

INTELLIGENT RADIO: WHEN ARTIFICIAL INTELLIGENCE MEETS THE RADIO NETWORK

Toward Reinforcement-Learning-Based Service Deployment of 5G Mobile Edge Computing with Request-Aware Scheduling

Yanlong Zhai, Tianhong Bao, Liehuang Zhu, Meng Shen, Xiaojiang Du, and Mohsen Guizani

Abstract

5G wireless network technology will not only significantly increase bandwidth but also intro-duce new features such as mMTC and URLLC. However, high request latency will remain a chal-lenging problem even with 5G due to the massive requests generated by an increasing number of devices that require long travel distances to the services deployed in cloud centers. By pushing the services closer to the edge of the network, edge computing is recognized as a promising technology to reduce latency. However, properly deploying services among resource-constrained edge servers is an unsolved problem. In this arti-cle, we propose a deep reinforcement learning approach to preferably deploy the services to the edge servers with consideration of the request patterns and resource constraints of users, which have not been adequately explored. First, the system model and optimization objectives are formulated and investigated. Then the problem is modeled as a Markov decision process and solved using the Dueling-Deep Q-network algo-rithm. The experimental results, based on the eval-uation of real-life mobile wireless datasets, show that this reinforcement learning approach could be applied to patterns of requests and improve performance.

Introduction

With the growth of wireless communication tech-nology, mobile technology encourages nearly the entire world to depend on mobile devices for communication and accessing various services, including some bandwidth-consuming multimedia services. However, due to the increasing num-ber of mobile applications and services, the cur-rent network faces some difficulty in handling the exponential increase of demands from mobile users [1].

The fifth generation (5G) wireless network is a significant leap in the evolution of mobile commu-nication. Carriers anticipate 5G speeds to be 20 to 100 times faster than current mobile networks. In addition to greater bandwidth, 5G networks will deploy ultradense, distributed networks of base stations in small cell infrastructures to provide con-tinuous connections. However, this does not mean the requirements of very low latency and massive

user requests will be met, because the applications and content are mostly deployed in centralized data centers, and time is required for data to travel over the fiber networks connecting the base sta-tions to the network core [2]. A straightforward idea is to move the applications, content, or ser-vices closer to the devices and users, which is the key principle of edge computing. The Internet of Things (IoT), wireless sensor networks [3], and so on are common application scenarios for edge computing. With edge computing, mobile devices using the 5G network can benefit from low latency and reduce the amount of data traffic required to be sent to the backhaul network. Meanwhile, some features will play an important role in 5G wireless networks, such as device-to-device (D2D) commu-nication and massive multiple-input multiple-output (MIMO), which are leveraged to provide massive edge device collaboration without using the core network.

In 5G mobile edge computing, moving appli-cation services and content to the edge network is a key issue. Compared to conventional cloud centers, edge servers, and edge clouds (a small cluster of colocated edge servers) are built from resource-constrained machines. It is vital to appro-priately distribute services to edge servers and edge clouds. However, ensuring multiple objec-tives such as the utilization of resources, efficient response time of applications, throughput of user requests, and so on is a challenging research prob-lem [4, 5]. By utilizing small cell infrastructures, 5G wireless networks could provide better quality of service to local users. Therefore, it is important to consider the geographic locality and access pattern of local user requests. While this prob-lem in edge computing has received increased attention, there are few studies that have inves-tigated the distribution of services on the edge network with the consideration of user location and request patterns. In this article, we investi-gate the problem of deploying the services to the edge servers in the 5G mobile edge computing environment. We propose a new evaluation met-ric called requests per second on edge servers (RPSE) as the optimization objective to increase the number of requests served on edge servers and decrease the response time in the mean-time. The reinforcement learning environment and the system model is established, and a Duel-

Digital Object Identifier: *Yanlong Zhai, Tianhong Bao, Liehuang Zhu and Meng Shen are with Beijing Institute of Technology; Xiaojiang Du is with Temple University;*

10.1109/MWC.001.1900298 *Mohsen Guizani is with Qatar University.*

84 1536-1284/20/$25.00 © 2020 IEEE IEEE Wireless Communications • February 2020

Authorized licensed use limited to: University of New South Wales. Downloaded on November 20,2020 at 02:23:53 UTC from IEEE Xplore. Restrictions apply.



ing-DQN- based service deployment algorithm is proposed to learn the patterns of requests. The experimental results show that the proposed rein-forcement-learning-based approach is superior to relevant approaches at total response time and local processing on the edge networks. The artifi - cial neural network (ANN) could learn the request patterns with a good generalization ability.

