Distributed Multi-Cloud Multi-Access Edge Computing by Multi-Agent Reinforcement Learning

Yutong Zhang, *Graduate Student Member, IEEE*, Boya Di, *Member, IEEE*, Zijie Zheng, *Member, IEEE*, Jinlong Lin, and Lingyang Song, *Fellow, IEEE*

***Abstract*— In this paper, we consider a three-layer distributed multi-access edge computing (MEC) network where multiple clouds, MEC servers, and edge devices (EDs) are deployed at the top layer, middle layer, and bottom layer, respectively. Each cloud center (CC) is associated with an independent service provider and publishes an application-driven computing task. To deliver the tasks, CCs rely on EDs to generate the raw data and offload part of the computing tasks to both EDs and MEC servers such that their computing and transmission resources can be fully utilized to reduce the system latency. However, in such a three-layer network, the distributed deployment of tasks leads to inevitable resource competition among CCs. To address this issue, we propose a distributed scheme based on multi-agent reinforcement learning, where each CC jointly determines the task offloading and resource allocation strategy based on its inference of other CCs' decisions. Simulation results indicate that a lower system latency is achieved via our proposed scheme compared with the existing schemes. In addition, the influence of the number of CCs, MEC servers, and EDs on latency performance is also discussed.**

***Index Terms*— Distributed, multi-access edge computing, multi-cloud, multi-agent reinforcement learning.**

I. Introduction

With the development of mobile communication networks, a number of latency-sensitive applications have emerged in the fields of transportation [1], healthcare [2], and smart environment [3], etc. These applications are generally processed in the remote cloud centers (CCs), such as Amazon EC2, which is called cloud computing [4]–[6]. To deliver the application-specific tasks, CCs require a large number of edge devices (EDs) to generate and upload numerous raw data. For example, in vehicle networks, EDs collect data about road information and upload to a remote CC, thereby constructing a detailed map of the real-time traffic conditions on road congestion and lane closures ahead [7]. However, an increasing number of mobile applications have posed great pressure on CCs and result in excessively high computing and transmission latency [8]–[10]. To cope with the issue and support more access modes including wired and wireless access in heterogeneous networks [11], [12], the concept of multi-access edge computing (MEC) has been proposed to enable the computing tasks to be offloaded to MEC servers which are deployed between the EDs and CCs [13]–[15].

In the literature, existing works have studied the task offloading problem either in a two-layer ED-MEC network [16]–[22], or in the three-layer ED-MEC-CC network with a single CC [23]–[25]. In the former, authors in [16]–[19] studied task offloading from EDs to a single MEC server, while MEC networks with multiple MEC servers were considered in [20]–[22]. Specifically, authors in [16] studied partial computation offloading by jointly optimizing the offloading ratio, computational speed, and transmit power for MEC networks with a single MEC server and multiple EDs, then extended the scenario to a multiple MEC servers network. Authors in [20] proposed an optimization framework of offloading from a single ED to multiple MEC servers to minimize both task latency and energy consumption. In [24], a distributed computation offloading strategy was studied for a MEC system based on orthogonal frequency-division multiple access in small cell networks, where the system includes multiple EDs, multiple MEC servers, and a core network. Authors in [25] combined the cloud and edge to construct a 3-layer data flow processing system consisting of multiple EDs, APs, and a single CC, such that the computing and transmission resources of the entire network can be fully utilized. Moreover, to design vehicle charging [26], task offloading [27], [28] scheme for MEC systems under a stochastic and dynamic environment, several initial works have considered reinforcement learning (RL) as a promising technique since it takes the future feedback from the environment into account, thereby achieving a long-term goal [29].

However, two practical issues have not been fully considered in the above works. First, since various service providers

Manuscript received February 4, 2020; revised June 2, 2020 and September 26, 2020; accepted December 3, 2020. Date of publication December 15, 2020; date of current version April 9, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61625101, Grant 61829101, and Grant 61941101; and in part by the NSF under Grant EARS-1839818, Grant CNS-1717454, Grant CNS-1731424, and Grant CNS-1702850. The associate editor coordinating the review of this article and approving it for publication was L. Duan.*(Corresponding author: Lingyang Song.)*

Yutong Zhang, Zijie Zheng, and Lingyang Song are with the Department of Electronics, Peking University, Beijing 100871, China (e-mail: yutongzhang@pku.edu.cn; zijie.zheng@pku.edu.cn; lingyang.song@pku.edu.cn).

Boya Di is with the Department of Computing, Imperial College London, London SW7 2BU, U.K. (e-mail: b.di@imperial.ac.uk).

Jinlong Lin is with the School of Software and Microelectronics, Peking University, Beijing 100871, China (e-mail: linjl@ss.pku.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TWC.2020.3043038.

Digital Object Identifier 10.1109/TWC.2020.3043038

tend to launch their applications on different CCs [30]–[32] instead of just a single CC,[1] it is necessary to investigate the multi-CC case where each CC is associated with an independent service provider. With incomplete information about other CCs, each CC separately determines the task offloading scheme and allocates the resources of the MEC servers and EDs for its application-specific task. Second, a three-layer MEC network should be given attention where not only the cloud but EDs and MEC servers can also perform computing tasks jointly. Different from the traditional cloud computing or a two-layer ED-MEC network, the computing and transmission resources of each device (CC, MEC server, or ED) can be fully utilized, thereby reducing the system latency.

To resolve the above two issues, we consider a *three-layer distributed heterogeneous* MEC (Het-MEC) network with *multiple independent CCs*. Specifically, multiple applications are distributedly deployed on various independent CCs, each of which maps an application to a computing task. Each task demands different amounts of computing and transmission resources, and each device owns different computing and transmission resources. Such heterogeneous nature leads to competition among multiple CCs to achieve the best latency performance. As task publishers, the CCs on the top layer decide the joint task offloading and resource allocation scheme in a distributed manner. On the middle layer of the network, MEC servers deployed at the small-cell base station can undertake part of the assigned tasks owning to their computation capacity comparable with that of a computer server. The EDs on the bottom layer collect data from the environment and can perform task processing locally. Devices between adjacent layers connect with each other through wired or wireless links.

Challenges have arisen in such a three-layer Het-MEC network with multiple CCs. $First$, different from the single-CC system, there is no central controller in the multi-CC case, and each CC aims to minimize its own task latency by occupying as many resources of MEC servers and EDs as possible. Thus, a distributed mechanism for multiple CCs is required to depict the resource competition. $Second$, since each task can be partially processed at different layers, the task offloading among these layers are coupled with each other due to the limited resources throughout the network, leading to the non-trivial mechanism design. $Third$, task offloading, computing and transmission resource allocation are coupled with each other. The task offloading is directly influenced by the resource allocation, and together they determine the task latency.

To address the above challenges, we aim to minimize the system latency by designing a distributed mechanism for multiple clouds to determines the task offloading and resource allocation strategy. Our main contributions can be summarized as below.

1) To the best of our knowledge, we are the first to consider the three-layer Het-MEC networks consisting of multiple independent CCs, MEC servers, and EDs. Each CC publishes a task to the EDs and MEC servers, and competes for their computing and transmission resources. A pipeline-based data flow processing diagram is constructed to depict the task latency.
2) We propose a multi-agent reinforcement learning based joint task offloading and resource allocation algorithm (MARL-JTORA) to let CCs coordinate the competitions in computing and transmission resources, and to minimize the system latency in a distributed manner. Each CC takes other CCs into account by learning their explicit models as stationary distributions over their task offloading and resource allocation decisions.
3) Simulation results indicate that a lower system latency can be achieved by utilizing our proposed scheme compared to cloud computing and local computing schemes. We also evaluate how the numbers of CCs, MEC servers, and EDs influence the system latency.

The rest of this paper is organized as follows. In Section II, we describe the system model of the multi-cloud Het-MEC network. Then, we discuss the resource constraints, and formulate the latency minimization problem in Section III. To solve this problem, the MARL-JTORA algorithm is designed and analyzed in Section IV. The simulation results are given in Section V to evaluate our proposed algorithm. Finally, we draw our conclusions in Section VI.

II. System Model

In this section, we first present a brief introduction of the multi-cloud Het-MEC network, and then we give task latency model, and task processing and transmission models. Besides, the major notations used in this section are summarized in Table I.

A. Scenario Description

As shown in Fig. 1, we consider a multi-cloud Het-MEC network including L independent CCs, N APs, and $M \times N$ EDs such that each AP connects with M EDs. Since each AP is integrated with a MEC server, for ease of presentation, we will refer to the MEC servers and APs interchangeably. The APs connect CCs through wired communication, and each AP can connect with multiple CCs. In contrast, each ED connects with one and only one AP via a wireless link.

Each CC publishes a task which requires raw data generated by the EDs. We assume that all tasks are bit-wise independent, i.e. the data of each task l can be arbitrarily divided into parts to be processed at different nodes (the ED itself, the connected AP, and the CC l). To minimize their own task latency, multiple CCs in the Het-MEC network compete for the limited computing and transmission resources of APs and EDs. Each CC l independently decides 1) how to offload task l to different nodes, i.e., the task offloading strategy, 2) how much computing and transmission resources of each node are allocated to perform task l, i.e., the resource allocation scheme.

After the CCs distributedly decides the task offloading and resource allocation strategy, APs and EDs perform data processing and transmission accordingly. Specifically, for each task l, each ED generates the data, processes part of the task

[1]Due to the difference of service types, storage, computing cost, etc. among the CCs, service providers prefer deploy their applications on different CCs distributedly.

