Deep Reinforcement Learning based Dynamic Resource Management for Mobile Edge Computing in Industrial Internet of Things

Ying Chen, *Member, IEEE,* Zhiyong Liu, Yongchao Zhang, Yuan Wu, *Senior Member, IEEE,* Xin Chen, and Lian Zhao, *Senior Member, IEEE*

***Abstract*—Nowadays, driven by the rapid development of smart mobile equipments and 5G network technologies, the application scenarios of Internet of Things (IoT) technology are becoming increasingly widespread. The integration of IoT and industrial manufacturing systems forms the Industrial IoT (IIoT). Because of the limitation of resources such as the computation unit and battery capacity in the IIoT equipments (IIEs), computation-intensive tasks need to be executed in mobile edge computing (MEC) server. However, the dynamics and continuity of task generation lead to a severe challenge to the management of limited resources in IIoT. In this paper, we investigate the dynamic resource management problem of joint power control and computing resource allocation for MEC in IIoT. In order to minimize the long-term average delay of the tasks, the original problem is transformed into a Markov decision process (MDP). Considering the dynamics and continuity of task generation, we propose a Deep reinforcement learning based Dynamic Resource Management (DDRM) algorithm to solve the formulated MDP problem. Our DDRM algorithm exploits the deep deterministic policy gradient and can deal with the high dimensional continuity of the action and state spaces. Extensive simulation results demonstrate that the DDRM can reduce the long-term average delay of the tasks effectively.**

***Index Terms*—Industrial Internet of things, mobile edge computing, dynamic resource management, deep reinforcement learning.**

I. Introduction

With the popularization of smart mobile equipments and the rapid development of 5G network technologies, the application scenarios of Internet of Things (IoT) technology are becoming increasingly widespread [1]. The world is expected to have about 28 billion connected equipments by 2021 [2]. The industrial sector is one of the beneficiaries of IoT application. The integration of the industrial manufacturing systems with IoT is defined as the industrial IoT (IIoT) [3]. In IIoT, data can be collected from sensors, actuators, and other devices in an industrial environment, and the devices are accessed and controlled through Internet. Similar to the general IoT, low cost, limited resources equipments and network extendibility are the basic communication requirements of IIoT. However, IIoT has higher requirements on quality of service (QoS), availability, reliability and security [2]. In the industrial environment, IIoT equipments (IIEs) may continuously generate different types of tasks, leading to a large amount of data traffic [3]. These task data need to be processed in a timely, reliable and efficient manner. However, in general, the resources of IIEs (e.g., sensors) are extremely limited [4]. Mobile edge computing (MEC) is an effective way to break through this bottleneck, in which tasks are offloaded to the MEC server for efficient execution [5].

MEC is an emerging architecture that uses mobile access points to extend cloud computing resources to wireless access networks close to users [6]. As an important part of the 5G, MEC aims to enhance the potential computing capacity of equipment and provide low-latency computation services for the computation-intensive tasks generated by IIEs. Compared with traditional centralized cloud computing, MEC has better performance in delay, scalability and reliability [7]. The emergence of MEC fills the blank from the equipment to the central cloud. The integration of IIoT with MEC will greatly reduce latency and improve bandwidth utilization to enhance the user experience and overall network performance. However, task offloading will incur an additional transmission delay and energy consumption. In addition, how to properly allocate the computing resources at the MEC server will also influence the task execution delay. Therefore, it is worth discussing how to effectively manage communication and computing resources in IIoT.

There have been many studies investigating the joint management of communication and computing resources of MEC. Some works have studied the wireless resource management problem in IIoT, through spectrum resource allocation, power control and other methods to improve QoS and network throughput, etc. [8]. In addition, the task offloading issues in MEC have also received much attention. For instance, some works studied how to properly allocate the computing resource of MEC server to enhance the execution ability of the system and lower the delay of task [9]. However, these existing works do not take the dynamics and continuity of task generation into account, making their proposed approaches difficult to be applied to the more complex realistic environments with dynamic and continuous task generation in IIoT. Meanwhile, limited battery capacity and computing resource limit the task processing efficiency. Therefore, how to dynamically control the transmission power of IIEs and allocate computing resource according to time-varying task attributes under the circumstances of limited battery capacity and computing resource is of great

Y. Chen, Z. Liu, Y. Zhang, and X. Chen are with School of Computer, Beijing Information Science and Technology University, Beijing 100101, China. (E-mail: chenying@bistu.edu.cn; liuzhiyong@mail.bistu.edu.cn; zhangyongchao@mail.bistu.edu.cn; chenxin@bistu.edu.cn).

Y. Wu is with State Key Lab of Internet of Things for Smart City, University of Macau, Macao, China. (E-mail: yuanwu@um.edu.mo).

L. Zhao is with Department of Electrical, Computer and Biomedical Engineering Ryerson University, Toronto, Canada. (E-mail: l5zhao@ryerson.ca).

research significance.