ApproAches to the deployment of

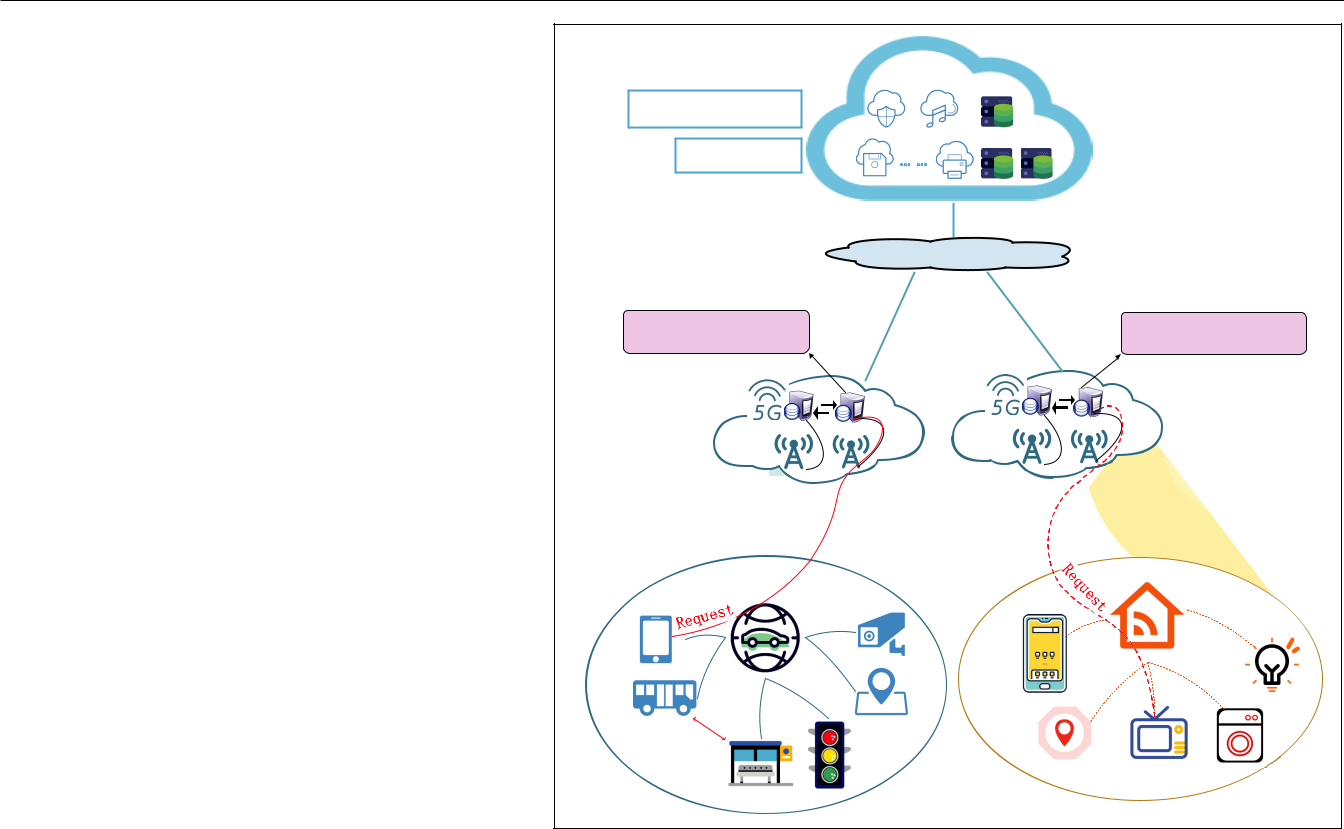
edge servIces/ApplIcAtIons

The approaches to the deployment of edge ser-vices or applications can be classified into two categories: conventional approaches and learn-ing-based approaches.

Conventional approaches usually model the deployment as an optimization problem. The opti-mization model generally considers the following factors [6, 7]: throughput, resource constraints, time, response quantity of the edge, energy, and so on. According to the presence or absence of a customized algorithm, the conventional approach-es can be divided into algorithm-based solutions and software optimizer-based (e.g., IBM ILOG CPLEX) solutions [8]. A software optimizer is the optimization engine that is used to solve some mathematical problems such as linear program-ming, quadratic programming, and mixed-integer programming. Algorithms like greedy search and Partical Swarm Optimization (PSO) are commonly used optimization algorithms. Because of the com-plexity of the application and the multi-objective constraints, it is not easy to define the optimization models. In most cases, these would become NP problems. Heuristic information is required by the algorithms to get the sub-optimal solution. The soft-ware optimizer solutions basically encounter the same problem as the algorithm-based solutions. The time required to find the optimal solution increases exponentially with the scale of the prob-lem, and the adaptability of the models provided by the optimizer is very limited. Moreover, contin-uously changing states make it difficult to model the problem. It is also the reason considering large numbers of requests for a certain period of time in the conventional deployment solutions is less prevalent.

Machine learning, especially in artifi cial neural networks, is being widely used in various applica-tions to solve nonlinear optimization problems. The machine-learning- based approach mainly employs reinforcement learning algorithms such as Q-Learning and DQN to provide an end-to-end solution to the deployment problems [9, 10]. Because the service deployment problem needs to explore and evaluate diff erent service deploy-ment schemes on the edge network, reinforce-ment learning is suitable for finding the optimal action sequence to obtain the optimal solution. Q-Learning is simple to implement and easy to converge, but the dimension explosion problem makes it not easy to adapt to applications with large state space. Deep reinforcement learning (e.g., DQN) uses a deep neural network as the Q function approximator to support large state space. Although some research proposes to use reinforcement learning to solve the service deployment problem, the user request pattern and locality is mostly not considered.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Central | Cloud |  |  |  |
| Enough Resource |  |  |  |  |  |
| All Services | … … |  |  |  |  |
|  | Internet | |  |  |  |
| Traﬃc Prediction & |  | Speech | Recognition | & |  |
| Vehicle Identification |  | Context Awareness | |  |  |
| Edge Clouds |  |  |  |  |  |
| Local Transportation |  | mart Home |  |  |  |
|  |  |  |  |  |
| Data |  |  |  |  |  |
| D2D |  |  |  |  |  |



**FIGURE 1.** Hierarchical 5G mobile edge computing.

problem formulAtIon

In this section, we discuss the architecture of 5G mobile edge computing and illustrate the relation-ship and communication models between diff er-ent elements. The concepts and system model are formulated in order to define the optimization objectives. The optimization goal of our approach is to reduce the overall response time and maxi-mize the number of requests served on the edge server with the resource constraints.

system model

5G mobile edge computing adopts the common hierarchical architecture shown in Fig. 1. The cen-tralized cloud resides in the core network and provides theoretically unlimited computational resources. Most of the complex algorithms, such as machine learning algorithms and offline big data analysis tasks, will be deployed in the cloud. Some of the application services will be duplicated and scheduled to the edge cloud according to the solution proposed in this article. In this architec-ture, the base stations are installed with diff erent densities in different areas to provide better cover-age and quality of service (QoS). Some colocated base stations will form edge clouds that serve the local requests to achieve better performance and provide personalized services. Each base station in the edge cloud also acts as a resource-constrained edge server that employs virtualization technolo-gy, such as containerization, to support the light-weight migration of application services. In some scenarios, such as smart homes, the devices will connect to the smart home gateway using tradi-tional WiFi and then connect to the edge cloud. In other scenarios, such as intelligent transpor-tation, vehicles can connect to the base station

|  |  |
| --- | --- |
| IEEE Wireless Communications • February 2020 | 85 |

Authorized licensed use limited to: University of New South Wales. Downloaded on November 20,2020 at 02:23:53 UTC from IEEE Xplore. Restrictions apply.

The number of end-us-er requests is used to measure communica-tion. For each service, the required four types of resources cannot exceed the available resources on an edge server. Each edge server stores several services, but each type of service has only one copy on an edge server. Multithreading is adopted to process simultaneous requests for the same service.

and simultaneously share data with other vehicles or infrastructure directly using 5G D2D technol-ogy. 5G can also enhance existing technology. For example, IPTV devices that require high-speed networks[11] are supercharged with 5G, which increases the user experience. Mobile users and devices will send requests for services or contents to the connected edge cloud.

Each edge server in the edge cloud has limit-ed computation, memory, storage, and communi-cation resources. The computation resources are measured by CPU performance and utilization. Each service request will attempt to allocate a certain amount of memory at runtime. Similarly, each installed service fills a certain amount of stor-age space, which is different from that of others. Communication capacity is related to the wireless access point[7], which gathers a large number of requests. The number of end-user requests is used to measure communication. For each service, the required four types of resources cannot exceed the available resources on an edge server. Each edge server stores several services, but each type of service has only one copy on an edge server. Multithreading is adopted to process simultaneous requests for the same service.

Data dependency of service is also considered in this work. Services may require a large amount of data from different places when scheduling requests from end users. The dependency of data is diverse and can include data on the edge server where the service is running, data on adjacent or even farther edge servers, or data on the central cloud. The problem with multiple data dependen-cies is that different channels have different band-widths, resulting in different response times for the same service in different locations. The response time of a request mainly includes the execution time of service and the time of data transmission (if the service has any). These two types of time are believed to make up most of the response time. Here, we consider two types of data dependen-cy: edge-cloud data and central-cloud data. The time required for data transmission is related to the size of the data and the bandwidth. Unlike wireless access points between end users and edge servers, dedicated communication links (e.g., optical cable) are used between edge servers, as well as between edge servers and cloud centers. Communication between two edge servers is considered communi-cation between two base stations.