TABLE I
MAJOR NOTATIONS

Notation	Description
l and L	The index and the total number of CCs
n and N	The index and the total number of APs
m and M	The index and the total number of EDs connected with each AP
$l^{n,m}$	The index of sub-task over the ED m - AP n - CC l link
$\lambda_{ED}^{n,m}(l)$	Data generation speed of the sub-task $l^{n,m}$
$x_{AP}^{n,m}(l)$	The percentage of the raw data to be processed at AP n in its total received raw data for the sub-task $l^{n,m}$
$x_{ED}^{n,m}(l)$	The percentage of the total raw data to be processed at ED m locally for the sub-task $l^{n,m}$
$\phi_{AP}^{n,m}(l)$	Wired Transmission resource for the sub-task $l^{n,m}$
$\phi_{ED}^{n,m}(l)$	Wireless Transmission resource for the sub-task $l^{n,m}$
$\theta_{CC}^{n,m}(l)$	Computing resource of CC l for the sub-task $l^{n,m}$
$\theta_{AP}^{n,m}(l)$	Computing resource of AP n for the sub-task $l^{n,m}$
$\theta_{ED}^{n,m}(l)$	Computing resource of ED m for the sub-task $l^{n,m}$
$t_{CC}^{n,m}(l)$	Data processing time at CC l for the sub-task $l^{n,m}$
$t_{AP}^{n,m}(l)$	Data processing time at AP n for the sub-task $l^{n,m}$
$t_{ED}^{n,m}(l)$	Data processing time at ED m for the sub-task $l^{n,m}$
$\tau_{AP}^{n,m}(l)$	Data transmission time from AP n to CC l for the sub-task $l^{n,m}$
$\tau_{ED}^{n,m}(l)$	Data transmission time from ED m to AP n for the sub-task $l^{n,m}$
$T^{n,m}(l)$	Latency of sub-task $l^{n,m}$
$T(l)$	Latency of task l
T	System latency
ρ	Compression ratio

locally, and transmits the processing results together with the remained raw data to the connected AP. The size of data will be compressed after data processing. After receiving from the ED, each AP also performs data processing and transmits both the remained raw data and processing results to CC l. Finally, all processing results of the task are aggregated at CC l.

B. Task Latency Model

Note that the raw data of each task l is generated by multiple EDs. For convenience, we refer to a sub-task of task l as the data generation and processing of one ED. Task l is completed if results of all sub-tasks are aggregated at CC l after processing at each layer. We now define the sub-task latency and the task latency.

For each CC l, how to deal with sub-task $l^{n,m}$ over the ED m - AP n - CC l link is illustrated in Fig.2. After data generation at ED m, the sub-task $l^{n,m}$ traverses five stages to be completed, i.e., data processing at the ED m, data transmission to the AP n, data processing at the AP n, data transmission to the CC l, and data processing at the CC l. Each stage can be divided into multiple time units. At each layer, data processing and transmission proceed simultaneously. Therefore, for the data of sub-task $l^{n,m}$ generated in each time unit, the total processing and transmission time $T_{sum}^{n,m}(l)$ can be given as

$$T_{sum}^{n,m}(l) = \max\{t_{ED}^{n,m}(l), \tau_{ED}^{n,m}(l)\} + \max\{t_{AP}^{n,m}(l), \tau_{AP}^{n,m}(l)\} + t_{CC}^{n,m}(l), \quad (1)$$

where $t_{ED}^{n,m}(l)$, $t_{AP}^{n,m}(l)$, and $t_{CC}^{n,m}(l)$ denote data processing time at ED m, AP n, and CC l, respectively. $\tau_{ED}^{n,m}(l)$ and $\tau_{AP}^{n,m}(l)$ represent the data transmission time from ED m to AP n, and from AP n to CC l, respectively. The detailed expression of each item in (1) will be given in Section II-C.

Next we define the sub-task latency with respect to data generated for sub-task $l^{n,m}$ in all time units. The devices are idle when the data to be processed or transmitted have not arrived. The time to wait for the data is called $idle\ time$. In the high load situation, the sub-task processing can be described as a pipeline. In a pipeline, the latency mainly depends on the longest computing/transmission time of all stages, including the data generation, since no idle time exists in this stage. Let $\lambda_{ED}^{n,m}(l)$ denote the raw data generation speed of sub-task $l^{n,m}$. The duration of this longest stage to deal with the data generated in a time unit can be expressed by

$$T_{max}^{n,m}(l) = \max\{\ \frac{1}{\lambda_{ED}^{n,m}(l)}, t_{ED}^{n,m}(l), \tau_{ED}^{n,m}(l), t_{AP}^{n,m}(l), \tau_{AP}^{n,m}(l), t_{CC}^{n,m}(l)\}. \quad (2)$$

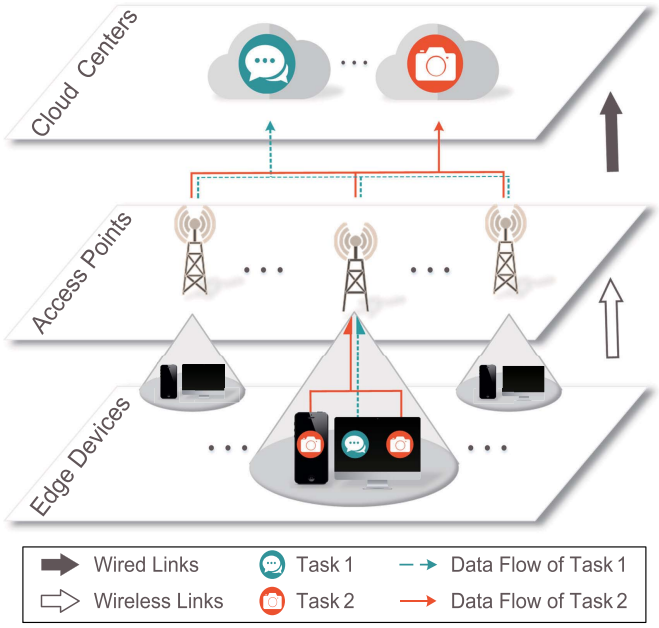


Fig. 1. The architecture of the distributed multi-cloud multi-access edge computing network.

As shown in Fig. 2, for the raw data generated in P time units, the total processing and transmission time of sub-task $l^{n,m}$ equals to $(P-1)\cdot T^{n,m}_{max}(l)+T^{n,m}_{sum}(l)$. The sub-task latency, i.e., the total processing and transmission time per time unit, is close to $T^{n,m}_{max}(l)$ as P approaches infinity, expressed by

$$T^{n,m}(l)=\lim_{P\to\infty}\frac{(P-1)\cdot T^{n,m}_{max}(l)+T^{n,m}_{sum}(l)}{P}=T^{n,m}_{\max}(l). \tag{3}$$

That is to say, for a long time view, the sub-task latency approximately equals to the longest stage latency. Since the all sub-tasks of CC l are processed simultaneously, the latency of task l is defined as the maximum latency of all sub-tasks of task l, expressed by

$$T(l)=\max_{n,m}T^{n,m}(l). \tag{4}$$

C. Task Processing and Transmission

In this part, for a sub-task $l^{n,m}$, we will introduce the models of data processing at the ED m, data transmission to the AP n, data processing at the AP n, data transmission to the CC l, and data processing at the CC l, namely, each item in (1).

1) Data Processing at the ED Layer: The computing resource[2] of ED m connected with AP n to process data of task l per time unit is denoted by $\theta^{n,m}_{ED}(l)$. Thus, the processing time at ED m can be given by

$$t^{n,m}_{ED}(l)=\frac{\lambda^{n,m}_{ED}(l)x^{n,m}_{ED}(l)}{\theta^{n,m}_{ED}(l)}, \tag{5}$$

[2]In this paper, we simply use the computing rate to describe the computing resource, since it can be converted from any other kinds of computing resources [33], such as the number of virtual machines and CPU cycles per second.

where $x^{n,m}_{ED}(l)$ denotes the percentage of the total raw data to be processed at ED m locally.

2) Data Transmission to the AP Layer: The data transmitted from ED m to its connected AP n includes the *remained raw data* and the *processing results* of ED m. We assume $\phi^{n,m}_{ED}(l)$ denotes the wireless transmission capacity for the sub-task $l^{n,m}$ per time unit. Note that the orthogonal resources, e.g., spectrum and time resources, are considered as the wireless transmission resources in this paper which influence the wireless data rate in a linear manner [34]. The transmission time from ED m to AP n can be given by

$$\tau^{n,m}_{ED}(l)=\frac{\rho\lambda^{n,m}_{ED}(l)x^{n,m}_{ED}(l)+\lambda^{n,m}_{ED}(l)(1-x^{n,m}_{ED}(l))}{\phi^{n,m}_{ED}(l)}, \tag{6}$$

where $\rho<1$ is the compression ratio of the data size before and after processing which describes the output characteristics of the whole computation task on average [35].

