed on one cloud, so the total amount of service deployed on the clouds at a time t is:

$$J_t = \sum_{i=1}^{J} u_i, \text{where } u_i = \begin{cases} 1 & \text{if } t_{s,i} \leq t \leq t_{e,i} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

In the process of deploying services on the clouds, the remaining traffic, capacity, and computing resources in the clouds will be considered. At the same time, in order to save the occupation of storage resources in the edge clouds, all services are provided to users in a multi-threaded way. So, when the same type of service is deployed on the edge cloud, the storage space occupied by them will be the maximum value of this type of service, i.e.,

$$\begin{aligned} &C_1 : \textstyle\sum_{i=1}^{M_{j,t}} f_{s,i} \leq f_{e,j} \quad C_2 : \sum_{k=1}^{W} \underset{k}{max}(s_{s,j,k}) \leq s_{e,j} \\ &C_3 : \textstyle\sum_{i=1}^{M_{j,t}} h_{s,i} \leq h_{e,j} \quad C_4 : M_{0,t} + \sum_{j=0}^{N} M_{j,t} = J_t \end{aligned} \tag{2}$$

where $j \in \mathcal{N}$, $\underset{k}{max}(s_{s,j,k})$represents the maximum storage resource occupied by services of the type k deployed on the edge cloud j, $M_{j,t}$ is the number of services deployed on the edge cloud j at the time t, and $M_{0,t}$ represents the number of services deployed on the central cloud at the time t.

2.3 Delay Model

After deployment, the service process may include two steps: the service i is calculated on the cloud, and the cloud transmits the calculation result to the user. Therefore, two kinds of delays are generated in the above process, i.e., the calculation delay $r_{c,i}$ and the transmission delay $r_{t,i}$.

If the service is deployed on the edge cloud, it will directly provide services to the users. For the wireless channel, assuming that the bandwidth allocated to the service i after development is W_i, and the signal-to-noise ratio between the user j and the edge cloud k of the service is $\frac{S}{N}_{j,k}$, then the maximum transmission rate of service i is

$$C_{i,k} = W_i log_2(1 + \frac{S}{N}_{j,k}) \tag{3}$$

Then the transmission delay of the service i can be expressed as

$$r_{t,i} = \frac{f_{s,i}}{C_{i,k}} \tag{4}$$

Otherwise, if the service is deployed on the central cloud, the required data will be transferred by the edge clouds when provides services. And there is no need to occupy the remaining traffic resources in the edge cloud during the transfer process. Therefore, the total delay includes the delay between the central and edge clouds and between edge clouds and users. Then the transmission delay of service i can be written as:

$$r_{t,i} = \min_{k} \left\{ \frac{f_{s,i}}{C_{i,k}} + r_{t,c,k} \right\} \tag{5}$$

where $r_{t,c,k}$ represents the transmission delay between the central cloud and the edge cloud k.

At the same time, we express the calculation delay as the ratio between the maximum computing resource demand of the service i and the allocated computing resources, i.e.,

$$r_{c,i} = \frac{h_{max,i}}{h_{s,i}} \tag{6}$$

Then after the service i is deployed, the total delay in providing services is

$$r_i = r_{t,i} + r_{c,i} \tag{7}$$

2.4 Cost Model

When a service deployed on the cloud provides services to users, it will incur a specific cost. Considering that due to different operators, there are differences in the rental costs of resources such as traffic on different edge clouds.

In this scenario, the service cost i consists of three parts: traffic cost $c_{f,i}$, storage rental cost $c_{s,i}$, and computing resource rental cost $c_{h,i}$. First, the cost of traffic is calculated according to the actual usage. That is

$$c_{f,i} = f_{s,i} l_{f,j}, i \in \mathcal{J}, j \in \mathcal{O} \tag{8}$$

where j is the operator of the edge cloud k. However, the rental cost of storage and computing is calculated according to the usage time and the amount of resource allocation, i.e.,

$$\begin{cases} c_{s,i} = s_{s,i} l_{s,j} (t_{e,i} - t_{s,i}) \\ c_{h,i} = h_{h,i} l_{h,j} (t_{e,i} - t_{s,i}) \end{cases}, \ i \in \mathcal{J}, i \in \mathcal{O} \tag{9}$$

Consider that if there is no service deployed on the current edge cloud, the edge cloud should be in a dormant state. Then, after the service is deployed for the first time, the edge cloud needs to enter the working state, and the startup cost c_0 of the edge cloud needs to be added.