Optimization Objectives

Edge computing brings services closer to the user and the data source so that the latency can be decreased. The optimal service deployment scheme will deploy more services to appropriate edge servers to decrease the overall response time of the local users. Multiple types of constraints need to be considered when finding the optimal service deployment scheme, which makes it a complicated optimization problem. In this article, we consider the resource constraints, data depen-dencies, and user request patterns as explained in the previous subsection. Using conventional approaches introduced earlier for these multi-ob-jective optimization problems will always encoun-ter the huge searching space problem, and it is difficult to design an accurate model to consider all the constraints, especially the user requests

patterns. Therefore, we leverage model-free rein-forcement learning algorithms to train the neural network using the historical user requests to learn the service deployment scheme. To evaluate the performance of a service deployment scheme, we define an evaluation metric called RPSE. For each deployment scheme, all user requests from the training dataset are scheduled to an edge server or the central cloud according to the scheduling policy. The detailed request scheduling policy and algorithm are introduced later. After scheduling all requests, the total number of requests processed on edge servers (i.e., response quantity on edge servers) and the sum of all requests’ response time (i.e., total response time) are calculated for the deployment scheme under evaluation. The RPSE is defined as the result of the response quan-tity on edge servers divided by the total response time. A large RPSE value represents more user requests processed on the edge servers with short response time, which is a feature of a reasonable deployment scheme. The formula to calculate the RPSE is shown as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| *RPSE* = | ∑*req*∈*Req* ∑*c*∈*C yreq*, *c* |  |  |
| ∑*req*∈*Req treq* | (1) |  |

∑*c*∈*C yreq*, *c* ≤1 ∀*req* ∈*Req*

*Req* is used to represent the set of all requestsfrom end users, and *C* represents the set of all edge servers. *yreq*,*c* denotes whether the request *req* will be scheduled on an edge server *c*. *yreq*,*c* can either be 0 or 1. *t* *req* represents the response time for each request. The optimization objective is to find a deployment scheme that can maximize the RPSE.

The response time of a request is the sum of the execution time of the requested service and the waiting time for all dependent data. The time a ser-vice waits for its dependent data is the maximum transmission time for all dependent data blocks. Each service relies on the data from either the edge servers or the central cloud, so the data transmission is between two edge servers or between an edge server and the central cloud. For each data block, the data transmission time is calculated according to the data block size and the available bandwidth. The data transmission will not be considered if the required data block is on the local edge server, or the required data block and the service are both deployed inside the central cloud.

Modeling the Problem as a Markov Decision Process

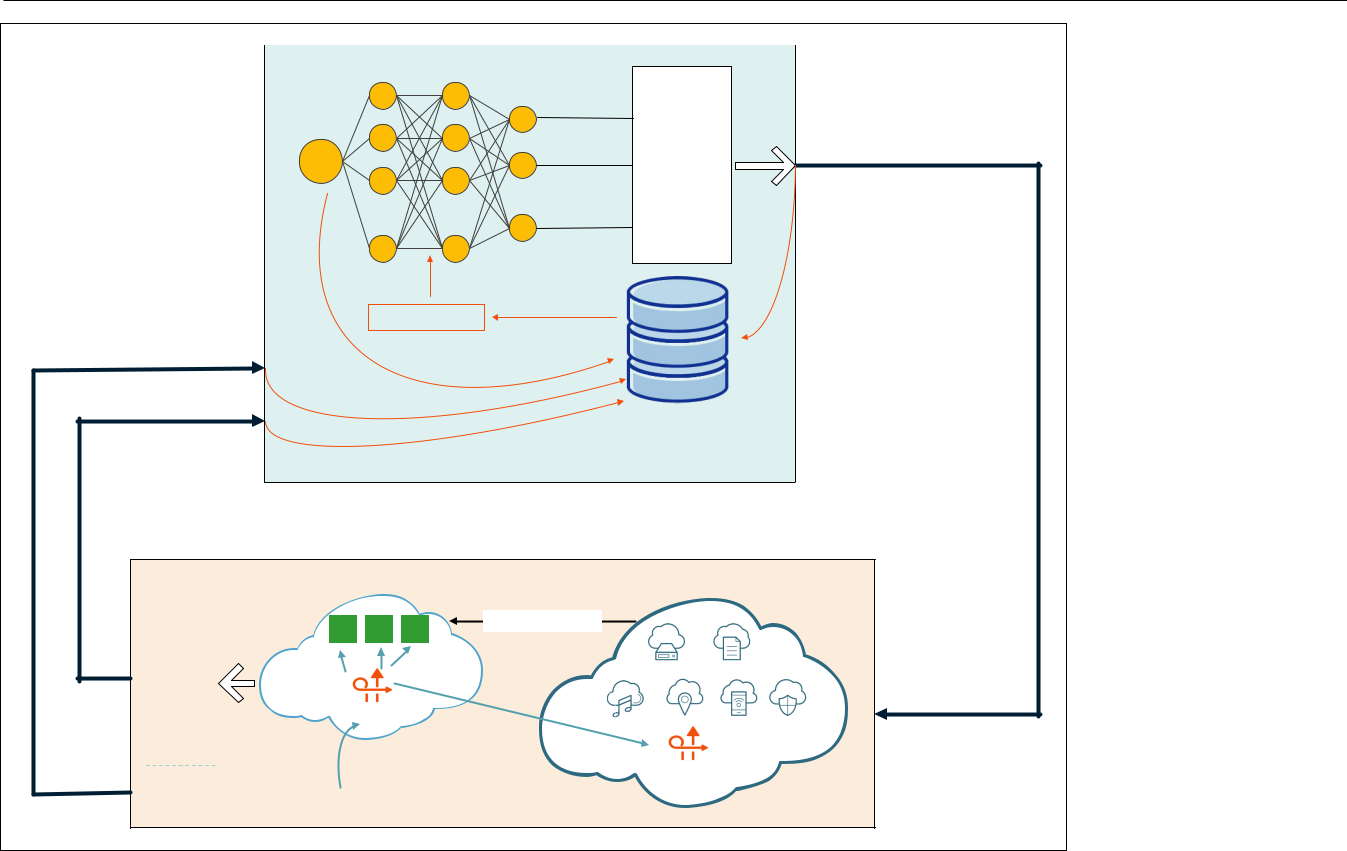
In order to establish the reinforcement learning environment, we first model the service deploy-ment problem as a Markov decision process (MDP) [12]. An MDP consists of a quaternion: *M*

* (*S*, *A*, *P*, *R*), where *S* is a finite set of states, *A* is a set of actions, *P* is the probability for state tran-sition by taking some action, and *R* is the reward function to measure the action. We formulate our problem using MDP to achieve mathematical rep-resentation of our decision making process to aid the deep reinforcement learning process.

In this work, *S* is defined as the set of all pos-sible service deployment schemes, and *s* ∈ *S* rep-resents one service deployment scheme, which is the adjacency matrix between services and edge servers such as *sj* and *sj*+1, shown in Fig. 2.