3) Data Processing at the AP Layer: As shown in (6), the data that AP n receives from ED m per time unit includes the processing results and the remained raw data, expressed by $\rho\lambda^{n,m}_{ED}(l)x^{n,m}_{ED}(l)$ and $\lambda^{n,m}_{ED}(l)(1-x^{n,m}_{ED}(l))$, respectively. Thus, the ratio of the raw data in all arriving data per time unit can be given as $(1-x^{n,m}_{ED}(l))/(1-x^{n,m}_{ED}(l)+\rho x^{n,m}_{ED}(l))$. Therefore, the raw data arriving speed at AP n can be calculated by

$$\lambda^{n,m}_{AP}(l)=\phi^{n,m}_{ED}(l)\cdot\frac{1-x^{n,m}_{ED}(l)}{1-x^{n,m}_{ED}(l)+\rho x^{n,m}_{ED}(l)}. \tag{7}$$

Let $x^{n,m}_{AP}(l)$ denote the percentage of the raw data volume to be processed at AP n in its total received raw data volume. The computing capacity of AP n to process the sub-task $l^{n,m}$ per time unit is denoted by $\theta^{n,m}_{AP}(l)$. Thus, the processing time at AP n can be given as

$$t^{n,m}_{AP}(l)=\frac{\lambda^{n,m}_{AP}(l)x^{n,m}_{AP}(l)}{\theta^{n,m}_{AP}(l)}. \tag{8}$$

4) Data Transmission to the CC Layer: For task l, after processing part of the received raw data, AP n delivers the remained raw data and the processing results of both AP n and ED m to the CC l. The data volume needs to be transmitted can be represented by $\lambda^{n,m}_{AP}(l)(1-x^{n,m}_{AP}(l))+\rho\lambda^{n,m}_{AP}(l)x^{n,m}_{AP}(l)+\beta^{n,m}_{AP}(l)$, where $\beta^{n,m}_{AP}(l)$ is the results processed by ED m, which can be calculated by

$$\beta^{n,m}_{AP}(l)=\phi^{n,m}_{ED}(l)\cdot\frac{\rho x^{n,m}_{ED}(l)}{1-x^{n,m}_{ED}(l)+\rho x^{n,m}_{ED}(l)}. \tag{9}$$

Thus, the transmission time from AP n to the CC layer is represented by

$$\tau^{n,m}_{AP}(l)=\frac{\rho\lambda^{n,m}_{AP}(l)x^{n,m}_{AP}(l)+\lambda^{n,m}_{AP}(l)(1-x^{n,m}_{AP}(l))+\beta^{n,m}_{AP}(l)}{\phi^{n,m}_{AP}(l)}, \tag{10}$$

where $\phi^{n,m}_{AP}(l)$ is the wired transmitting capacity for sub-task $l^{n,m}$ per time unit from AP n to CC l.

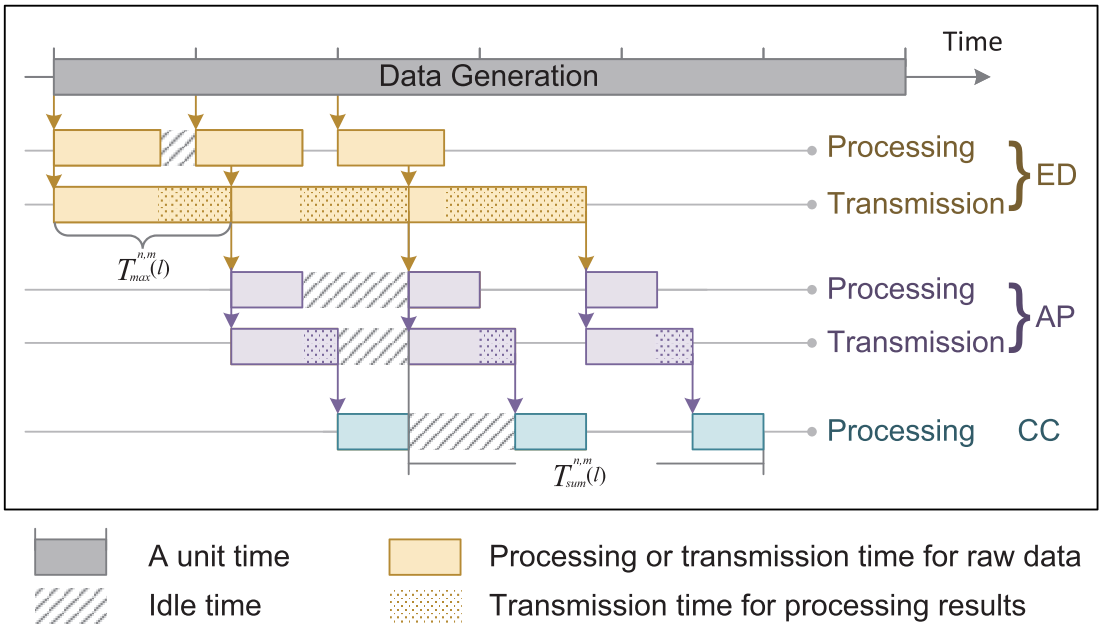


Fig. 2. The pipeline of the processing of data of task l which generated by ED m connected with AP n.

5) Data Processing at the CC Layer: After collecting the data from AP layer, the CC processes the remained raw data. The raw data arriving speed of CC l can be given as

$$\lambda_{CC}^{n,m}(l) = \phi_{AP}^{n,m}(l) \cdot \frac{1 - x_{AP}^{n,m}(l)}{1 - x_{AP}^{n,m}(l) + \rho x_{AP}^{n,m}(l) + \frac{\rho x_{ED}^{n,m}(l)}{1 - x_{ED}^{n,m}(l)}}. \tag{11}$$

Let $\theta_{CC}^{n,m}(l)$ denote the computing capacity of CC l per time unit to process sub-task $l^{n,m}$, then the processing time at CC l can be given as

$$t_{CC}^{n,m}(l) = \frac{\lambda_{CC}^{n,m}(l)}{\theta_{CC}^{n,m}(l)}. \tag{12}$$

III. Problem Formulation

In this section, we first discuss the computing and transmission constraints. Then we formulate the system latency minimization problem for the multi-cloud Het-MEC network.

A. Computing and Transmission Resource Constraints

Each node has its own computing and transmission resources. Note that all APs and EDs can take part in multiple tasks published by different CCs, respectively. Thus, the computing or transmission resources required for all tasks should not exceed the total resources owned by each node. Moreover, the computing/transmission resources allocated for all sub-tasks of task l are less than the total resources of the CC l. We provide the computing and transmission constraints when each node takes part in multiple tasks in this part.

1) Computing Resource Constraints: The total computing capacity of each ED m connected with AP n is denoted by $\theta_{ED}^{n,m}$. Then the computing resources required for all tasks are restricted by total computing resources of ED m, which is given by

$$\sum_{l=1}^{L} \theta_{ED}^{n,m}(l) \le \theta_{ED}^{n,m}. \tag{13}$$

Let θ_{AP}^{n} denote the total computing resources of AP n. Thus, the computing resources of AP n assigned to all tasks satisfy the following constraint.

$$\sum_{l=1}^{L} \sum_{m=1}^{M} \theta_{AP}^{n,m}(l) \le \theta_{AP}^{n}. \tag{14}$$

The computing resources required for all sub-task of task l must be less than the total resources of the CC l, which can be given as

$$\sum_{n=1}^{N} \sum_{m=1}^{M} \theta_{CC}^{n,m}(l) \le \theta_{CC}(l), \tag{15}$$

where $\theta_{CC}(l)$ denotes the total computing resources of CC l.

2) Transmission Resource Constraints: The total wireless transmission data volume of all tasks from EDs connected with AP n is linearly constrained by the wireless transmission resources of AP n, denoted by ϕ_{AP}^{n}. Therefore, we have

$$\sum_{l=1}^{L} \sum_{m=1}^{M} \phi_{ED}^{n,m}(l) \le \phi_{AP}^{n}. \tag{16}$$

The wired transmission resources allocated to all sub-tasks of task l limited by the total wired transmission resources of CC l, which can be represented by

$$\sum_{n=1}^{N} \sum_{m=1}^{M} \phi_{AP}^{n,m}(l) \le \phi_{CC}(l), \tag{17}$$

where $\phi_{CC}(l)$ denotes the total wired transmission resources of CC l.

B. System Latency Minimization

Each CC aims to minimize its task latency by adjusting the joint task offloading and resource allocation scheme. According to the barrel effect, the maximum latency of all tasks determines the system latency. Naturally, the system latency can be given as

$$T = \max_{l} T(l). \tag{18}$$

Therefore, the system latency minimization problem for the entire network can be formulated as follows.

$$\min_{\{x,\theta,\phi\}} \ T, \quad s.t. \ (13) - (17). \tag{19}$$

Considering the distributed deployment of multiple CCs and the close coupling among task offloading and resource allocation of three layers, the latency minimization problem is a non-convex distributed fractional-programming problem, which is generally NP-hard.

As a task publisher, each CC l aims to minimize the task latency by jointly deciding the task offloading strategy, $\boldsymbol{x}(l)$, computing resource allocation scheme, $\boldsymbol{\theta}(l)$, and transmission resource allocation scheme, $\boldsymbol{\phi}(l)$. Since these CCs in the network are independent, each CC attempts to occupy all computing resources of APs and EDs, and all wireless transmission resources, in order to reduce the task latency. Thus, it is necessary to design an efficient distributed mechanism to depict the competition among CCs.

IV. Multi-Agent Reinforcement Learning Algorithm Design

Considering the time variations of system status caused by data durst and unstable wireless environments, it is necessary to optimize the task offloading and resource allocation in an on-line manner. Therefore, we proposed an MARL based algorithm. In this section, we first reformulate the latency minimization problem (19) as an MARL problem. We then design an MARL-JTORA algorithm to let each CC jointly optimize the task offloading and resource allocation in a distributed manner. Finally, we provide theoretical analysis of the proposed algorithm.

A. Learning-Based Problem Formulation

MARL which consists of an environment and multiple agents provides an online algorithm for the CCs to choose the optimal action by continuously interacting with the environment. In each iteration, each agent observes current state of the environment, the systematic trial and error about state transition is indicated by a scalar reinforcement signal. Each agent chooses action that tends to increase the long-run sum of values of the reinforcement signal [36].