Motivated by this consideration, we design an efficient intelligent algorithm to jointly optimize power control and computing resource allocation for MEC in IIoT. Thanks to the latest advance in the technique of deep reinforcement learning (DRL) and its advantage in addressing large-scale dynamic optimization problems [10], we adopt the deep deterministic policy gradient (DDPG) method to study this problem. In particular, compared to the conventional reinforcement learning (RL) schemes, the advantage of using DDPG algorithm is two-fold. Firstly, the adopted DDPG method is able to efficiently address the dimensional disaster issue of system states and actions. Secondly, the DDPG can accurately select actions in continuous action spaces. The main contributions of this paper are as follows:

- We investigate the dynamic resource management problem for the joint power control and computing resource allocation for MEC in IIoT. In the case of dynamic task generation, the delay of all generated tasks over a period of time is optimized.
- Then, the problem of minimizing the long-term average delay is transformed into a markov decision process (MDP). We propose a DRL based Dynamic Resource Management (DDRM) algorithm to solve the MDP problem. Our DDRM algorithm exploits the DDPG and can deal with the high dimensional continuity of the action and state spaces.
- We conduct extensive experiments to evaluate the performance of the DDRM. Experiment results show that compared with uniform resource management and random resource management algorithms, the DDRM can effectively lower the delay of tasks under various environments.

In the rest of this paper, related work is summarized in section II. We present the system model and problem description in Section III. The DDRM algorithm is proposed in Section IV. Section V discusses the experiment results. Finally, Section VI summarizes our research work.

II. Related Work

With the integration of industrial manufacturing systems and IoT, the IIoT is becoming more and more complex. How to properly manage various resources in the IIoT and optimize the network performance has become a research focus in recent years. In order to ensure the fairness of users, Wang *et al*. [11] designed a distributed Q-learning auxiliary power distribution algorithm to guarantee the fairness of different devices in a two-layer heterogeneous IIoT network with limited network resource. For the QoS of system, Hasan *et al*. [12] studied the optimal deployment mode to improve the connectivity of IIoT, and proposed a canonical particle multiswarm optimization (CPMSO) algorithm based on biological heuristics. For the system throughput, Lai *et al*. [13] dynamically prioritized nodes by evaluating the self-similarity of IIoT node observation data, and then designed a Deep Q Network (DQN) based adaptive resource allocation algorithm to enhance network throughput. In order to improve resource utilization, Li *et al*. [14] designed a resource federation allocation method based on hybrid multi-access policy that could adapt to user hierarchy and transport requests. These above studies all optimized the performance of the IIoT networks, but did not take into account the problem of optimizing the task delay for IIoT tasks. This means that these works could not effectively reduce the task delay.

To solve this issue, some studies considered combining IIoT with MEC, where MEC could provide powerful computing services for IIEs to reduce task delay. With this advantage, the resource management for MEC in IIoT has also attracted increasingly attention. To reduce task delay, Yang *et al*. [15] developed a layered machine learning (ML) task allocation framework for the IIoT based on MEC, which minimized task delay while considering the complexity of the ML model, data quality, device level and MEC server computing capacity, and communication bandwidth. For delay and energy consumption, Zhao *et al*. [16] studied the communication and computation problems of an IIoT network, and proposed a three-level optimization framework based on second-hop wireless relaying and discrete particle swarm optimization algorithms to reduce delay and energy consumption. For system cost, Chen *et al*. [17] investigated the problem of multi-hop computation offloading of data processing tasks and mining tasks, and designed a collaborative distributed computation offloading scheme to minimize the economic cost. To improve the system utility, Wu *et al*. [18] developed an energy-aware resource scheduling (ERS) algorithm based on Lyapunov method to maximize system utility. In these studies, the resource management problem from different perspectives were studied separately. However, they did not take into account the continuity and dynamics of task generation in the IIoT scenario, and failed to ensure the long-term benefits of the IIoT system.

In this paper, we formulate a dynamic resource management problem to minimize the long-term processing delay of tasks in IIoT systems with MEC while taking into account the dynamics and continuity of task generation, and propose an efficient and intelligent resource management scheme based on DRL.

III. Network Model and Problem Formulation

A. Task Model

Fig. 1 depicts the considered MEC-enabled IIoT network in this work, where the center of the system is configured with a base station (BS) and one MEC server. There are N IIEs denoted by $\mathcal{N} = \{1, 2, ..., N\}$. We adopt a time-slotted model denoted $\{1, 2, ..., T\}$, in which each time slot t is of the duration of τ [19].

At the beginning of time slot t, each IIE generates a computation task which needs to be offloaded to the MEC server for execution. At each time slot t, the computation task generated by IIE n $(n \in \mathcal{N})$ is represented by a tuple $Q_n(t) = \{\omega_n(t), z_n(t)\}$, where $\omega_n(t)$ indicates the computing amount of the task (i.e., the number of CPU cycles required by task), and $z_n(t)$ represents the task data size [20]. Similar to [21], the maximum completion duration for each task is no more than the time slot length τ. Table I shows the main notations used in the system model.

B. Communication Model

We consider that the IIE adopts the orthogonal frequency division multiple access (OFDMA) for their task offloading, in which different IIEs occupy different channels for their