86 IEEE Wireless Communications • February 2020

Authorized licensed use limited to: University of New South Wales. Downloaded on November 20,2020 at 02:23:53 UTC from IEEE Xplore. Restrictions apply.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Agent |  | Dueling - DQN | |  |  |  |
|  |  |  |  |  |
|  |  |  |  | Q(sj,a1) | Action: Deploy/Remove a |  |
|  |  |  |  |  |  |
| sj |  |  |  | Q(sj,a2) | service on edge server |  |
|  |  |  |  | MAX |  |
|  |  |  |  | ... |  |  |
|  |  | ... | ... | Q(sj,an) |  |  |
|  |  | si, ai, ri, si+ 1 | |  |  |  |
| Observation: sj+ 1 |  |  |  |  |  |  |
|  |  |  |  |  | Experience |  |
|  |  |  |  |  | Replay Buﬀer |  |
| Reward : rj= RPSE j- RPSE j- 1 |  |  |  |  |  |  |
| Environment | Edge Cloud | | |  | Central Cloud |  |
| RPSE 1 | a | b | c | Service Replica |  |  |
|  |  |  |  |  |  |
| RPSE 2 |  |  |  |  |  |  |
| RPSE 3 | Scheduler | |  |  |  |  |
| …… |  |  |  |  |
|  |  |  |  |  |  |
| RPSE n |  |  |  |  |  |  |
|  |  |  |  |  | Scheduler |  |
|  |  | Requests : r1, r2, r3, … rn | | |  |  |

**FIGURE 2.** Service deployment with Dueling-DQN and request scheduling.

Each request in the training set has several attributes including the location of a user who sends the request, the requested service, the

starting and ending timestamp, and so on.

In reality, there is no centralized scheduler for all base stations. The scheduling rules are distributively deployed in the base stations.

An action *a* ∈ *A* refers to deploying or removing a service from an edge server. *P* represents the probability of state transition. For instance, if an action *a* (e.g., deploy a service to an edge serv-er) is taken in state *s*, the probability of transition to state *s*’ (a new deployment scheme) can be expressed as *p*(*s*’| *s*, *a*). The reward function *R* is a measure of how well an action is performed in a given state. *R* is calculated using the difference in RPSE before and after an action is taken. The value function represents the long-term value of a state or behavior. The value function of a state in a Markov reward process is the expectation of the Markov chain beginning from that state. The services deployment in mobile edge computing as an MDP is carried out according to the following general process. Assuming the initial state is a ran-dom service deployment scheme, the probability of state transition is considered, and one action from all possible actions is selected to execute. The action may be to remove or deploy a service on an edge server. A new service deployment scheme is obtained after the action. The scheduler will sched-ule all requests according to the scheduling policy and constraints. The new RPSE value will be cal-culated durning the execution of all requests. The objective is for the given initial service deployment scheme to select the best sequence of actions to obtain the maximum cumulative reward value.

leArnIng servIce deployment from

user reQuests

The general process of training the neural net-work is illustrated in Fig. 2. All services are initial-ly deployed in the cloud center. If the action of

deploying one service to an edge server is taken, the service will be replicated to the chosen edge server. All user requests from the training dataset are stored in request arrays in the order of their request timestamp. The scheduler will schedule the request to a nearby edge server that deployed the required service. If the edge server cannot process the request because of resource con-straints, it will redirect the request to the cloud center. The deep neural network will learn from the input to generate an action to either remove the service from the edge server or deploy the service on the edge server. Next, the reward sec-tion will evaluate the deployment of services by scheduling all requests again to see whether it is a positive action or a negative action. The result will be saved into an experience buff er for future training.

reQuest schedulIng

Each request in the training set has several attri-butes including the location of a user who sends the request, the requested service, the starting and ending timestamp, and so on. In reality, there is no centralized scheduler for all base stations. The scheduling rules are distributively deployed in the base stations. To simulate the scheduling pro-cess, we define a scheduler using a first come first served (FCFS) strategy to dispatch the requests to the edge server or cloud center. When the sched-uler receives a request, it will determine whether the requested service exists in the edge server or not. If the requested service does not exist, the request will be redirected to the central cloud for processing. If the service is already deployed in the edge server, the scheduler will check the

|  |  |
| --- | --- |
| IEEE Wireless Communications • February 2020 | 87 |

Authorized licensed use limited to: University of New South Wales. Downloaded on November 20,2020 at 02:23:53 UTC from IEEE Xplore. Restrictions apply.



1. Initialize experience replay buffer and ANN~
2. for *each episode in range*(*M*) do
3. Initialize state *s*1;
4. for *each t in range*(*T*) do
5. Deploy services according to state *st*;
6. Schedule the requests to get RPSE *x*1;
7. Select an action *at* with probability e;
8. Execute action *at* to get new state *st*+1;
9. Re-deploy services according to state *st*+1;
10. Declare reward *rt*;
11. if *st* equal to *st*+1 then
12. Assign *rt* to a negative value;
13. else
14. Schedule the requests to get RPSE *x*2;
15. Calculate the reward value:
16. *rt* = *x*2– *x*1;
17. Store (*st*, *at*, *rt*, *st*+1) in the replay buffer;
18. Sample a batch of transitions (*si*, *ai*, *ri*, *si*+1) from the experience replay buffer;
19. Calculate the target Q-value;
20. The gradient descent method is used to update the Q-network;
21. Each K-step updates the target Q-network;

**ALGORITHM 1.** Service deployment.

resource constraints, which include computation, memory, and network resources. If the available resources on the edge server are insufficient, the request will be redirected to the central cloud. Latency must also be considered. Sometimes, pro-cessing the request on the edge server results in higher latency because of the data dependency and limited computation power. The requested service may need data from other edge servers or the central cloud, so the data transmission time is part of the service response time. The schedul-er calculates the total response time in different cases and selects a suitable location to process the request.

Service Deployment with Dueling-DQN

A large number of users’ requests are often in some implicit access patterns, such as different users’ preferences for different services, or access frequency of different services in different time periods. Leveraging these access patterns when deploying the services to the edge of the network could achieve better QoS for local requests. But the patterns are difficult to accurately describe using mathematical models. Therefore, we choose an end-to-end reinforcement learning algorithm to learn policies directly from user requests and mobile edge computing environments. DQN [13] is a model-free reinforcement learning algorithm to select the best action sequence according to the Q-value in a Markov environment. The Q- val-ue is defined as the accumulated rewards that will be received if the current action is taken following the policy. More specifically, Dueling-DQN [13], which is an extension of DQN, is employed in this work to learn the pattern to guide the deploy-ment of services on edge servers. The structure of the neural network contains two fully con-nected hidden layers and two streams. The two streams are used to estimate the state value and the advantage for each action, respectively. The service deployment scheme is represented as an adjacency matrix of the services and edge servers. Every element in the matrix can have a value of 1 or 0, which represents whether the service is

deployed on the edge server or not. Every ele-ment in the matrix is taken as an input neuron of the neural network. The output layer has the same structure as the input layer, and each output neuron corresponds to a type of service on one of the edge servers. The output of each neuron is the Q-value of taking an action in current state. If the corresponding service of a neuron has been deployed, the action is service removal; other-wise, the action is not service removal.