For the multi-cloud Het-MEC network, each CC is considered as an agent and everything except the CC is regarded as the environment, i.e., a collective condition related to the task offloading and resource allocation of other CCs. Diverse applications are deployed on each CC one by one. When the application-specific tasks are published by the CCs, each CC makes a sequence of task offloading and resource allocation decisions during multiple iterations for trial and error. In each iteration, each CC observes current state of the environment including the actions selected by other CCs in last iteration and predicts the current action selection of other CCs. Based on the inference of other CCs' actions, the CC chooses action that tends to increase the long-run sum of values of the reward function which indicates the systematic trial and error about state transition. An optimal scheme which can minimize the system latency is obtained when reaching the maximum number of iterations k_m and the current system latency meets the pre-determined requirement, i.e., $\beta = 1$. Note that the time interval between the publication of two tasks is much larger than the duration for obtaining the optimal scheme [37]. We define the state space, action space, and reward function for our Het-MEC network in this part.

1) State Space: A binary variable β is introduced to indicate whether the current system latency meets the pre-determined system latency requirement. Specifically, $\beta = 1$ indicates that the system latency by performing the newly selected action is lower than the maximum tolerable system latency, and vice versa. The state space includes the binary variable β and all possible data generation speeds, expressed by $\mathcal{S} = [\beta, \boldsymbol{\lambda_{ED}}]$.

2) Action Space: To jointly depict the task offloading and resource allocation of the multi-cloud Het-MEC network, we consider each action consisting of two parts, i.e., the task offloading strategy $\boldsymbol{x}$, and resource allocation scheme $[\boldsymbol{\theta}, \boldsymbol{\phi}]$. The corresponding sets of actions available to agents are expressed by $\mathcal{A}_1, \ldots, \mathcal{A}_L$. Each set $\mathcal{A}_l$ consists of all available actions of CC l, i.e., $\mathcal{A}_l = [\mathcal{X}_l, \mathcal{F}_l]$, while $\mathcal{X}_l = [\boldsymbol{x}(l)]$ and $\mathcal{F}_l = [\boldsymbol{\theta}(l), \boldsymbol{\phi}(l)]$.

3) Reward Function: For each iteration, each agent l will get a reward $R(s, a_l)$ in a certain state s after executing the action a_l. The objective of our optimization problem is to minimize the system latency and the goal of MARL is to get the maximum reward, so the value of reward should be negatively correlated to the size of the system latency. We define the immediate reward as $R = \frac{T_{local} - T(s,a_l)}{T_{local}}$ for all the CCs where T_{local} is the system latency when all tasks are executed locally at the ED layer, and $T(s, a_l)$ gives the actual system latency of current state s, as shown in (18).

Hence, the system latency minimization problem can be formulated as a Markov decision process (MDP). Specifically, in each iteration, the CCs observe the environment state and decide the task offloading strategy and resource allocation schemes, which are optimization variables in problem (19). After performing the action, the environment transit from one state to another since the binary variable β may change with the system latency. A reward is fed back to the CCs which is highly correlated with the optimization objective in problem (19) and indicates the quality of the action selection.

B. Joint Action Learners Q-Learning for Multi-Cloud Het-MEC

Joint action learners Q-learning [38], [39] is a classical MARL algorithm, where agents learn the mapping from states to their own actions in conjunction with other agents by integrating of RL with equilibrium (or coordination) learning

methods. Specifically, for each agent, explicit models of the other agents are learned as stationary distributions over their actions. In the rest of this part, we provide the details of the joint action learners Q-learning for the multi-cloud Het-MEC network.

1) Algorithm Initialization: We start with the following initialization operations.

a) State/Action Space Discretization: In each iteration of MARL, each agent requires to derive a desirable action by exploring all possible actions, and finds the best action which can maximize expected discounted reward, and thus, the state/action space is required to be enumerable. Therefore, the data generation speed, resource allocation scheme, and task offloading strategy are discretized such that both the state and action spaces are constructed as discrete sets.

b) Action Space Reduction: To reduce the size of action space $\mathcal{A}$, some impractical actions are eliminated from the action space. Specifically, a sub-task $l^{n,m}$ does not be processed in upper layer repeatedly if all data processed locally, and a device does not allocate any computing resource for the sub-task if there is no data of this sub-task need to be processed at the device. Therefore, an action vector $\boldsymbol{a}$ in the action space is set as null if it contains one of the following states: i) $x_{ED} = 1$ and $x_{AP} \neq 0$; ii) $x_{ED} = 0$ and $\theta_{ED} \neq 0$; iii) $x_{AP} = 0$ and $\theta_{AP} \neq 0$; iv) $x_{AP} = 1$ and $\theta_{CC} \neq 0$.

c) Q-table Initialization and Joint Action Pair Set Reduction: Each agent takes all agents into account during the learning process, thus, possible actions of multiple agents combine with each other and form a joint action pair set. The expected discounted rewards corresponding to each joint action pair are recorded in Q-table. However, a huge amount of joint action pairs are limited by the computing and transmission constraints in problem (19). These pairs make no sense for the learning processing except for increasing the complexity of computation and slowing the convergence speed. Therefore, these joint action pairs are reduced from the set by initiated the corresponding Q-values as $-\infty$ to assure the computing and transmission constraints in problem (19) is obeyed.

In each iteration, we sequentially update sub-region of the Q-table to accelerate convergence. Specifically, starting with a randomly initiated action $a = [\boldsymbol{x}_{ED}, \boldsymbol{x}_{AP}, \boldsymbol{\theta}_{ED}, \boldsymbol{\theta}_{AP}, \boldsymbol{\theta}_{CC}, \boldsymbol{\phi}_{ED}, \boldsymbol{\phi}_{AP}]$, each agent learns to obtain an optimal $\boldsymbol{x}^*_{ED}$ by setting all other items fixed. Specifically, the following process 2) - 4) is performed by each CC l to optimize $\boldsymbol{x}_{ED}$. Given the fixed $\boldsymbol{x}^*_{ED}$, other items are optimized sequentially.

2) ϵ-Greedy Exploration Strategy: At the beginning of each iteration, each CC l selects an action a_l according to the ϵ-greedy exploration strategy [40]. The main purpose of this strategy is to make the trade-off between exploitation and exploration where the former allows the agent to reinforce the evaluation of the actions it already knows to be good, and the latter leads agent to explore new actions. Specifically, the strategy selects a random action with probability ϵ, and the best action a^*_l with probability $1-\epsilon$.

$$a_l = \begin{cases} \forall a_l \in \mathcal{A}, & \text{for } \epsilon \\ a^*_l, & \text{for } 1-\epsilon \end{cases} \tag{20}$$

3) Best Action Selection: The best action a^*_l is the one that can maximize the expected discounted reward R currently, according to the observed frequency distribution of other agents' action in the current state s. Specifically, each agent l keeps a count $\Phi(s, \boldsymbol{a_{-l}})$ on the number of times other agents have selected action $\boldsymbol{a_{-l}}$ in the state s historically. In subsequent iterations, agent l treats the relative frequencies of other agents' actions as indicative of its current action selection. The policy used to obtain the best action a^*_l is denoted by $\pi_l(s)$, expressed by

$$a^*_l = \pi_l(s) = \underset{a_l}{\operatorname{argmax}} \sum_{\sum \boldsymbol{a_{-l}}} \frac{\Phi(s, \boldsymbol{a_{-l}})}{n(s)} Q_l(s, (a_l, \boldsymbol{a_{-l}})), \tag{21}$$

in which $n(s)$ is the total number of times the state s has been visited. And $Q_l(s, (a_l, \boldsymbol{a_{-l}}))$ represents the discounted sum of the immediate reward obtained by taking action a_l at state s.

4) Q-Table Update: After performing the action a_l in the state s, each agent l observes the environment and reward R, then updates the Q-table as follows:

$$Q_l(s, (a_l, \boldsymbol{a_{-l}})) = (1-\alpha) Q_l(s, (a_l, \boldsymbol{a_{-l}})) + \alpha(R + \gamma V_l(s')), \tag{22}$$

where $V_l(s') = \max_{a'_l} \sum_{\boldsymbol{a'_{-l}}} \frac{\Phi(s', \boldsymbol{a'_{-l}})}{n(s')} Q(s', (a'_l, \boldsymbol{a'_{-l}}))$ indicates the maximum expected discounted reward value in the new state s' based on the empirical action profile distribution of other agents. $\gamma \in (0, 1)$ is the discount parameter. If γ tends to 0, then the agent mainly considers the immediate reward; otherwise the agent is also very focused on the future reward.

The learning rate $\alpha \in (0, 1)$ depicts to what extent the new action and the corresponding Q-value replace the current estimate. In this paper, we adopt a dynamic learning rate $\alpha^{(k)} = 0.95^k \cdot \alpha_0$ in the k-th iteration, where the initial learning rate α_0 is an empirical parameter. Learning rate is critical for network convergence, which is analyzed through simulation results which will be provided in Section V.

C. Algorithm Description

The MARL-JTORA of the multi-cloud Het-MEC network can be summarized in Algorithm 1. For each CC l, an environment state and an action are randomly chosen from the state space $\mathcal{S}$ and action space $\mathcal{A}$, respectively. The Q-table of reduced joint action pair set and the policy π_l are also initiated. In each iteration, sub-regions of the Q-table are updated sequentially. With probability ϵ, action a_l is chosen according to the policy $\pi_l(s)$, otherwise, for exploration, a_l is selected randomly from the action space $\mathcal{A}$. After performing the action a_l in current environment, agent l observes the reward R, updates Q-table and the policy according to (22) and (21) for next iteration. The iteration ends by both reaching the maximum number of iterations k_m and meeting the system latency requirement (i.e., b = 1). The latter can avoid trapping in local optimum at a certain extent.