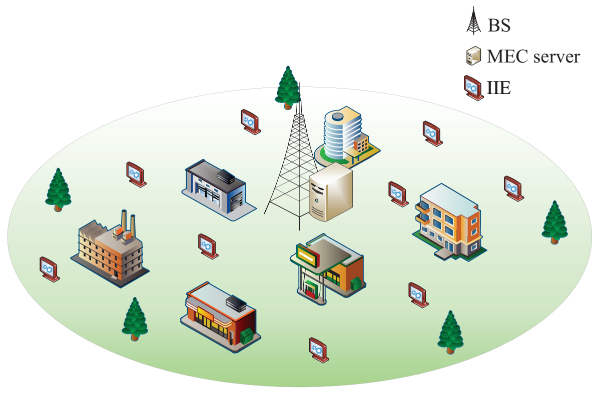


Fig. 1. MEC-enabled Industrial IoT network

TABLE I
NOTATIONS

Notation	Definition
$\mathcal{N}$	The set of IIEs
B	Channel bandwidth
$p_n(t)$	Transmission power of IIE n in time slot t
$g_n(t)$	Wireless channel gain from the IIE n to the BS in time slot t
N_0	Gaussian noise power spectrum density
$Q_n(t)$	Computation task generated at time slot t
$\omega_n(t)$	Computing amount of the task $Q_n(t)$
$z_n(t)$	Data size of task $Q_n(t)$
E_n	Total battery capacity of IIE n
$f_n(t)$	Computing resource that MEC server allocates to task $Q_n(t)$
$p_{\min}$	Minimum transmission power of IIEs
$p_{\max}$	Maximum transmission power of IIEs
F	Total computing capacity in the MEC server

transmissions. According to [22], the maximum uplink rate from the IIE n to the BS is obtained by Shannon formula as follows:

$$R_n(t) = B\log_2\left(1+\frac{p_n(t)g_n(t)}{BN_0}\right), \tag{1}$$

where $p_n(t)$ denotes the transmission power of IIE n in time slot t, $g_n(t)$ indicates the wireless channel gain from the IIE n to the BS, B indicates the wireless channel bandwidth, and N_0 represents the Gaussian noise power spectrum density.

The IIE n incurs a delay when the computation task $Q_n(t)$ is offloaded to the MEC server. The wireless transmission delay is

$$d_n^W(t) = \frac{z_n(t)}{R_n(t)}. \tag{2}$$

The wireless transmission energy consumption is

$$e_n^W(t) = p_n(t)d_n^W(t) = \frac{p_n(t)z_n(t)}{R_n(t)}. \tag{3}$$

Each IIE should have enough energy to support the offloading operation for all tasks. Thus, the total energy consumption for all tasks of the IIE n should satisfy

$$\sum_{t=1}^{T} e_n^W(t) \le E_n, \forall n \in \mathcal{N}, \tag{4}$$

where E_n denotes the battery capacity of IIE n.

C. Computation Model

For edge computation, the MEC server incurs a delay when computing the task $Q_n(t)$. Then, the computing delay of the task $Q_n(t)$ is

$$d_n^C(t) = \frac{\omega_n(t)}{f_n(t)}, \tag{5}$$

where $f_n(t)$ indicates the computing resource allocated by the MEC server to computation task $Q_n(t)$ in time slot t.

The computation task $Q_n(t)$ will generate a result after it is completed on the MEC server. Similar to [20], since the data size of the result is relatively small, the downlink delay and energy consumption for returning the result are ignored. Thus, the total delay of the computation task $Q_n(t)$ consists of two parts: (i) the transmission time for offloading the computation task $Q_n(t)$ to the BS, (ii) the delay caused by executing the computation task $Q_n(t)$ in the MEC server. Therefore, the delay required to complete the computation task $Q_n(t)$ can be calculated according to

$$d_n(t) = d_n^W(t) + d_n^C(t). \tag{6}$$

D. Problem Formulation

In each slot, the executions of different tasks are parallel, while the delays of the different tasks affect each other. Similar to [20], our goal is to minimize the average delay for all computation tasks by jointly optimizing the power control and the computing resource allocation. Specifically, for each IIE, we control its transmit-power when offloading its task in time slot t. In addition, the MEC server allocates the computing resource for each task when executing the offloaded tasks in time slot t.

We formulate the following dynamic resource management problem over a T-slot time-horizon.

$$\begin{aligned}
\textbf{P1:}\quad & \min_{\mathbf{p},\mathbf{f}} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N}\sum_{n=1}^{N} d_n(t), \\
s.t.\quad & C1: p_{\min} \le p_n(t) \le p_{\max}, \forall n \in \mathcal{N}, \\
& C2: \sum_{t=1}^{T} e_n^W(t) \le E_n, \forall n \in \mathcal{N}, \\
& C3: f_n(t) > 0, \forall n \in \mathcal{N}, \\
& C4: \sum_{n=1}^{N} f_n(t) \le F.
\end{aligned} \tag{7}$$

Here, $C1$ constrains the transmission power of IIE n in time slot t, where $p_{\min}$ denotes the minimum transmission power, and $p_{\max}$ denotes the maximum transmission power. $C2$ indicates that the energy consumption of IIE n to offload all computation tasks should not exceed the total battery capacity of IIE n. $C3$ indicates that the MEC server should allocate computing resource to each task. $C4$ states that the total computing resource allocated to all tasks in time slot t should not exceed the computing capacity in the MEC server, where F denotes the total computing capacity in the MEC server. Problem **P1** is a typical dynamic program problem over a finite horizon of T time slots. To solve Problem **P1**, we transform it into an MDP problem with a finite time-horizon of T [23]. The details are illustrated in the next section.

IV. Deep Reinforcement Learning Based Dynamic Resource Management Scheme

In this section, we transform Problem **P1** to an MDP problem and then design a DRL-based algorithm for solving it.