So, the total cost of the service deployment is

$$c_i = c_{f,i} + c_{s,i} + c_{h,i} + c_0 \tag{10}$$

At the same time, because the central cloud has unlimited traffic, storage, and computing resources, edge clouds have little traffic, storage, and computing resources. Therefore, the resource prices on edge clouds provided by operators are greater than the resource prices on the central cloud, i.e.,

$$l_{f,i} > l_{f,0}, l_{s,i} > l_{s,0}, l_{h,i} > l_{h,0}, i \neq 0 \tag{11}$$

Therefore, if the service provider cannot pay the deployment cost on edge clouds, the service must be deployed on the central cloud, i.e.,

$$C_5 : c_i \leq c_{\text{tolerate},i} \tag{12}$$

where $c_{\text{tolerate},i}$ represents the maximum deployment cost that the provider of service can tolerate.

2.5 Problem Formation

Finally, in the actual service deployment process, we will comprehensively consider the delay and the deployment cost to achieve better development effects. So we constructed the following optimization problem

$$\begin{aligned} &min\,\{\alpha r_i + \beta c_i\} \\ &\text{s.t.}\, C_1, C_2, C_3, C_4, C_5 \end{aligned} \tag{13}$$

where α and β are constants.

The above optimization problem is NP-hard. In order to obtain the optimal solution to this problem, we need to search the entire combination space. However, the combination space will increase exponentially as users and edge clouds increase, bringing great difficulties.

3 Service Deployment Algorithm

For each network, it is difficult not easy to find the optimal service deployment strategy. The reasons are as follows: First, in future network environment, the cloud environment is dynamic due to the continuous change of service status; Second, the interaction between different services is relatively complex. Therefore, for this dynamic environment, we use MDP to model the problem.

3.1 Markov Decision Process

An MDP model can be expressed as $[S, A, R, S']$, where S,A and R are the state space, action space, and reward function, respectively. In the state $s \in S$, the agent will get different rewards when taking actions. Then, according to the state transition function, the state moves from s to $s' \in S$ [8]. The goal of each step is to maximize the reward value. Our model is designed as follows:

State Space. We use nine states included in the service that are waiting to be deployed and the remaining traffic, storage, and computing resources of all edge clouds as the model's state. Specifically, the state $s_n \in S$ of the model at episode n is designed as:

$$s_n = [ser_n; clo_{n,1}, \ldots, clo_{n,N}]^T \tag{14}$$

Which ser_n represents the service ID, continuous service time, traffic, storage, computing, and maximum computing resource demands, type, user of the service, and maximum deployment cost that the service provider can tolerate; $clo_{n,i}$ represents the remaining traffic, capacity, and computing resources in the edge cloud i.

Action Space. According to the state information, the global controller needs to determine the location of service, including edge clouds and central cloud, that is $a \in \{0, 1, 2, \ldots, N\}$, where 0 represents deploying the service on the central cloud.

Reward. When a specific action transforms the state s into another state s', the network will be rewarded immediately. We design the reward as follows

$$R = -(\alpha r_i + \beta c_i) \tag{15}$$

among them, α and β are constants, which are used to adjust the ratio between delay and deployment cost [9].

3.2 DQN Based Service Deployment Algorithm

In order to support the dynamic changes of services and edge clouds, we introduced DQN to optimize the deployment strategy of the services. First, we define the agent's action-value function (Q function) as the maximum expected return achievable after taking some action a in state s, i.e.,

$$Q^*(s, a) = max_\pi(r_n|\pi, s_n = s, a_n = a) \tag{16}$$

where π is the joint policy. Then, we use the following method to update the Q function based on the Bellman equation, i.e.,

$$Q^*(s, a) = r_n + \gamma[\max_{a'} Q^*(s', a')|s, a] \tag{17}$$

It can be observed from (14) that the size of the state space increases exponentially with the arrival of services and the continuous changes of the edge cloud states. In order to solve the problem of dimensionality, we use a deep neural network (DNN) to approximate the Q-function. We define the training loss function of training the DNN as

$$L(\theta_n) = (y_n - Q(s, a; \theta_n))^2 \tag{18}$$

where n represents the number of episodes, y_n represents the target Q-value, θ_n represents the network weight of the Q-network in the nth episode. Then, the gradient of $L(\theta_n)$ is

$$\bigtriangledown_{\theta_n} L(\theta_n) = (y_n - Q(s, a; \theta_n)) \bigtriangledown_{\theta_n} Q(s, a; \theta_n) \tag{19}$$

Based on this, we can use a gradient-based optimizer to train the Q-network. Algorithm 1 gives the details of the algorithm.