To perform the proposed algorithm in the Het-MEC network, the EDs and APs need to register to the CCs and upload the node information (i.e., the location, the amount

Algorithm 1 MARL-JTORA Algorithm

Input: Learning ratio sequence($\alpha^{(k)} \in (0,1]$); Exploration ratio ($\epsilon > 0$))
Output: Task offloading strategy $\boldsymbol{x}^*$, resources allocation scheme $\boldsymbol{\theta}^*, \boldsymbol{\phi}^*$

Initialize $Q_l(s,(a_l,\boldsymbol{a'_{-l}})), \forall s \in \prod_i^N \mathcal{S}, a_l \in \mathcal{A}_l \boldsymbol{a'_{-l}} \in \prod_{r\neq l}^N \mathcal{A}_r, \pi_l(s,a_l) := \frac{1}{|A|}$;
for *each iteration* $k \leq k_m$ *or* $\beta = 0$*, each CC* l **do**
  for *each item in action space* **do**
    With probability $1-\epsilon^{(k)}$, choose action a_l from the policy $\pi_l(s)$, or with probability $\epsilon^{(k)}$, randomly choose an available action for exploration;
    Perform the action a_l in the k-th iteration;
    Observe the the reward R;
    Update the Q-table according to (22);
    Update the policy $\pi_l(s)$ at state s according to (21), in order to obtain the action that maximizes the expected discounted reward;
    $k \leftarrow k+1$;
  end
end

of computing and transmission resource, the data generation speed). Based on the registration information, the CCs then establish the logical graph of the Het-MEC network. After the CCs obtain the optimal task offloading and resource allocation scheme by performing our MARL-JTORA algorithm, the CCs need to broadcast the scheme to all devices corresponding to the task.

D. Properties of the Multi-Agent Reinforcement Learning Algorithm

We now analyze the action space reduction, the joint action pair set reduction, the complexity, the interchange information, and the scalability of the proposed MARL-JTORA.

1) Action Space Reduction: For brevity, we assume that the L agents have the same action space size, expressed by

$$|\mathcal{A}_1| = |\mathcal{A}_2| = \ldots = |\mathcal{A}_L| = |\mathcal{A}|. \tag{23}$$

Considering the Het-MEC network with multiple clouds in the Section II, we assume that the resource allocation scheme $\mathcal{F} = [\boldsymbol{\theta}, \boldsymbol{\phi}]$ are discretized into a different levels, and the task offloading strategy $\mathcal{X} = [\boldsymbol{x}]$ are discretized into b levels. Before the reduction, the action space size can be calculated by $|\mathcal{A}| = a^{5MN} \cdot b^{2MN}$, when $M \times N$ EDs take part in the task published by each agent. For each ED, 5 resources and 2 offloading percentages are required to be determined, i.e. the computing resources of 3 layers, the wireless and wired transmission resources, task division percentages at ED and AP.

Proposition 1: When $M \cdot N$ EDs generate raw data for each task, the size of reduced action space can be calculated by

$$|A^-| = (\frac{a^2b^2 - 4a^2b + 4a^2 + 3ab - 5a + 2}{a^2b^2})^{MN} \cdot |\mathcal{A}|. \tag{24}$$

Proof: Please see Appendix A. ∎

2) Joint Action Pair Set Reduction: The cardinality of joint action pair set is expressed by

$$\begin{aligned} |\{\boldsymbol{\mathcal{Q}}\}| &= |\mathcal{A}_1 \times \mathcal{A}_2 \times \ldots \times \mathcal{A}_L| \\ &= |\mathcal{A}_1| \cdot |\mathcal{A}_2| \cdot \ldots \cdot |\mathcal{A}_L| \\ &= |\mathcal{A}|^L. \end{aligned} \tag{25}$$

However, a huge amount of joint action pairs can be skipped during the Q-table updating, since they violates the computing and transmission constraints.

Proposition 2: For a Het-MEC with multiple CCs, the total number of joint action pairs which do not take part in Q-table updating can be calculated as

$$|Q^-| = [a^L - \frac{\prod^{0\leq i\leq L-1}(a+i)}{L!}](\frac{|\mathcal{A}|}{a})^L \cdot \frac{1-[\frac{\prod^{0\leq i\leq L-1}(a+i)}{L!a^L}]^{5MN}}{1-\frac{\prod^{0\leq i\leq L-1}(a+i)}{L!a^L}} \tag{26}$$

Proof: Please see Appendix B. ∎

3) Complexity: The computational complexity is $\mathcal{O}(L|\mathcal{A}|^L)$ in each iteration [41], due to the calculation of the observed frequency distribution of other agents in (21) and (22). However, the computational complexity can be reduced by reducing the action space size $|\mathcal{A}|$ and the joint action pair set size $|\mathcal{A}|^L$.

4) Information Interchange: In the multi-cloud Het-MEC system, CCs aim to achieve the system latency minimization by performing a distributed scheme. Information about the situations of other CCs in scalars is necessary for each CC, in order to obtain an optimal scheme. In practice, it is difficult to exchange and share information among these independent CCs, and MARL-JTORA can keep the amount of interchange information to a minimum. Namely, only the data generation speed of each ED and a index that indicates which action in the action space is chosen by CC l need to be sent to other CCs. The amount of information exchanged per iteration can be calculated by $L(L-1)\cdot(MN+1)$.

5) Scalability: The scalability of the multi-cloud Het-MEC network is discussed by analyzing the influence of the number of CCs, APs, or EDs on the network.

Since the system latency is defined as the maximum sub-task latency according to Section II-B and Section III-B, we aims to minimize the longest sub-task latency which is the bottleneck for the system latency minimization, i.e., solving the min-max problem in (19), until all the sub-tasks' latency are equal. Given equal data generation speeds for each sub-task, the same latency can be achieved for all sub-tasks by performing the same task offloading strategy and resource allocation scheme.

Remark 1: Given equal data generation speeds for each sub-task, the minimal system latency can be achieved when all sub-tasks has the same task offloading strategy and resource allocation scheme. The system latency equals to the latency of each sub-task.

As the number of CCs grows, more and more tasks to be processed flood the ED and AP layers. The restricted resources need to allocated to satisfy more computing or transmission

TABLE II
SIMULATION PARAMETERS

Parameter	Value
Computing capacity of each ED θ_{ED}	0.12 Mbps
Computing capacity of each AP θ_{AP}	0.4 Mbps
Computing capacity of each CC θ_{CC}	1.5 Mbps
Wireless Transmission capacity of each AP ϕ_{AP}	0.3 Mbps
Wired Transmission capacity of each CC ϕ_{CC}	1 Mbps
Compression ratio ρ	10%
Discounted ratio γ	0.9
Initial Learning rate α_0	0.15
Initial exploration ratio ϵ_0	0.9

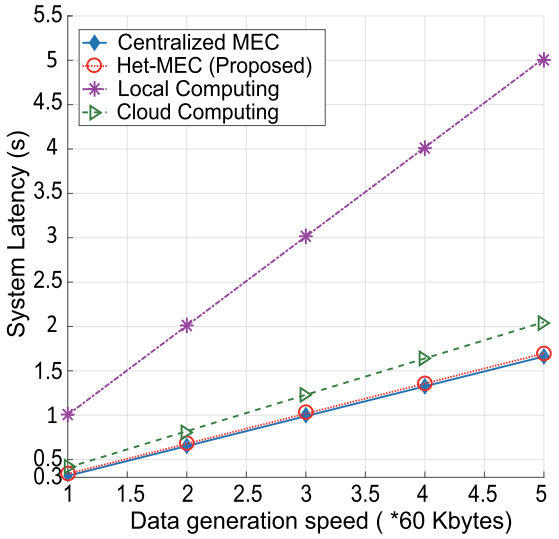


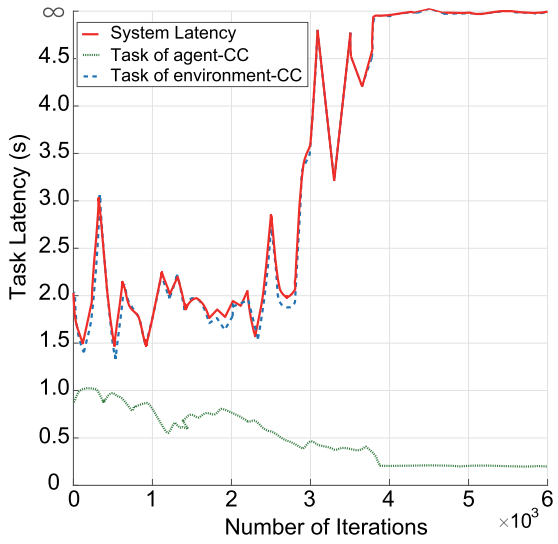
Fig. 3. System latency versus the data generation speed with 2 CCs.

requirements, i.e., computing resources in the AP and ED layer, and the wireless transmission resources. Considering a sub-task l which data generated from ED m connected with AP n, the stages of data processing at ED m, data transmission to AP n, or data processing at AP n, will be the longest stage which is the main decisive factor for the sub-task latency. We assume that the data generation speeds are equally for each sub-task, the above resources decrease linearly, as the number of CCs increases. Accordingly, when the task offloading strategy $\boldsymbol{x}$ is unchanged, the system latency increases in $\mathcal{O}(L)$.