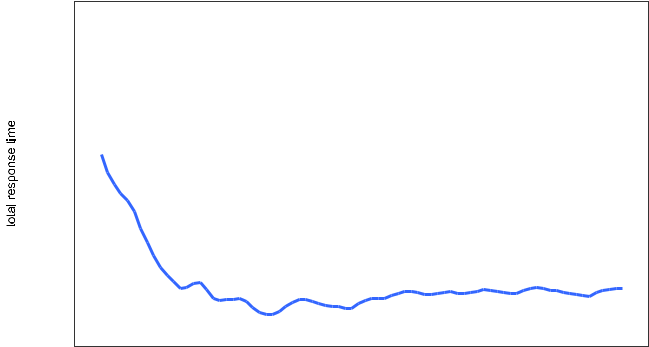
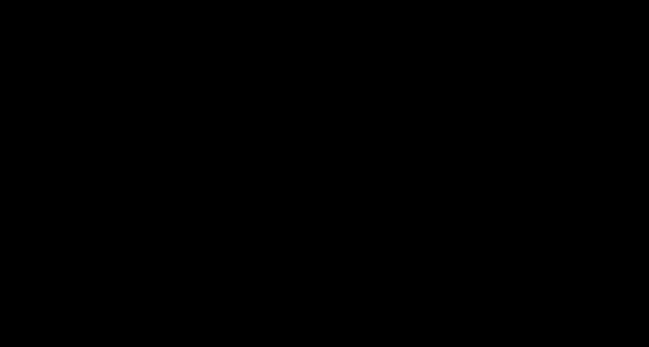
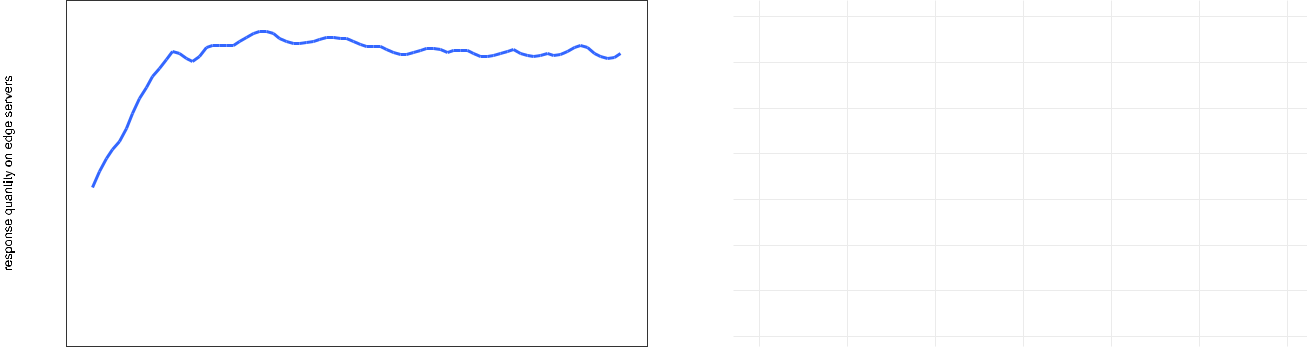
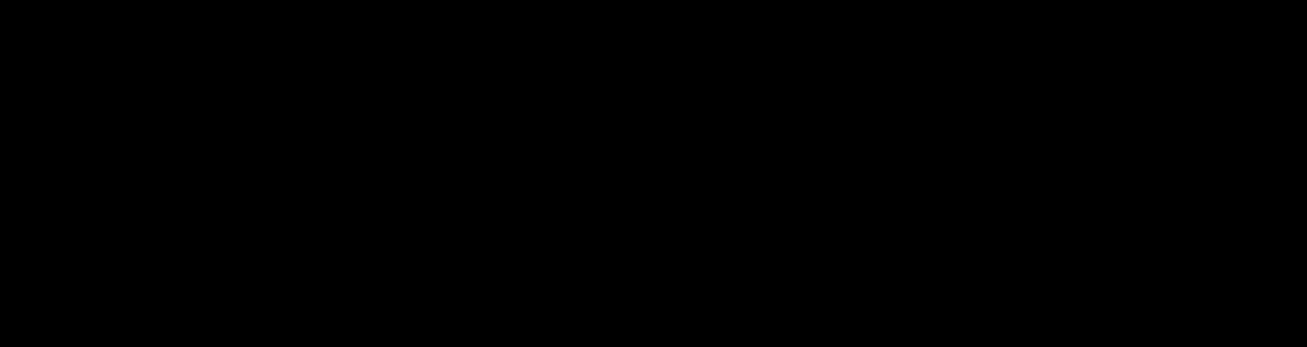
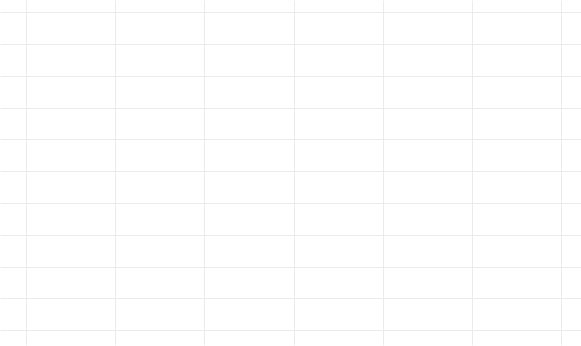
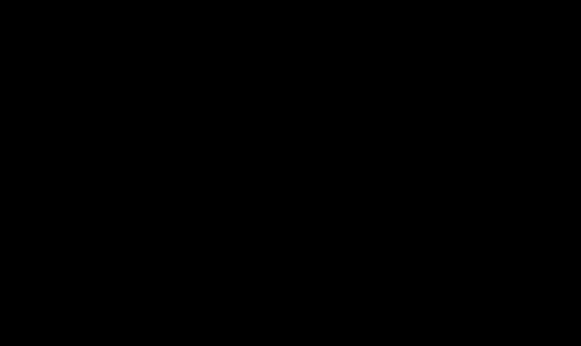
An experience replay buffer is established in Dueling-DQN to break the correlation and ensure the data is independent and identically distributed. We use a uniform random sampling method to extract data from the experience replay buffer and use the extracted data to train the neural network. Previous studies show that using the same neu-ral network to calculate the current Q-value and the target Q-value is not conducive to the conver-gence of the DQN algorithm. Therefore, we use two neural networks to generate the current Q-val-ue and the target Q-value, respectively. The net-work that generates the current Q-value is updated at each learning step, but the network used to calculate the target Q-value is updated every sev-eral learning steps. To avoid the local optimal solu-tion, an e-greedy policy is introduced to select a random action with a probability to ensure some exploration. During the neural network training process, it is important to cope with invalid actions since not all actions can successfully proceed. For example, the action to deploy a new service on an edge server without enough storage space is an invalid action. We assign a negative reward to be opposed to an invalid action. The reward is relat-ed to the scheduling results of all requests. Every time an action is executed, all user requests must be rescheduled. The time spent on each service invocation and the location where each service is processed are recorded, and RPSE is calculated. The reward value is the difference between two RPSE values corresponding to the states before and after an action.