A. MDP Based Resource Management Problem

An MDP consists of a 5-tuple $\mathcal{M} = \{\mathbf{S}, \mathbf{A}, \mathbf{P}, \mathbf{R}, \gamma\}$, where $\mathbf{S}$ indicates the state space, $\mathbf{A}$ indicates the action space, $\mathbf{P}$ represents the state transition probability, $\mathbf{R}$ stands for the reward function, and $\gamma \in [0, 1]$ represents the discount factor. At the beginning of each time slot t, the agent senses the current system state $\mathbf{s}(t) \in \mathbf{S}$ and selects an action $\mathbf{a}(t) \in \mathbf{A}$ to execute. After the action $\mathbf{a}(t)$ is completed, the system state is updated to $\mathbf{s}(t+1)$ and the agent receives the rewards from environmental feedback. Then, we design the MDP model in detail as follows.

*1) **State Space**:* At the beginning of time slot t, the agent collects information about the current IIoT system environment. Therefore, the state space $\mathbf{s}(t)$ is

$$\mathbf{s}(t) = \{\mathbf{g}(t), \mathbf{w}(t), \mathbf{z}(t), \mathbf{e}(t)\}, \tag{8}$$

where $\mathbf{g}(t)$ indicates the wireless channel gain information for all IIEs, $\mathbf{w}(t)$ and $\mathbf{z}(t)$ represent the computing amount and data size information of the computation tasks generated by all IIEs in time slot t, respectively. $\mathbf{e}(t)$ denotes the remaining battery energy information of all IIEs at the current time slot.

*2) **Action space**:* By observing the current IIoT system state $\mathbf{s}(t)$, the agent need to make the following two decisions:

$$\mathbf{a}(t) = \{\mathbf{p}(t), \mathbf{f}(t)\}, \tag{9}$$

where $\mathbf{p}(t)$ indicates the transmission power of all IIEs and $\mathbf{f}(t)$ represents the computing resources allocated by the MEC server to all computation tasks. In particular, the actions in action space $\mathbf{a}(t)$ is given by constraints C1, C3, and C4.

*3) **State transition probability**:* The state transition probability $P(\mathbf{s}(t+1)|\mathbf{s}(t), \mathbf{a}(t))$ indicates the probability distribution of $\mathbf{s}(t+1)$ given $\mathbf{s}(t)$ and the selected $\mathbf{a}(t)$. Note that the agent has no prior knowledge of $P(\mathbf{s}(t+1)|\mathbf{s}(t), \mathbf{a}(t))$, which is only determined by the environment [24].

*4) **Reward function**:* The reward function $r(t)$ represents the immediate reward when selecting $\mathbf{a}(t)$ in $\mathbf{s}(t)$. In general, MDP is used to maximize the long-term reward of system. However, the goal of Problem **P1** is to minimize the average delay of the total task. Therefore, we can maximize the negative of the objective function of Problem **P1**. Given constraint C2, we should set different rewards to make the agent select only the actions that satisfy the constraints. The immediate reward function is as follows:

$$r(t) = \begin{cases} -\frac{1}{N}\sum\limits_{n=1}^{N} d_n(t), & \text{the C2 is satisfied;} \\ -\frac{T}{N}\sum\limits_{n=1}^{N} d_n(t), & \text{otherwise.} \end{cases} \tag{10}$$

When the C2 is satisfied, the immediate reward is the negative of the average delay of the computation task generated at each time slot. When the C2 is not satisfied, the immediate reward is set to an extremely small value that can be regarded as a penalty. For each time slot t, there must be $-\frac{1}{N}\sum\limits_{n=1}^{N} d_n(t) \gg -\frac{T}{N}\sum\limits_{n=1}^{N} d_n(t)$, and $-\frac{1}{T}\sum\limits_{t=1}^{T}\frac{1}{N}\sum\limits_{n=1}^{N} d_n(t) \gg -\frac{1}{T}\sum\limits_{t=1}^{T}\frac{T}{N}\sum\limits_{n=1}^{N} d_n(t)$. Therefore, the actions selected by the agent will satisfy the constraint C2.

B. DDPG Framework

RL has been considered as one of the most conventional schemes for solving the dynamic programming problems (e.g., MDP problems) [24]. However, conventional RL based methods usually suffer from low-efficiency due to computing the value function for all the possible pairings of the state and action spaces. The recent advanced DRL, which exploits the deep neural network (DNN) to gradually approximate the policy function and value function, has been considered as a promising approach to deal with large-scale complicated MDP problems [10]. In our MDP problem, the state and action spaces are high-dimensional and continuous. If the state and action spaces are discretized, not only the solution accuracy will be insufficient, but also the space explosion will be caused. Therefore, other DRL algorithms such as DQN are not suitable for our formulated MDP based resource allocation problem. In this work, we adopt the DDPG, which is one of the categories of the Actor-Critic based DRL, to solve this MDP problem. Specifically, DDPG directly generates a deterministic action through the DNN and meanwhile is trained by the determined strategy gradient to approach to the optimal action of each possible state. In addition, another DNN is used to approximate the value function to evaluate the generated action and meanwhile is trained for minimizing the loss function. The detailed training process of the DDPG framework is described in the following.