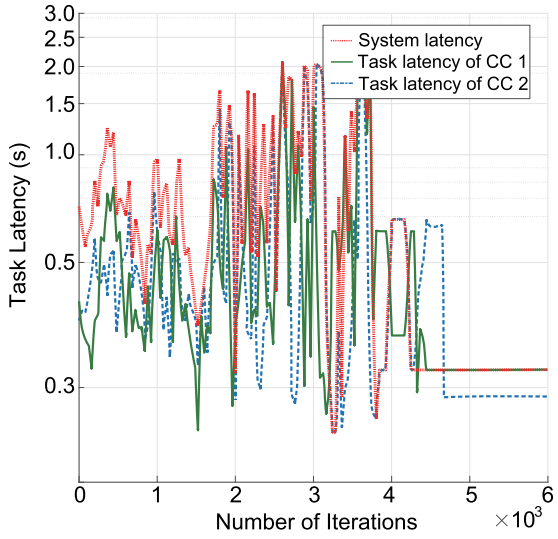
Similarly, with the increasing of the number of APs or the number of EDs connected with each AP, the system latency also increases linearly.

V. SIMULATION RESULTS

In this section, we provide the simulation results our MARL-JTORA algorithm for the multi-cloud Het-MEC system. To evaluate the network performance, several metrics are adopted, such that the system latency, resource competition depiction, scalability, convergence speed, and complexity.



(a) Single-agent reinforcement learning.



(b) Multi-agent reinforcement learning.

Fig. 4. Task latency versus number of iterations in different reinforcement learning frameworks.

The performance of our proposed scheme is evaluated based on these metrics, compared with the existing ones[3] including

- Local Computing: All data are processed at the EDs locally and then delivered to the corresponding CCs. This is suitable for the case that the task load is light and the EDs are able to process the tasks timely.
- Cloud Computing: All data are processed at the CCs. The input data stream is uploaded to CCs directly, and all tasks are processed centrally at the CCs. This is suitable for the case that the tasks are resource-intensive, while the edge nodes (including the APs and EDs) do not possess enough resources to deal with the tasks.
- Centralized MEC: Tasks are divided into parts, processed at different layers, and finally aggregated at CCs. Differ-

[3]For a fair comparison, we set each scheme has the same number of CCs/APs/EDs, same data generation speeds, and same computing and transmission capacity of each device.

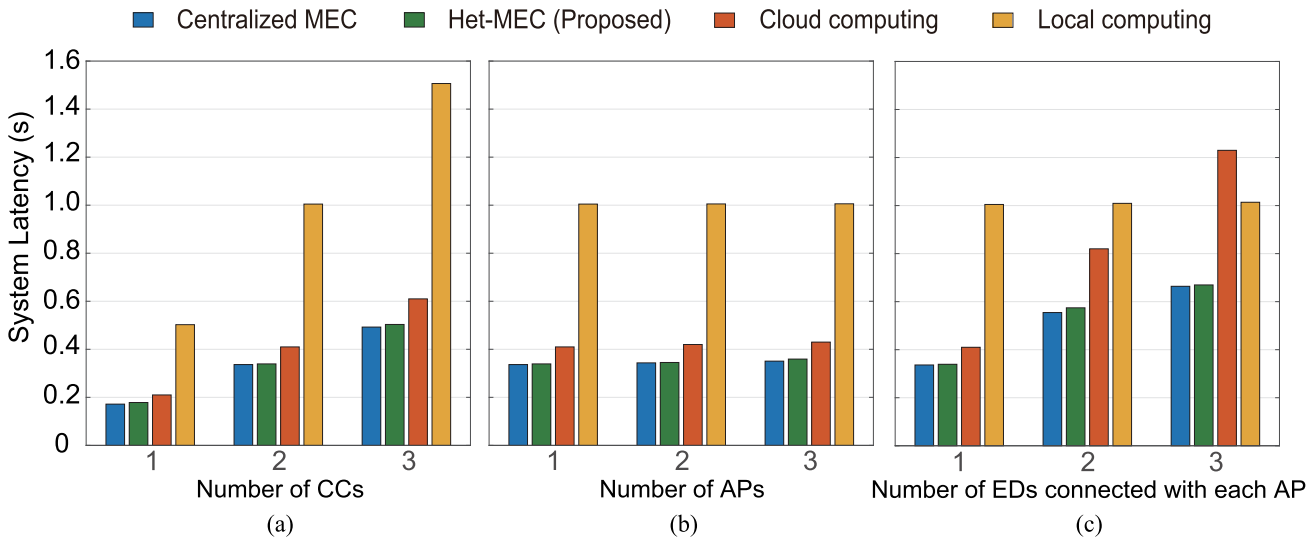


Fig. 5. The system latency vs. the number of CCs/APs/EDs in the Het-MEC system with different schemes ($\lambda = 1$).

ent from our proposed algorithm, the centralized MEC can provide an optimal scheme by exhaustively searching with a center controller which has the global information and decide the task offloading and resource allocation. We consider it as a theoretical lower band of system latency since there is no center controller in practice.

A. Parameters Setting

Simulation parameters are set up based on the existing works [23], [24] and a common set of values of RL [42], [43], as given in Table II. In the multi-cloud Het-MEC network, the computing/transmission capacity of each node represents the maximum volume of the processed/transmitted data per second. The exploration ratio is the exponent function of the iterations, expressed by $\epsilon^{(k)} = 0.95^k \epsilon_0$, satisfying $\lim_{k\to\infty} \epsilon^{(k)} = 0$.

B. System Latency Evaluation

In Fig. 3, we illustrate the performance of our proposed MARL-JTORA algorithm in a HetMEC system with 2 CCs, 1 AP, and 1 ED, where the resource allocation scheme and task offloading strategy are discretized into 5 and 6 levels, respectively (i.e., a = 5 and b = 6). It shows that the system latency of all schemes increase linearly with the data generation speed,[4] since more data requires more computing and transmission resources, and resulting in large system latency. By performing our proposed algorithm, the system latency remains lower than the cloud and local computing given different data generation speed, and very close to that of centralized MEC scheme.

C. Multiple Clouds Coordination With MARL

Since tasks are deployed on multiple independent CCs in a distributed manner, these CCs naturally compete for computing and transmission resources of the APs and EDs, in order to minimize the task latency. In this part, we depict such resource competition by comparing our MARL-JTORA scheme with a single-agent RL framework.

Fig. 4 shows the task latency and system latency versus the number of iterations in single-agent RL and MARL. In a HetMEC system with 2 CCs, 1 AP, and 1 ED, we discretize the resource allocation scheme and task offloading strategy into 5 and 6 levels, respectively (i.e., a = 5 and b = 6). For the single-agent RL, one CC learns as an agent (named as agent-CC) and the other CC is regarded as part of the environment (named as environment-CC). We assume that the environment-CC adopts the optimal task offloading scheme by utilizing the rest resources. In the single-agent RL process, the agent-CC is driven to occupy more resources of the APs and EDs, until no resources is available for the environment-CC. In contrast with the agent-CC, the latency of environment-CC significantly increases with the number of iterations. When the agent-CC obtains all resources after several iterations, the environment-CC latency is infinite indicating that the CC is crashed. Therefore, single-agent RL leads to an enormous imbalance between task latency of two CCs, as well as an infinite system latency.

Compared to the single-agent RL algorithm, all CCs in our MARL-JTORA algorithm achieves a lower task latency. For the entire system, the latter can achieve an optimal system latency, i.e., 0.34 second, while the former makes the system breakdown. By comparing Fig. 4(a) and Fig. 4(b), we can observe that the MARL-JTORA algorithm can better depict the competitive behaviors of multiple clouds, and achieve a minimum system latency since that each CC takes all other CCs into account.

D. Scalability

Fig. 5(a) shows the system latency versus the number of CCs in the multi-cloud MEC system. When only one CC exists in the system, the system latency of MARL-JTORA scheme

[4]Given the fixed learning rate, exploration strategy, and discount parameter, our proposed algorithm achieves similar results with different initial actions.

TABLE III
THE NUMBER OF ITERATIONS TO CONVERGE WITH DIFFERENT NUMBER OF CCS / APS / EDS

		Number of Iterations to Converge ($\times 10^3$)		
		CC	**AP**	**ED**
Number of CCs/APs/EDs	**1**	3.9	4.8	4.7
	2	4.8	5.2	5.3
	3	52	5.9	5.4

is 0.179 seconds, it increases to 0.339 seconds in the 2-CCs MEC system, and for the 3-CCs network, the system latency rises to 0.503 seconds. With the increasing of the number of CCs, the system latency increases linearly, since the restricted computing and transmission resources of APs and EDs are required to handle more tasks. Evidently, the proposed scheme achieves a superior system latency performance very close to the centralized MEC with a negligible gap.

Fig. 5(b) shows the system latency versus the number of APs in the multi-cloud MEC system. Different from the case in Fig. 5(a), the system latency with different number of APs is much the same. This is because when the number of APs varies, the data volume required to be processed/transmitted at the AP and ED layers remain the same, and the CC layer has the sufficient resources to handle/deliver extensive computation and transmission.

Fig. 5(c) shows the system latency versus the number of EDs connected with each AP in the multi-cloud MEC system.[5] As the number of EDs connected with each AP increasing, the system latency grows. However, the system latency of local computing does not change. The reason is that with more EDs, the computing resource of the AP and CC layers, and both the wireless and wired transmission resources, are becoming scarcer. By enabling ED to process the entire task locally, the computing resources of APs and CCs become irrelevant. Moreover, the local computing scheme can reduce the amount of data to be uploaded since the processing results usually have a much smaller size than the raw data. Therefore, as the number of EDs connected with each AP increases, it shows a relatively small change.

E. Convergence Speed

Table III shows the influence of number of CCs, APs and EDs on the convergence speed. As the number of CCs grows, the convergence speed slows down. When there is only one CC in the MEC network, learning curve converges to the stable value by 3.9×10^3 number of iterations. In the 2-CC case, 4.8×10^3 number of iterations is required. For the Het-MEC network with 3 CCs, it converges after 5.2×10^4 iterations. The number of iterations to converge grows exponentially since the Q-table size is a exponent function of the number of CCs.