The service deployment is described in Algo-rithm 1. The neural networks, data structure, and service deployment environment are first initial-ized. In each training step of an episode, services are first deployed on edge servers according to the current state *st*. The request scheduling algo-rithm is executed in line 8 to determine the value of RPSE *x*1. Then the algorithm randomly selects an action *at* with the probability of e or chooses action *at* with the largest Q- value and obtains the nextstate *st*+1. The services are redeployed according to the new state *st*+1. Here, the validity of the action must be verified. If the action is invalid, the algo-rithm simply sets the new state *st*+1 equal to *st* and sets the reward *rt* with a negative value to avoid the model selecting this invalid action again. If the action is valid, the request scheduling algorithm is executed again to calculate new RPSE *x*2 . Reward *rt* is the difference between *x*2and *x*1. In addition,a batch of transitions from the experience replay buffer is sampled to train the neural network, and gradient descent is used to update the neural net-work. In this article, each episode has *T* steps, and we set step *T* + 1 as the terminating step. When the neural network converges to an asymptotically stable point, we can either choose the generated state as the optimal service deployment scheme

88 IEEE Wireless Communications • February 2020

Authorized licensed use limited to: University of New South Wales. Downloaded on November 20,2020 at 02:23:53 UTC from IEEE Xplore. Restrictions apply.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5000 |  |  |  | 60000 |  |  |  |  |
| 4000 |  |  |  |  |  |  |  |  |
|  |  |  |  | 55000 |  |  |  |  |
| 3000 |  |  |  |  |  |  |  |  |
| 2000 |  |  |  | 50000 |  |  |  |  |
| 100 |  |  |  | 45000 |  |  |  |  |
|  |  |  |  |  |  |  |  |
| 0 |  |  |  |  |  |  |  |  |
| 0 | 1000 | 2000 | 3000 | 0 | 1000 | 2000 | 3000 |  |
|  | episode (Learning Rate:0.001) | |  |  | episode (Learning Rate:0.001) | |  |  |
|  |  | (a) |  |  |  | (b) |  |  |



**FIGURE 3.** Convergence performance with the increase of steps.

or preserve the structure and parameters of the neural network as the deployment decision model.

Performance Evaluation

To evaluate our approach, we use a real dataset to simulate user requests. Other common service deployment algorithms, including Q-learning, Least Recently Used (LRU), and Least Frequently Used (LFU), are also implemented. The total response time and response quantity on edge servers for dif-ferent algorithms are compared. The experiment results show that our algorithm is superior to other algorithms. Our algorithm can obviously reduce the total response time and improve the number of requests served on edge servers.

Experiment Setup

The dataset used to train and test the model is provided by Shanghai Telecom and contains 7.2 million network access records of 9481 mobile phones using 3233 base stations [15]. We select network access records from 10 base stations in an area from June 1 to June 10 as the training set, whereas the records from June 11 to June 20 constitute the testing set. The start time of a record represents the arrival time of a request from the wireless end user; the location of the base station is used to represent the edge serv-er location, and the number of users visiting a base station is the request quantity on an edge server. We choose six types of services request-ed in the dataset including path planning service, weather report service, location service, image classification service, and two more. Each type of service will consume a certain amount of resourc-es including memory, storage, and CPU time. We also specify the bandwidth among edge servers and the central cloud according to typical 5G networks. We randomly select several services and generate data blocks as the dependencies of these services. The data blocks are placed on dif-ferent edge servers. The data transfer time is cal-culated dynamically according to the location of the request and the remaining bandwidth. There-fore, the response time of service mainly includes the round-trip delay, the service execution time, and the transfer time of the dependent data block. The execution time of each service is spec-ified along with the service type. We simplify the environment by setting some thresholds for some

resources to blunt the impact of resource sched-uling on the response time. Specifically, when the CPU utilization is high in a time-sharing multipro-gramming system, the waiting time of the process will increase because of the process scheduling. Accordingly, we set a threshold of 80 percent for the CPU utilization on each edge server. If the CPU utilization exceeds the threshold, no new requests can be scheduled to the edge server. A similar situation exists in memory consumption. A threshold of 80 percent for used memory is set up to reduce swapping. The communication capacity of each edge server is expressed by the number of requests that can be accepted at the same time. The selected 10 base stations act as the edge servers in the experiments. In order to evaluate the performance of our approach, we compare three commonly used deployment algorithms: random algorithm, LRU, and LFU. A Q-learning-based deployment algorithm is also implemented and compared in the experiments.

Results

We evaluate the response quantity on edge servers and the total response time during train-ing and testing phases. As shown in Fig. 3, with the increase of training episodes, the response quantity on edge servers increases gradually, and the total response time decreases rapidly. After training for many episodes, both values basically converge to an asymptotically stable range. This shows that with the formulated system model, the designed Dueling-DQN network can converge to an optimal policy. At this time, training can be stopped, and the model can be tested using the testing dataset.

We compared the response quantity on edge servers and total response time using different algorithms, as shown in Fig. 4. Compared to other algorithms, using reinforcement learning algorithms (including Q-learning and Dueling-DQN) can sig-nificantly increase response quantity on edge serv-ers and reduce total response time. In contrast to the Q-learning algorithm, the Dueling-DQN algo-rithm can improve response quantity on edge serv-ers by approximately 13.4 percent and reduce total response time by approximately 5 percent.

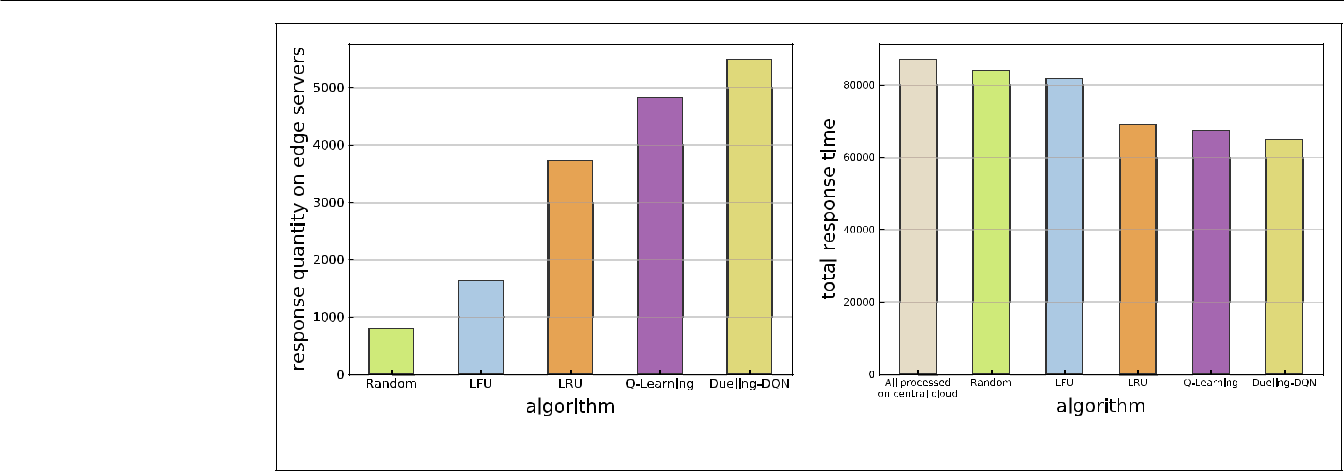
To test the performance of our algorithm, we run the algorithm on the testing dataset (data records from June 11 to June 20) to measure