Define a policy μ as a mapping from the state to the action, i.e. $\mathbf{a}(t) = \mu(\mathbf{s}(t))$. For a given policy μ, define an action-state pair value function Q-value $Q^\mu(\mathbf{s}(t), \mathbf{a}(t))$ that indicates the expected return of the $\mathbf{a}(t)$ taken in $\mathbf{s}(t)$. According to Bellman's equation, $Q^\mu(\mathbf{s}(t), \mathbf{a}(t))$ is as follows:

$$Q^\mu(\mathbf{s}(t), \mathbf{a}(t)) = \mathbb{E}\left[r(t) + \gamma Q(\mathbf{s}(t+1), \mu(\mathbf{s}(t+1)))\right]. \tag{11}$$

The Critic network in the DDPG is a DNN that approximates the Q-values, i.e.,

$$Q(\mathbf{s}(t), \mathbf{a}(t)|\theta^Q) \approx Q(\mathbf{s}(t), \mathbf{a}(t)), \tag{12}$$

where θ^Q indicates the trainable parameter of the Critic network. Similar to the Critic network, the Actor network is a DNN with parameters θ^μ that approximates the optimal policy μ^*, i.e.,

$$\mu(\mathbf{s}(t)|\theta^\mu) \approx \mu^*(\mathbf{s}(t)), \tag{13}$$

where $\mu^*(\mathbf{s}(t))$ indicates the optimal behavior policy. In order to enhance the stability during network training, we introduce copy networks $Q'(\mathbf{s}, \mathbf{a}|\theta^{Q'})$ and $\mu'(\mathbf{s}|\theta^{\mu'})$ for the Critic and Actor networks respectively, called Critic and Actor target networks.

In order to improve the approximation accuracy of the Critic network to Q-value, we update parameters θ^Q by the Adam algorithm [25] to minimize the loss function as follows:

$$L(\theta^Q) = \mathbb{E}\left[(Q(\mathbf{s}(t), \mathbf{a}(t)|\theta^Q) - y(t))^2\right], \tag{14}$$

where

$$y(t) = r(t) + \gamma Q'(\mathbf{s}(t+1), \mu'(\mathbf{s}(t+1)|\theta^{\mu'})|\theta^{Q'}). \tag{15}$$

The sample data used for our training comes from replay memory buffer $\mathcal{R}$ with a limited size. At each time slot t, the tuple $\{\mathbf{s}(t), \mathbf{a}(t), r(t), \mathbf{s}(t+1))\}$ is stored in $\mathcal{R}$. In each time slot t, the Critic and Actor update the network parameters by randomly selecting a mini-batch data $\{\mathbf{s}(i), \mathbf{a}(i), r(i), \mathbf{s}(i+1))\}$ from the $\mathcal{R}$. Let M denote the size of this mini-batch data. Therefore, the loss function in (14) before can be written as

$$L(\theta^Q) = \frac{1}{M}\sum_{i=1}^{M}\left[(Q(\mathbf{s}(i), \mathbf{a}(i)|\theta^Q) - y(i))^2\right]. \tag{16}$$

The Actor network uses the performance objective to measure the performance of the policy μ [25]. In order to explore potential better policies, a random noise is introduced into the decision mechanism. Define the distinct behaviour policy that introduces random noise as

$$\beta = \mu(\mathbf{s}(t)|\theta^\mu) + \varrho(t), \tag{17}$$

where $\varrho(t)$ is the random noise. Similar to [25], Ornstein-Uhlenbeck (OU) process is used as the introduced time-related random noise to improve the training speed and exploration efficiency.

Therefore, the performance objective function $J_\beta(\mu)$ of β is expressed as

$$J_\beta(\mu) = \mathbb{E}_{p^\beta}\left[Q^\mu(\mathbf{s}(t), \mu(\mathbf{s}(t)|\theta^\mu))\right], \tag{18}$$

where p^β is the probability distribution function of the states based on the distinct behavior policy β. The optimal behavior policy is obtained by maximizing the performance function, i.e., $\mu^*(\mathbf{s}(t)) = \arg\max J_\beta(\mu)$. In order to maximize $J_\beta(\mu)$, the parameter θ^μ needs to be trained by adjusting the gradient of $J_\beta(\mu)$. Based on the mini-batch sample data, the gradient is

$$\nabla_{\theta^\mu} J_\beta(\mu) = \frac{1}{M}\sum_{i=1}^{M}\left[\nabla_{\mathbf{a}(i)}(Q^\mu(\mathbf{s}(i), \mathbf{a}(i)|\theta^Q)\nabla_{\theta^\mu}\mu(\mathbf{s}(i)|\theta^\mu)\right]. \tag{19}$$

C. DRL Based Dynamic Resource Management Algorithm

We next propose a DRL based dynamic resource management (DDRM) algorithm in IIoT. Specifically, our DDRM algorithm is an offline algorithm, in which all decisions made during algorithm training are self-exercised by the agent, and no actual decisions are made. Therefore, the algorithm requires K episodes of iterations. The agent will make a real decision at the end of the K-th episode.

For each episode, at the beginning of each time slot t, the agent controls the transmission power of each IIE and allocates the computing resource for each computation task according to the distinct behaviour policy β. Then, the delay of all tasks $d_n(t)$ in the current time slot t is collected.