However, Table III also shows that both the number of APs and the number of EDs connected with each AP do not influence the convergence speed significantly. This is because the number of APs and EDs connected with each AP influence the Q-table size only in $\mathcal{O}(n)$.

F. Computational Complexity

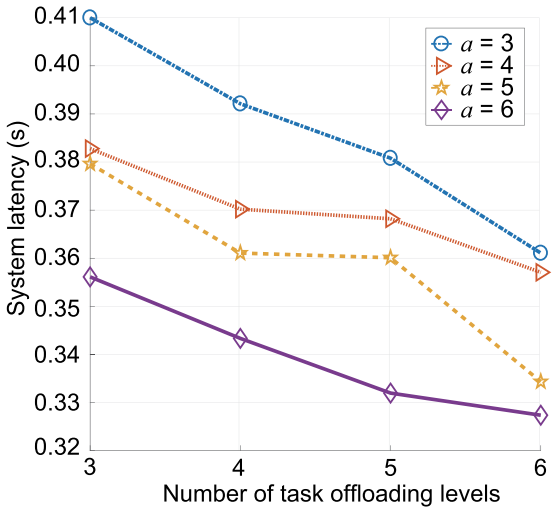
As shown in Section IV-D.3, the computational complexity is $\mathcal{O}(L|\mathcal{A}|^L)$ in each iteration. It can be reduced significantly by decreasing the size of action space $|\mathcal{A}|$ which is an important index to measure the complexity. To bound the size of action space consisting of task offloading and resource allocation, both of them are discretized into several levels. The number of action levels, i.e., the number of resource levels, a, and task offloading levels, b, directly determines the action space size.

As shown in Fig. 6, the higher the number of action levels, the larger the action space which leads to a higher computational complexity, and the calculation precision of latency increases. Therefore, the number of action levels should be carefully selected to achieve a tradeoff between the complexity and system latency optimization precision. Similarly, the discretization of state space also causes decline of the calculation precision. Fig. 6(b) also shows that the action space reduction operation in Section IV-B can reduce the action space efficiently.

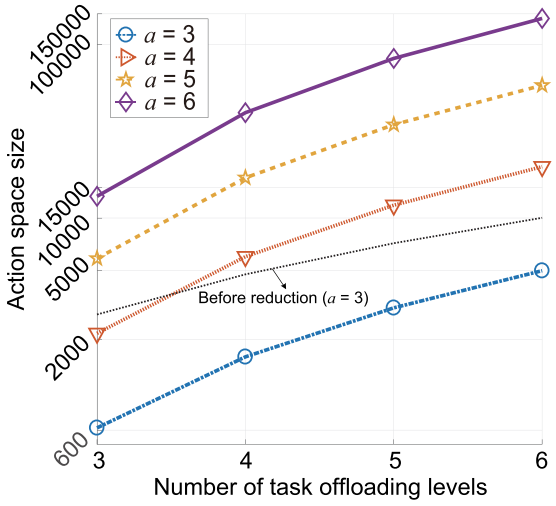
VI. CONCLUSION

In this paper, we have studied an uplink heterogeneous multi-cloud MEC network consisting of multiple independent CCs, APs, and EDs. A number of tasks are published to the EDs and APs by the CCs, and the results are aggregated at the CCs after processing at each layer. We then proposed a distributed scheme to minimize the system latency, where the CCs compete for computing and transmission resources of APs and EDs, and each CC determines the task offloading strategy independently. Competition among the CCs is depicted by the MARL-JTORA algorithm where each CC learns explicit models of other CCs as stationary distributions over their actions. From the simulation results, we conclude that our proposed scheme achieves a lower latency of the entire system compared with the cloud computing and local

[5]In practice, the CCs rely on a large number EDs to generate raw data. Without loss of generality, we evaluate the latency performance of the Het-MEC network with a small amount of EDs as examples [44].



(a)



(b)

Fig. 6. (a) Latency vs. the number of action levels; (b) Action space size vs. the number of action levels.

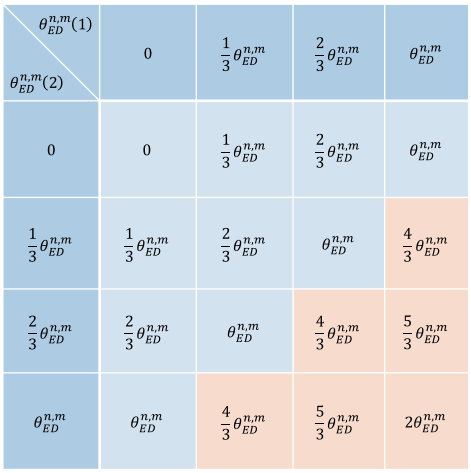
TABLE IV
ACTION SPACE REDUCTION

Elimination Principles	Proportion of $\lvert\mathcal{A}\rvert$
$x_{ED} = 1$ and $x_{AP} \neq 0$	$\frac{1}{b}(1-\frac{1}{b})$
$x_{ED} = 0$ and $\theta_{ED} \neq 0$	$\frac{1}{b}(1-\frac{1}{a})$
$x_{AP} = 0$ and $\theta_{AP} \neq 0$	$\frac{1}{b}(1-\frac{1}{a})$
$x_{AP} = 1$ and $\theta_{CC} \neq 0$	$\frac{1}{b}(1-\frac{1}{a})$

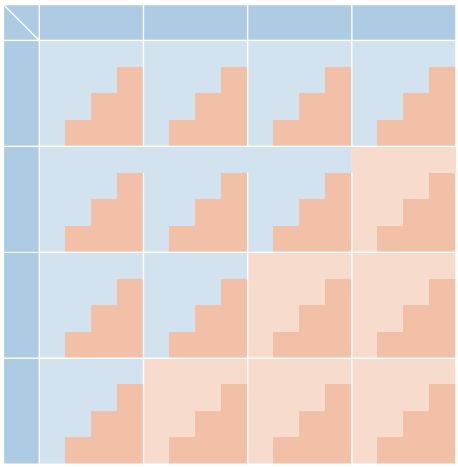
computing schemes. Moreover, when each ED has the same data generation speed, the system latency increases linearly with the number of CCs/APs/EDs.

APPENDIX A
PROOF OF PROPOSITION 1

When only one ED takes part in the task, the actions in Table IV are deleted from the action space according



(a)



(b)

Fig. 7. Joint action pair set reduction of 2 agents reinforcement learning ($a = 4$).

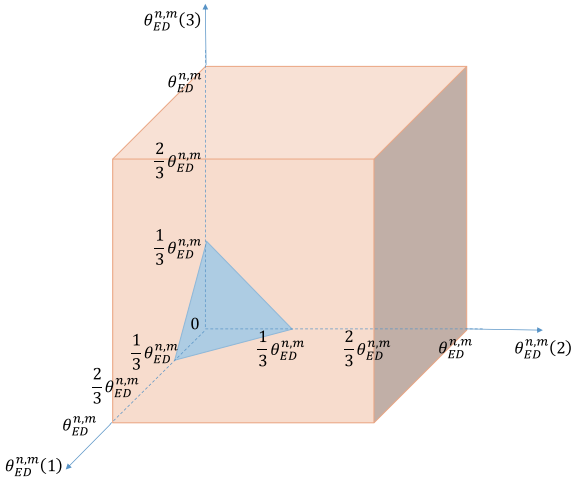


Fig. 8. Joint action pair set reduction of 3 agents reinforcement learning ($a = 4$).

to the action space elimination principles. Note that these principles are partially overlapped which are summarized in Table V.

TABLE V
OVERLAP BETWEEN THE ELIMINATION PRINCIPLES

Overlap	Proportion of $\lvert\mathcal{A}\rvert$
$x_{ED}=1$, $x_{AP}=1$ and $\theta_{CC}\neq 0$	$\frac{1}{b}\times\frac{1}{b}\times(1-\frac{1}{a})$
$x_{ED}=0$, $x_{AP}=0$, $\theta_{ED}\neq 0$ and $\theta_{AP}\neq 0$	$[\frac{1}{b}(1-\frac{1}{a})]^2$
$x_{ED}=0$, $x_{AP}=1$, $\theta_{ED}=0$ and $\theta_{CC}\neq 0$	$[\frac{1}{b}(1-\frac{1}{a})]^2$

Therefore, for the case that only one ED takes part in the task, the action space can be reduced to $\zeta\cdot|\mathcal{A}|$, where

$$\begin{aligned}\zeta &= 1-\{\frac{1}{b}(1-\frac{1}{b})+\frac{3}{b}(1-\frac{1}{a})-\frac{1}{b}\times\frac{1}{b}\times(1-\frac{1}{a})\\ &\quad -2\times[\frac{1}{b}(1-\frac{1}{a})]^2\}\\ &= \frac{a^2b^2-4a^2b+4a^2+3ab-5a+2}{a^2b^2}. \end{aligned}\tag{27}$$

When $M\cdot N$ EDs generate raw data for the task, the action space decreases exponentially, namely, it can be reduced to $\zeta^{MN}\cdot|\mathcal{A}|$.

APPENDIX B
PROOF OF PROPOSITION 2

For a Het-MEC network with 2 CCs, we suppose that only the computing resource constraint of ED m connected to AP n is adopted to reduce the joint action set. As shown in Fig. 7(a), the pairs in the lower triangular portion of the matrix exceed the limit and should be skipped during Q-table updating. Therefore, $[a^2-\frac{a(a+1)}{2}]\cdot(\frac{|\mathcal{A}|}{a})^2$ joint action pairs do not participate in the Q-table updating, where $(\frac{|\mathcal{A}|}{a})^2$ indicates that there are $(\frac{|\mathcal{A}|}{a})^2$ joint action pairs for each $\theta_{ED}^{n,m}(1)-\theta_{ED}^{n,m}(2)$ pair, since the matrix is symmetric and each action includes other task offloading strategies and resource allocation schemes.