|  |  |
| --- | --- |
| IEEE Wireless Communications • February 2020 | 89 |

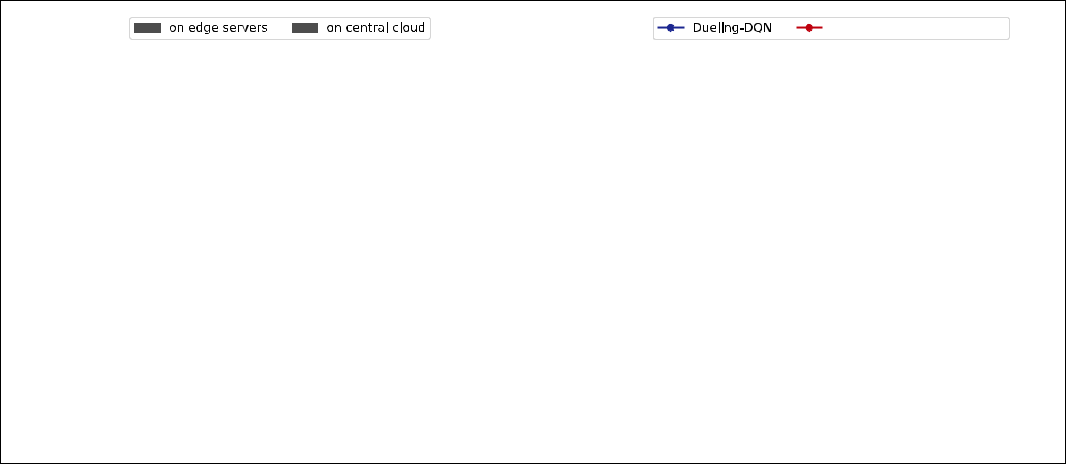
Authorized licensed use limited to: University of New South Wales. Downloaded on November 20,2020 at 02:23:53 UTC from IEEE Xplore. Restrictions apply.

For future work, we will consider the col-laboration of services to our optimization objectives. Because multiple services often work together to com-plete a request in the real world. The conver-gence speed and stabil-ity of model can also be improved, hence diﬀ erent network mod-els like Asynchronous Advantage Actor-Critic would be evaluated in the system.

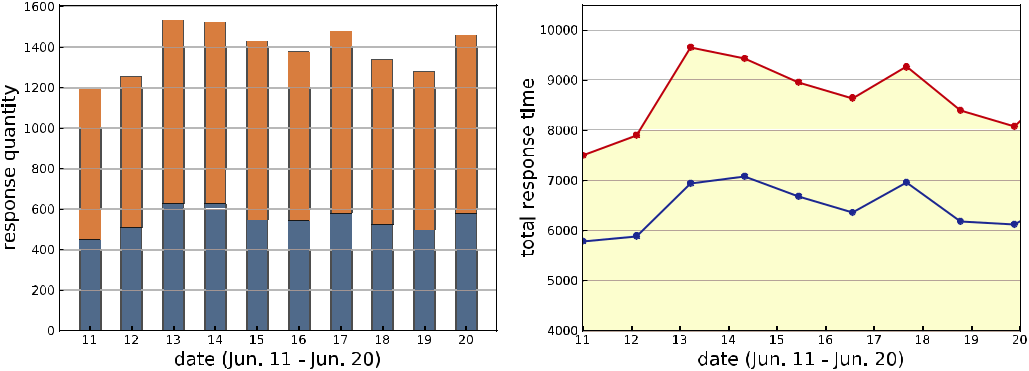
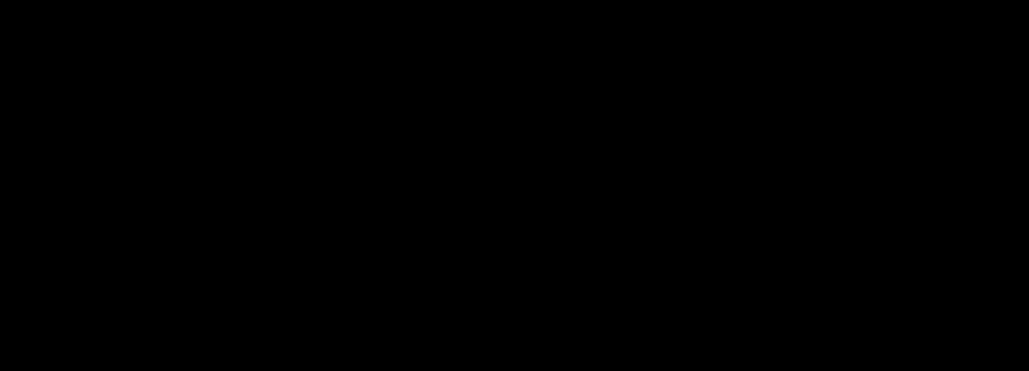
(a) (b)



**FIGURE 4.** Comparison of requests on edge servers and total response time with different algorithms.



on central cloud



(a) (b)

**FIGURE 5.** Response quantity and total response time.

the response quantity and total response time as shown in Fig. 5. In Fig. 5a, the response quantity, which means the number of requests processed on edge servers or on the central cloud, is mea-sured. An average of 40 percent of requests are handled on the edge servers with the deployment schema learned from the Dueling-DQN model. The total response time of all requests on each day is compared in Fig. 5b with a baseline of no services deployed on the edge servers. The total response time fluctuates slightly because of dif-ferent numbers of requests on different days. As shown in this fi gure, the total response time is sig-nifi cantly reduced compared to the baseline when the service deployment scheme generated by the Dueling-DQN algorithm is used.

conclusIons And future work

In this article, we study the service deployment in 5G mobile edge computing. The objectives are to improve the number of services executed on the edge servers and to reduce the total response time. The deployment of services on edge servers is formulated as an MDP problem, and a Duel-ing-DQN based algorithm is designed to learn the access patterns of a large number of requests on edge servers with the result of request scheduling to guide the deployment process. The experimen-tal results show that our algorithm can signifi cant-ly improve the response quantity on edge servers and reduce the total response time. Compared to

different algorithms, we find that learning-based algorithms achieve better performance than con-ventional algorithms, and the Dueling-DQN-based algorithm is superior to the Q-learning -based deployment approach. Training the model is time consuming for a large edge network, but our solu-tion focuses on learning the user request pattern in a local area. Diff erent neural networks should be trained for different areas. Thus, the number of users and base stations will not be very large for training. For future work, we will consider the collaboration of services toward our optimization objectives. Multiple services often work together to complete a request in the real world. The con-vergence speed and stability of the model can also be improved; hence, diff erent network mod-els like asynchronous advantage actor-critic (A3C) will be evaluated in the system.