At the end of each time slot t, the agent calculates $r(t)$ of $\mathbf{a}(t)$ based on the remaining battery energy of all IIEs $\mathbf{e}(t)$ and the average completion time of all tasks. Meanwhile, agent collects $\mathbf{s}(t+1) = \{\mathbf{g}(t+1), \mathbf{w}(t+1), \mathbf{z}(t+1), \mathbf{e}(t+1)\}$. Then, $\mathbf{s}(t)$, $\mathbf{a}(t)$, $r(t)$ and $\mathbf{s}(t+1)$ are stored in $\mathcal{R}$. After that, the agent train the Actor and Critic networks by randomly selecting a mini-batch of samples $\{\mathbf{s}(i), \mathbf{a}(i), r(i), \mathbf{s}(i+1))\}$ from $\mathcal{R}$. DDRM uses DNNs as approximations to approximate functions and policy functions. Specifically, the input of the

Algorithm 1: DRL Based Dynamic Resource Management (DDRM) Algorithm

```
1  Initialize the network parameters $\theta^Q$ and $\theta^\mu$;
2  Initialize the target network parameters: $\theta^{Q'} \leftarrow \theta^Q$ and
   $\theta^{\mu'} \leftarrow \theta^\mu$;
3  Initialize replay memory buffer $\mathcal{R}$ with a capacity of $C$;
4  for episode=1 to $K$ do
5      Initialize random noise $\varrho(t)$;
6      Initialize the IIoT system environment and receive the
       initial state $\mathbf{s}(1)$;
7      for $t = 1$ to $T$ do
8          Generate $\mathbf{a}(t)$ with (17);
9          Execute $\mathbf{a}(t)$ and calculate $r(t)$ with (10);
10         Receive $\mathbf{s}(t+1)$;
11         Store tuple $\{\mathbf{s}(t), \mathbf{a}(t), r(t), \mathbf{s}(t+1))\}$ in $\mathcal{R}$;
12         Randomly sample a mini-batch of $M$ tuple
           $\{\mathbf{s}(i), \mathbf{a}(i), r(i), \mathbf{s}(i+1))\}$ from $\mathcal{R}$;
13         Calculate $y(i)$ with (15);
14         Train Critic network by minimizing (16);
15         Train Actor network by policy gradient (19);
16         Update the parameters with (20);
17     end
18 end
```

Actor network is $\mathbf{s}(i)$ and the output is $\mu(\mathbf{s}(i)|\theta^\mu)$. For the Critic network, the input is $\mathbf{s}(i)$, $\mathbf{a}(i)$ and $\mu(\mathbf{s}(i)|\theta^\mu)$, and the output is $Q(\mathbf{s}(i), \mathbf{a}(i)|\theta^Q)$ and $Q^\mu(\mathbf{s}(i), \mu(\mathbf{s}(i)|\theta^\mu))$, where $Q(\mathbf{s}(i), \mathbf{a}(i)|\theta^Q)$ is used to calculate the loss function and $Q^\mu(\mathbf{s}(i), \mu(\mathbf{s}(i)|\theta^\mu))$ is used to update the parameters of (19). Similar to the Actor network, the input of the Actor target network is $\mathbf{s}(i+1)$ and the output is $\mu'(\mathbf{s}(i+1)|\theta^{\mu'})$. For the Critic target network, the input is $\mathbf{s}(i+1)$ and $\mu'(\mathbf{s}(i+1)|\theta^{\mu'})$, and the output is $Q'(\mathbf{s}(i+1), \mu'(\mathbf{s}(i+1)|\theta^{\mu'}))$. The approximate accuracy of DNNs to the policy function and the value function is gradually improved by the above iterative training. When the trainings of the Actor and Critic networks are completed, the target network parameters are updated as follows,

$$\begin{cases}\theta^{Q'} = \delta\theta^Q + (1-\delta)\theta^{Q'}, \\ \theta^{\mu'} = \delta\theta^\mu + (1-\delta)\theta^{\mu'},\end{cases} \tag{20}$$

where $\delta \ll 1$ is used to limit the rate of change for the target value, which can improve the stability of neural network training. Finally, the pseudo-code of DDRM algorithm in IIoT is given by Algorithm 1.

V. Performance Evaluation

We conduct a series of simulations to evaluate the performance of the DDRM scheme. Firstly, we investigate the impacts of several parameters on the performance of the DDRM scheme. Then, the DDRM is compared with other three schemes in different environments.

A. Simulation Settings

We consider an IIoT with one BS in the network center. The coverage radius of BS is 500 m. A MEC server with a computing capacity of $U[5, 25]$ GHz is deployed on BS. The

TABLE II
EXPERIMENTS PARAMETERS

Parameters	**Value**
Number of IIEs	$[10, 40]$
System total bandwidth	20 MHz
Transmission power	$[5, 38]$ dBm
Channel gain	$127 + 30 \log d$
Gaussian noise power spectrum density	-174dBm/Hz
Average computing amount of the tasks	$[0.5, 2]$ Gigacycles
Average data size of tasks	$[2, 10]$ MB
Maximum battery capacity of IIE	1000 J
Total computing capacity in the MEC server	$[5, 25]$ GHz
Number of time slots	100
Replay memory buffer size	3000
Mini-batch size	64

number of IIEs in the IIoT ranges from 10 to 40. According to [26], the lower limit of transmission power $p_{\min}$ is 5 dBm and the upper limit $p_{\max}$ is 38 dBm. Similar to [20], the channel gain follows $127 + 30 \log d$. Besides, the maximum battery capacity of an IIE is 1000 J, the system total bandwidth is 20 MHz, and the Gaussian noise power spectrum density $N_0 = -174$ dBm/Hz. In DDRM algorithm, random noise follows an OU process, where the mean value is 0, variance is 0.01, and regression intensity is 0.05. All DNNs are 4-layer neural networks, including an input layer, an output layer and two hidden layers. The number of neurons in the two hidden layers of the Actor network is 2048 and 1024, respectively, and the number of neurons in the two hidden layers of the Critic network is 1024 and 512, respectively. The main parameter settings in the experiments are given in Table II.