Algorithm 1. DQN based service deployment algorithm.

Input: The next service's status that needs to be deployed, edge clouds' status and channel status according to (14)

Output: Deployment location of service on the clouds

```
Initialize state s and experience replay buffer D.
for episode = 0, 1, 2, ... do
    Select an action a^n = i, i ∈ 𝒩 based on the greedy policy, where i represents the
    number of edge clouds.
    if restrictions C_1 ∼ C_5 can be satisfied when the service deployed on the edge
    cloud i then
        Deploy the service on the edge cloud i.
    else
        Deploy the service on the central cloud.
    end if
    Calculate the delay, deployment cost and reward according to (7), (10) and (15)
    Update s' and store the experience (s, a, r, s') in the experience replay buffer D.
    Compute ▽_{θ_n} L(θ_n) by minimizing L(θ_n) and update weights of the Q-network
    according to (18) and (19).
end for
```

3.3 MCJOTL Based Service Deployment Algorithm

Although the DQN algorithm can achieve great service deployment effects, the network is constantly changing due to the continuous changes of the remaining resources in the edge clouds. In addition, the DQN algorithm usually takes a long time to converge, so it is not easy to adapt to the future network.

Therefore, we propose a transfer learning-based MCJOTL algorithm to avoid large-scale training networks from scratch during service deployment. In our MCJOTL algorithm, the DQN results we trained in the previous subsection are used as the source model. We freeze all layers in the DQN model except the last few layers, then add two linear layers at the end, and only the last few layers and the two linear layers can be trained. Thus, each model consists of a frozen pre-trained body and a linear layer body that must be retrained. Finally, the two bodies are merged by linear weighting to improve the overall performance.

At the same time, considering that there are great differences in various types of services in different scenarios. For example, in a cloud gaming scenario, eMBB services may have a large number of requirements, so its heavy traffic

needs need to be prioritized during deployment. However, other types of services, such as URLLC, may have a small amount of requirements. Therefore, the actual requirements of some types of services may be ignored when one model is used for training all types of services in different scenarios. So, our proposed MCJOTL algorithm trains different transfer learning models for different categoties of service in every scenarios. At the same time, each model improves the training effect by sharing replay buffer including state space, reward functions, etc.

Therefore, when the global controller receives service requests, it can select an appropriate transfer learning model according to the service category to which it belongs to determine the optimal service deployment location in MCJOTL. Such as when eMBB service arrives, the model correspond to eMBB will be used. At the same time, the parameters of the model to which the eMBB belongs will also be updated. Thus, the MCJOTL algorithm can better adapt to the most categories of services in different scenarios. By using this algorithm, we can well solve the demand between different categories of services in different scenarios. The simulation results of this algorithm are shown in the following section.

4 Simulation Result

In this section, we demonstrate the effectiveness of the proposed MCJOTL algorithm in terms of deployment cost and delay. We performed the simulation on a 64-bit computer with 32 GB of RAM and Intel i7 2.6 GHz. First, we considered 16 edge clouds, 32 users, and 50 services. In order to evaluate the performance of the algorithm, we used a method similar to the literature [10] to generate datas. For example, we set the delay between the central cloud and edge clouds between 10 and 20 ms. Then, the spectrum bandwidth, service's traffic, storage, computing requirements, etc. are set according to the specific characteristics of the services.

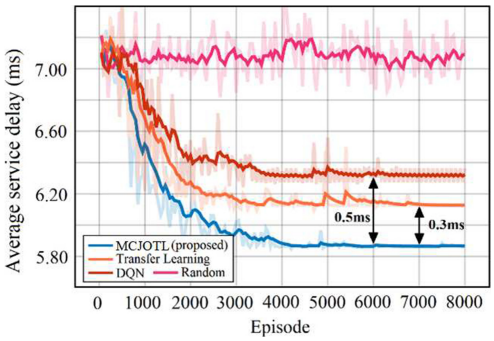

Fig. 2. Variation curve of service delay with episode

We selected DQN, traditional transfer learning, and random deployment algorithms for comparison. The DQN is composed of five linear layers. In transfer

learning and our MCJOTL algorithm, the transfer source comes from the DQN algorithm trained for 50,000 episodes. At the same time, to achieve the training effect faster, all models in the simulation result have only been trained for 8,000 episodes. Figure 2 shows the performance of the MCJOTL algorithm in terms of delay. It can be seen that under the same number of episodes, the MCJOTL algorithm can achieve better service deployment effects by using more prior knowledge and adapting to different service scenarios. Figure 3 shows the algorithm's performance in terms of deployment costs, which shows that the MCJOTL algorithm can also achieve better performance effects in terms of deployment costs.