Acknowledgment

This work is supported by the National Nature Sci-ence Foundation of China (Grant No. 61602037) and the Equipment Pre-Research Field Foundation (Grant No. 61400010104).

references

1. M. Chen *et al*., “Data-Driven Computing and Caching in 5G Networks: Architecture and Delay Analysis,” *IEEE Wireless* *Commun*., vol. 25, no. 1, Feb. 2018, pp. 70–75.
2. T. Ouyang, Z. Zhou, and X. Chen, “Follow Me at the Edge: Mobility-Aware Dynamic Service Placement for Mobile Edge Computing,” *IEEE JSAC*, vol. 36, no. 10, 2018, pp. 2333–45.

90 IEEE Wireless Communications • February 2020

Authorized licensed use limited to: University of New South Wales. Downloaded on November 20,2020 at 02:23:53 UTC from IEEE Xplore. Restrictions apply.

1. X. Du *et al*., “Transactions Papers: A Routing-Driven Elliptic Curve Cryptography Based Key Management Scheme for Heterogeneous Sensor Networks,” *IEEE Trans. Wireless Com-mun*., vol. 8, no. 3, 2009, pp. 1223–29.
2. H. Badri *et al.*, “A Sample Average Approximation-Based Par-allel Algorithm for Application Placement in Edge Comput-ing Systems,” *Proc. 2018 IEEE Int’l. Conf. Cloud Engineering*, 2018, pp. 198–203.
3. G. Premsankar *et al*., “Efficient Placement of Edge Comput-ing Devices for Vehicular Applications in Smart Cities,” *Proc.* *NOMS 2018*, Apr. 2018, pp. 1–9.
4. F. Faticanti *et al*., “Cutting Throughput on the Edge:App-Aware Placement in Fog Computing,” arXiv:1810.04442 [cs], 2018; http://arxiv.org/abs/1810.04442.
5. T. He *et al*., “It’s Hard to Share: Joint Service Placement and Request Scheduling in Edge Clouds with Sharable and Non-Sharable Resources,” *Proc. 2018 IEEE 38th Int’l. Conf.* *Distributed Computing Systems*, 2018, pp. 365–75.
6. O. Skarlat et al., “Towards QoS-Aware Fog Service Place-ment,” *Proc. IEEE 1st Int’l. Conf. Fog and Edge Computing*, May 2017, pp. 89–96.
7. Z. Zhang *et al*., “Q-Placement: Reinforcement-Learning-Based Service Placement in Software-Defined Networks,” *Proc.* *2018 IEEE 38th Int’l. Conf. Distributed Computing Systems,*

2018, pp. 1527–32.

1. Y. He *et al*., “Software-Defined Networks with Mobile Edge Computing and Caching for Smart Cities: A Big Data Deep Reinforcement Learning Approach,” *IEEE Commun. Mag.,* vol. 55, no. 12, Dec. 2017, pp. 31–37.
2. Y. Xiao *et al.*, “Internet Protocol Television (IPTV): The Killer Application for the Next-Generation Internet,” *IEEE Com-mun. Mag*., vol. 55, no. 11, Nov. 2007, pp. 126–34.
3. M. A. Alsheikh *et al*., “Markov Decision Processes With Applications in Wireless Sensor Networks: A Survey,” *IEEE* *Commun. Surveys & Tutorials*, vol. 17, no. 3, 2015, pp.1239–67.
4. V. Mnih *et al*., “Playing Atari with Deep Reinforcement Learning,” arXiv:1312.5602 [cs], 2013; http://arxiv.org/ abs/1312.5602.
5. Z. Wang *et al*., “Dueling Network Architectures for Deep Reinforcement Learning,” *Proc. Int’l. Conf. Machine Learning*, 2016, pp. 1995–2003; http://proceedings.mlr.press/v48/ wangf16.html.
6. S. Wang *et al.*, “Edge Server Placement in Mobile Edge Computing,” *J. Parallel and Distributed Computing*, vol. 127, 2019, pp. 160–68; http://www.sciencedirect.com/science/ article/pii/S0743731518304398.

Biographies

Yanlong Zhai received his B.Eng. degree and Ph.D. degree in computer science from Beijing Institute of Technology, China, in 2004 and 2010. He is an assistant professor in the School of Computer Science, Beijing Institute of Technology. He was a visiting scholar in the Department of Electrical Engineering and

Computer Science, University of California, Irvine. His research interests include cloud computing, big data, and edge computing.

Tianhong Bao joined Beijing Institute of Technology in 2018 as a graduate student. His research interests include edge comput-ing and machine learning.

Liehuang Zhu received his B.Eng. and Master’s degrees in com-puter application from Wuhan University, Hubei, China, in 1998 and 2001, respectively. He received his Ph.D. degree in computer application from Beijing Institute of Technology in 2004. He is cur-rently a professor in the Department of Computer Science, Beijing Institute of Technology. He has been selected into the Program for New Century Excellent Talents in University from Ministry of Edu-cation, China. His research interests include the Internet of Things, cloud computing security, the Internet, and mobile security.

Meng Shen [M] received his B.Eng. degree from Shandong University, Jinan, China, in 2009, and his Ph.D degree from Tsinghua University, Beijing, China, in 2014, both in computer science. Currently he serves at Beijing Institute of Technology as an associate professor. His research interests include priva-cy protection for cloud and IoT, blockchain applications, and encrypted traffic classification. He received the Best Paper Run-ner-Up Award at IEEE IPCCC 2014.

Xiaojiang Du received his B.S. and M.S. degrees in electrical engineering from Tsinghua University in 1996 and 1998, respec-tively, and M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park in 2002 and 2003, respectively. He was an assistant professor in the Department of Computer Science, North Dakota State University, between August 2004 and July 2009. He is currently a professor in the Department of Computer and Information Sciences, Temple University, Philadelphia, Pennsylvania. He is a Life Member of the ACM. He received the Excellence in Research Award at North Dakota State University in May 2009. His research inter-ests include security, wireless networks, computer networks, and systems. He has published more than 200 journal and confer-ence papers in these areas.

Mohsen Guizani received his B.S. (with distinction) and M.S. degrees in electrical engineering and M.S. and Ph.D. degrees in computer engineering from Syracuse University, New York, in 1984, 1986, 1987, and 1990, respectively. He was an associate vice president at Qatar University, the chair of the Computer Science Department, Western Michigan University, the chair in the Computer Science Department, University of West Florida, and the director of graduate studies at the University of Missou-ri-Columbia. He is currently a professor in the Department of Computer Science and Engineering, Qatar University. He is the author/coauthor of nine books and more than 450 publications in refereed journals and conferences. His research interests include wireless communications and mobile computing, smart grid, cloud computing, and security.

|  |  |
| --- | --- |
| IEEE Wireless Communications • February 2020 | 91 |

Authorized licensed use limited to: University of New South Wales. Downloaded on November 20,2020 at 02:23:53 UTC from IEEE Xplore. Restrictions apply.