Generally, $5MN$ resources are required to be decided by each agent when MN EDs take part in each task. As shown in Fig. 7(b), joint action pairs in more lower triangular portions should be skipped during Q-table updating when more resource constraints are taken into consideration. For every extra resource, the additional parts forms a geometric progression, and the common ratio is $\frac{a(a+1)}{2a^2}$.

As shown in Fig. 8, the joint action set presents a 3-dimensional matrix when there are 3 CCs in the Het-MEC network. Similarly, considering one resource constraint, only the blue triangular pyramid can participate in the Q-table updating, the skipped part can be calculated by $a^3-\frac{a(a+1)(a+2)}{6}$. When taking more constraints into account, the common ratio of additional parts is $\frac{a(a+1)(a+2)}{6a^3}$.

It can be extended to Het-MEC with multiple CCs, the total number of joint action pairs which do not take part in Q-table updating can be calculated as

$$\begin{aligned}Q^- = [a^L-\frac{\prod^{0\le i\le L-1}(a+i)}{L!}](\frac{|\mathcal{A}|}{a})^L\\ \cdot\frac{1-[\frac{\prod^{0\le i\le L-1}(a+i)}{L!a^L}]^{5MN}}{1-\frac{\prod^{0\le i\le L-1}(a+i)}{L!a^L}}\end{aligned}\tag{28}$$

REFERENCES

[1] R. Deng, B. Di, and L. Song, "Cooperative collision avoidance for overtaking maneuvers in cellular V2X-based autonomous driving," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4434–4446, May 2019.

[2] C. Doukas and I. Maglogiannis, "Bringing IoT and cloud computing towards pervasive healthcare," in *Proc. 6th Int. Conf. Innov. Mobile Internet Services Ubiquitous Comput.*, Jul. 2012, pp. 922–926.

[3] F. Cicirelli *et al.*, "Edge computing and social Internet of Things for large-scale smart environments development," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2557–2571, Aug. 2018.

[4] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[5] W. Guo, J. Gong, W. Jiang, Y. Liu, and B. She, "OpenRS-cloud: A remote sensing image processing platform based on cloud computing environment," *Sci. China Technol. Sci.*, vol. 53, no. S1, pp. 221–230, May 2010.

[6] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, Oct. 2011.

[7] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.

[8] B. Di, H. Zhang, L. Song, Y. Li, and G. Y. Li, "Ultra-dense LEO: Integrating terrestrial-satellite networks into 5G and beyond for data offloading," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 47–62, Jan. 2019.

[9] B. Di, H. Zhang, L. Song, Y. Li, Z. Han, and H. V. Poor, "Hybrid beamforming for reconfigurable intelligent surface based multi-user communications: Achievable rates with limited discrete phase shifts," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 8, pp. 1809–1822, Aug. 2020.

[10] T. Wang, S. Wang, and Z.-H. Zhou, "Machine learning for 5G and beyond: From model-based to data-driven mobile wireless networks," *China Commun.*, vol. 16, no. 1, pp. 165–175, Jan. 2019.

[11] H. Guo, J. Liu, and J. Zhang, "Computation offloading for multi-access mobile edge computing in ultra-dense networks," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 14–19, Aug. 2018.

[12] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

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

[14] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper 11, Sep. 2015.

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

[16] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[17] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[18] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7432–7445, Aug. 2017.

[19] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[20] T. Quang Dinh, J. Tang, Q. Duy La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[21] K. Cheng, Y. Teng, W. Sun, A. Liu, and X. Wang, "Energy-efficient joint offloading and wireless resource allocation strategy in multi-MEC server systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, May 2018, pp. 1–6.

[22] S.-W. Ko, K. Han, and K. Huang, "Wireless networks for mobile edge computing: Spatial modeling and latency analysis," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5225–5240, Aug. 2018.

[23] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[24] L. Yang, H. Zhang, X. Li, H. Ji, and V. C. M. Leung, "A distributed computation offloading strategy in small-cell networks integrated with mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2762–2773, Dec. 2018.

[25] P. Wang, C. Yao, Z. Zheng, G. Sun, and L. Song, "Joint task assignment, transmission, and computing resource allocation in multilayer mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2872–2884, Apr. 2019.

[26] J. Liu, H. Guo, J. Xiong, N. Kato, J. Zhang, and Y. Zhang, "Smart and resilient EV charging in SDN-enhanced vehicular edge computing networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 1, pp. 217–228, Jan. 2020.

[27] H. Guo, J. Liu, J. Ren, and Y. Zhang, "Intelligent task offloading in vehicular edge computing networks," *IEEE Wireless Commun.*, vol. 27, no. 4, pp. 126–132, Aug. 2020.

[28] H. Guo and J. Liu, "UAV-enhanced intelligent offloading for Internet of Things at the edge," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2737–2746, Apr. 2020.

[29] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, Jan. 1996.

[30] M. A. AlZain, E. Pardede, B. Soh, and J. A. Thom, "Cloud computing security: From single to multi-clouds," in *Proc. 45th Hawaii Int. Conf. Syst. Sci.*, Maui, HI, USA, Jan. 2012, pp. 5490–5499.

[31] N. Grozev and R. Buyya, "Multi-cloud provisioning and load distribution for three-tier applications," *ACM Trans. Auto. Adapt. Syst.*, vol. 9, no. 3, pp. 1–21, Oct. 2014.

[32] Y. Zhang, B. Di, P. Wang, J. Lin, and L. Song, "HetMEC: Heterogeneous multi-layer mobile edge computing in the 6 G era," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4388–4400, Apr. 2020.

[33] M. Hilbert, P. López, and C. Vásquez, "Information societies or 'ICT equipment societies?' measuring the digital information-processing capacity of a society in bits and bytes," *Inf. Soc.*, vol. 26, no. 3, pp. 157–178, Apr. 2010.

[34] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

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

[36] E. Yang and D. Gu, "Multiagent reinforcement learning for multi-robot systems: A survey," Dept. Comput. Sci., Univ. Essex, Colchester, U.K., 2004.

[37] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Trans. Cognit. Commun. Netw.*, vol. 3, no. 3, pp. 361–373, Sep. 2017.

[38] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," in *Proc. Conf. Innov. Appl. Artif. Intell.*, Madison, WI, USA, Jul. 1998, pp. 746–752.

[39] D. Fudenberg and D. M. Kreps, "Lectures on learning and equilibrium in strategic form games," in *CORE* (Lecture Series). Newark, NJ, USA: Mimeo, 1990.

[40] E. Rodrigues Gomes and R. Kowalczyk, "Dynamic analysis of multiagent Q-learning with-greedy exploration," in *Proc. 26th Annu. Int. Conf. Mach. Learn. (ICML)*, 2009, pp. 369–376.

[41] A. K. Sadhu and A. Konar, "An efficient computing of correlated equilibrium for cooperative Q-learning-based multi-robot planning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 8, pp. 2779–2794, Aug. 2020.

[42] M. Wunder, M. L. Littman, and M. Babes, "Classes of multiagent Q-learning dynamics with ϵ-greedy exploration," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 1167–1174.

[43] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[44] P. Wang, Z. Zheng, B. Di, and L. Song, "HetMEC: Latency-optimal task assignment and resource allocation for heterogeneous multi-layer mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4942–4956, Oct. 2019.

Yutong Zhang (Graduate Student Member, IEEE) received the B.S. degree in electronics information science and technology from Yunnan University, Kunming, China, in 2017, and the M.E. degree from the School of Software and Microelectronics, Peking University, China, in 2020, where she is currently pursuing the Ph.D. degree with the Department of Electronics. Her current research interests include wireless communications and edge computing.

Boya Di (Member, IEEE) received the B.S. degree in electronic engineering from Peking University, in 2014, and the Ph.D. degree from the Department of Electronics, Peking University, China, in 2019. She currently works as a Post-Doctoral Researcher with Imperial College London. Her current research interests include integrated aerial access and satellite networks, reconfigurable intelligent surfaces, multiagent systems, edge computing, and vehicular networks. One of her journal papers is currently listed as ESI highly cited articles. She serves as an Associate Editor of the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.

Zijie Zheng (Member, IEEE) received the B.S. degree in electronic engineering from Peking University, China, in 2014, and the Ph.D. degree from the Department of Electronics, Peking University, China, in 2019. His current research interests include game theory and optimization in 5G networks, wireless powered networks, mobile social networks, and wireless big data.

Jinlong Lin received the Ph.D. degree from Peking University, Beijing, China, in 2003. He is currently a Professor with the School of Software and Microelectronics, Peking University. His main research interests include digital image processing, digital image analysis pattern recognition, and embedded system design.

Lingyang Song (Fellow, IEEE) received the Ph.D. degree from the University of York, U.K., in 2007, where he received the K. M. Stott Prize for excellent research. He worked as a Research Fellow with the University of Oslo, Norway, until rejoining Philips Research UK in March 2008. In May 2009, he joined the Department of Electronics, School of Electronics Engineering and Computer Science, Peking University, and is currently a Distinguished Professor. His main research interests include wireless communication and networks, signal processing, and machine learning. He was a recipient of the IEEE Leonard G. Abraham Prize in 2016 and the IEEE Asia Pacific (AP) Young Researcher Award in 2012. He has been an IEEE Distinguished Lecturer since 2015.