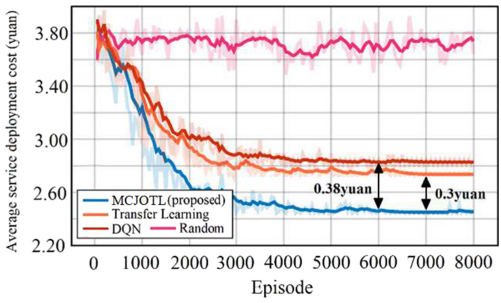

Fig. 3. Variation curve of deployment cost with episode

Finally, Fig. 4 shows the curve of the average delay and deployment cost to the number of edge clouds in the case of 32 users and 50 services. With the increase in the number of edge clouds, delay and deployment costs have gradually improved. However, as the number of edge clouds increases, the improvement gradually slows down in the end.

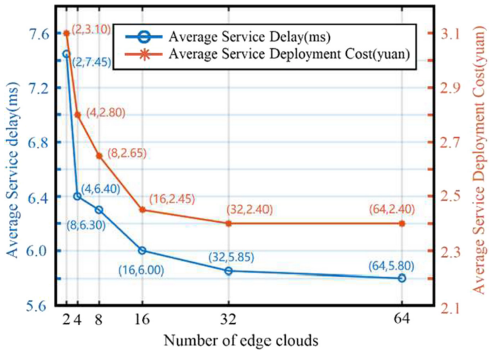

Fig. 4. The change curve of average service deployment cost and service delay with the number of edge clouds

5 Conclusion

In this paper, we studied a deployment algorithm for services. Based on the dynamically changing edge clouds, we model the service deployment problem as an MDP to find an optimal strategy that minimizes delay and development costs. First, we use the DQN algorithm to realize the intelligent deployment of the services. Then, for the slow convergence of DQN, we introduce the MCJOTL algorithm, which trains different models according to the categories of the service. So, when new service arrives, the algorithm uses the corresponding model according to the category of the service to achieve the optimal deployment effects. Beyong this, the MCJOTL algorithm uses the mutual collaboration between different models and the sharing of experience replay buffer to adapt to the service characteristics in different scenarios, so as to achieve better service deployment effects. The simulation results show that the algorithm can effectively improve training efficiency and obtain better service deployment effects.

References

1. Tomkos, I., Klonidis, D., Pikasis, E., Theodoridis, S.: Toward the 6G network era. Opportunities and challenges. IT Prof. **22**(1), 34–38 (2020)
2. Wang, H.: 6G vision: unified network enabling intelligent megalopolis. ZTE Technol. J. **25**(06), 55–58 (2019)
3. Guo, T., Zhang, H., Huang, H., Guo, J., He, C.: Multi-resource fair allocation for composited services in edge micro-clouds. In: 2019 IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), Xiamen, pp. 405–412 (2019)
4. Wang, S., Guo, Y., Zhang, N., Yang, P., Zhou, A., Shen, X.: Delay-aware microservice coordination in mobile edge computing: a reinforcement learning approach. IEEE Trans. Mob. Comput. **20**(3), 939–951 (2021)
5. Badri, H., Bahreini, T., Grosu, D., Yang, K.: A sample average approximation-based parallel algorithm for application placement in edge computing systems. In: 2018 IEEE International Conference on Cloud Engineering (IC2E), Orlando, pp. 198–203 (2018)
6. Nguyen, T., Huh, E., Jo, M.: Decentralized and revised content-centric networking-based service deployment and discovery platform in mobile edge computing for IoT devices. IEEE Internet Things J. **6**(3), 4162–4175 (2019)
7. Guo, S., Dai, Y., Xu, S., Qiu, X., Qi, F.: Trusted cloud-edge network resource management: DRL-driven service function chain orchestration for IoT. IEEE Internet Things J. **7**(2), 6010–6022 (2020)
8. Wang, X., Xu, Y., Chen, J., Li, C., Xu, Y.: 2020 International Conference on Wireless Communications and Signal Processing (WCSP), pp. 195–200, Nanjing (2020)
9. Liu, L., Niyato, D., Wang, P., Han, Z.: Scalable traffic management for mobile cloud services in 5G networks. IEEE Trans. Netw. Serv. Manag. **15**(4), 1560–1570 (2018)
10. Samanta, A., Tang, J.: Dyme: dynamic microservice scheduling in edge computing enabled IoT. IEEE Internet Things J. **7**(7), 6164–6174 (2020)