B. Parameter analysis

We first analyze the impact of learning rate and discount factor on the convergence of DDRM scheme.

1) The impact of learning rate on convergence: Fig. 2 depicts the impact of the different learning rates of Adam optimizer in DDRM on the convergence performance of average delay. The learning rates are set as 0.1, 0.01, and 0.001, respectively. In Fig. 2, when the learning rate is 0.1, the average delay curve converges at 750 episodes. When the learning rate is 0.01, the average delay curve converges at 1000 episodes. When the learning rate is 0.001, the average delay curve still cannot converge completely at 2000 episodes. It can be concluded that the larger the learning rate is, the faster the convergence rate will be. However, when the convergence rate is too large, the convergence value cannot be guaranteed to be optimal. On the contrary, if the learning rate is too small, the convergence rate will be extremely slow, and even the optimal solution is difficult to obtain within an acceptable time. This is because the learning rate decides the step size of the weight changing in the direction of gradient. The larger the learning rate, the bigger the weight change may be. Thus, it's easy to skip over the global optimal value and fall into the local optimal or diverge all the way. If the learning rate is smaller, each step toward the minimum of the loss function is smaller. Therefore, it takes more time to optimize.

2) The impact of discount factor on convergence: Fig. 3 depicts the impact of the different discount factors on the convergence performance of average delay. In this experiment, the

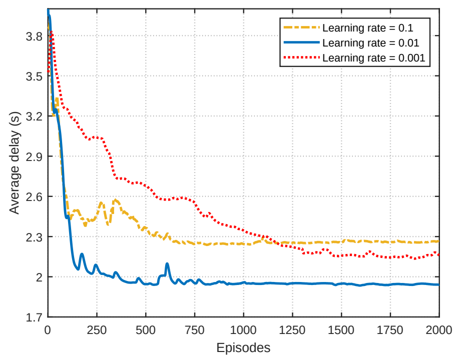


Fig. 2. The impact of learning rate on convergence

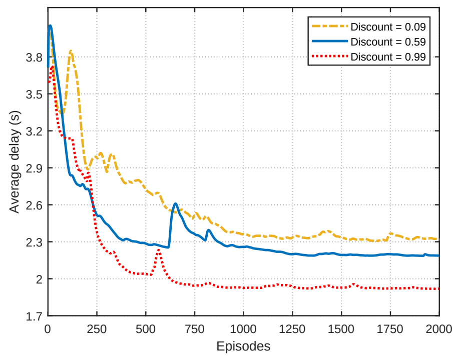


Fig. 3. The impact of discount factor on convergence

discount factors γ are set as 0.09, 0.59 and 0.99, respectively. In Fig. 3, when the discount factor is 0.09, the average delay of tasks after convergence is the highest, and the convergence value is higher than 2.3s. When the discount factor is 0.99, the average delay of tasks after convergence is the lowest, and the convergence value is lower than 2.0s. We can conclude that the larger the discount factor, the smaller the average delay of the task at convergence. This is because the smaller the discount factor, the more the present reward is valued and the less the future reward is valued. As a result, the selected action can only lower the delay in the short term, but fail to optimize the long-term average delay. The larger the discount factor, the more future rewards are valued and the more guaranteed long-term reward.

C. Comparison experiments

Then, we compare our DDRM scheme with three different resource management schemes:

- Asynchronous advantage actor-critic scheme (A3C) [27]: A3C is a DRL method based on Actor-Critic framework. Specifically, A3C uses multi-threaded exploration action space and computes the policy gradient in parallel to maintain a total update volume.
- Uniform resource management scheme (URM): In URM, the transmission power of the equipment is fixed as the average of the upper and lower limits of the power range. The MEC server allocates computing resource equally for each task.

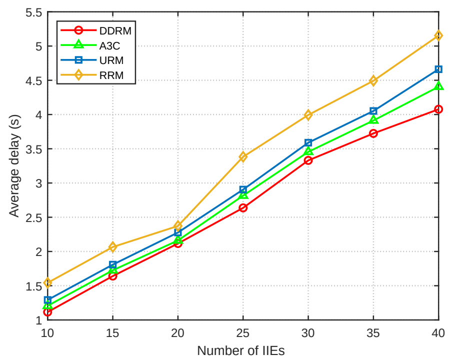


Fig. 4. Average delay with different numbers of IIEs

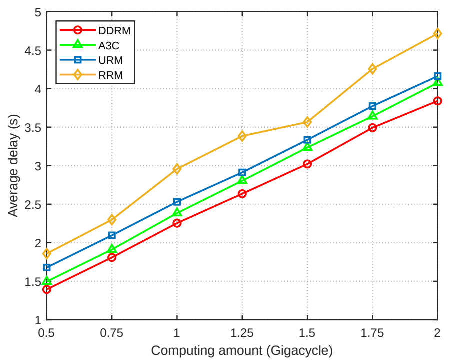


Fig. 5. The impact of computing amount of task on average delay.

- Random resource management scheme (RRM): Its main idea is that both the transmission power of the IIE and the computing resource allocated by the MEC server for each task are random.

According to the above three methods, we evaluate the average delay of tasks in terms of the number of IIEs, the computing amount of task, the data size and the computing capacity in MEC server.

1) Impact of the number of IIEs: We here consider the impact of different numbers of IIEs on the average delay of tasks. The total computing capacity in the MEC server is set to 15 GHz. The computing amount and the data size of tasks follow Gaussian distributions, with mean value of 1.25 Gigacycles and 6 MB, respectively.

Fig. 4 depicts the average delay of tasks for DDRM, A3C, URM, and RRM with different numbers of IIEs. The number of IIEs is varied from 10 to 40 with an increment of 5. In Fig. 4, the greater the number of IIEs, the longer the average delay of tasks. This is because as the number of IIEs increases, each IIE gets fewer uplink bandwidth, and each task gets fewer computing resources from the server. Therefore, the transmission and computing delay of each task increases, which results in an increase in the average delay of the tasks. In addition, as shown in Fig. 4, on average, the DDRM can reduce the average delay by 5% compared with the A3C scheme. In comparison with the URM and RRM schemes, the DDRM can reduce the average delay by 9% and 19%, respectively. In particular, due to the randomness of RRM, the decisions generated by RRM are not optimal and also have strong uncertainties. Therefore, the curve of RRM fluctuates. We conclude that DDRM outperforms A3C, URM and RRM, because the DDRM can integrate the current and future states of the whole system when optimizing the long-term average delay of tasks. Moreover, since DDRM uses two more target networks than A3C, it can optimize the long-term average delay of tasks more stably.

2) Impact of Computing Amount: We evaluate the impact of different computing amounts of tasks on the average delay. The number of IIEs is set to 25. The total computing capacity in the MEC server is set to 15 GHz. And the data size is a Gaussian distribution, with mean value of 6 MB.

Fig. 5 shows the average delay of tasks for DDRM, A3C, URM, and RRM with different computing amounts of tasks. The average computing amount of tasks is varied from 0.5 Gigacycles to 2 Gigacycles with an increment of 0.25 Gigacycles. It can be seen from Fig. 5 that the more computing amount, the longer the average delay of the tasks. The reason is that when the total computing capacity of the MEC server is fixed, as computing amount of each task increases, the computing delay of each task will be longer. Therefore, the average delay of tasks increases. Moreover, DDRM has a shorter average delay of tasks than A3C, URM and RRM. From Fig. 5, on average, the DDRM can reduce 6% of the average delay compared with the A3C scheme. In comparison with the URM and RRM schemes, the DDRM can reduce 10% and 20% of the average delay, respectively.

3) Impact of Data Size: We compare the impact of the different data sizes on the average delay. The number of IIEs is set to 25. The total computing capacity in the MEC server is set to 15 GHz. And the computing amount of tasks is a Gaussian distribution, with mean value of 1.25 Gigacycles.

Fig. 6 shows the average delay of tasks for DDRM, A3C, URM, and RRM with different data sizes. The data size is varied from 2 MB to 10 MB with an increment of 1 MB. In Fig. 6, the larger the data size, the greater the average delay of tasks. The transmission power is limited by the battery capacity of the IIE. Therefore, as the data size increases, the delay of each task also increases, which results in an increase in the average delay of all tasks. Furthermore, the DDRM achieves a shorter average delay of tasks than A3C, URM and RRM. As shown in Fig. 6, on average, the DDRM can reduce 5% of average delay compared to the A3C scheme. In comparison with the URM and RRM schemes, the DDRM can reduce the average delay by 9% and 16%, respectively.

4) Impact of Total Computing Capacity in The MEC Server: Finally, we consider the impact of different total computing capacities in the MEC server. The number of IIEs is set to 25. The computing amount and the data size of tasks follow Gaussian distributions, with an average of 1.25 Gigacycle and 6 MB, respectively.

Fig. 7 shows the average delay of tasks for DDRM, A3C, URM, and RRM with different total computing capacities in the MEC server. The computing capacity is varied from 5 to 25 with an increment of 5. In Fig. 7, the greater the computing capacity, the lower the average task delay. The reason can be explained as follows. A greater computing capacity means more computing resource allocated to the task, and thus the average

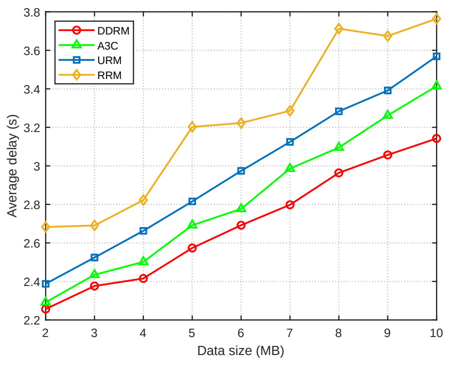


Fig. 6. The impact of task data size on average delay.

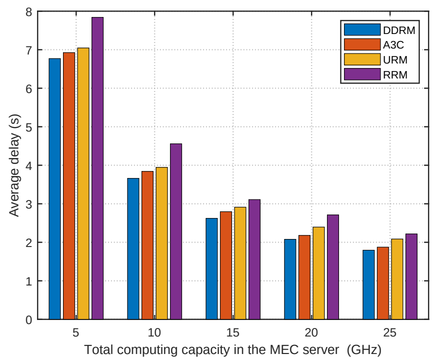


Fig. 7. The impact of total computing capacity in the MEC server.

delay becomes smaller. Moreover, the DDRM achieves a lower average delay of tasks than A3C, URM and RRM. As shown in Fig. 7, on average, the DDRM can reduce 4% of the average delay compared to the A3C scheme. In comparison with the URM and RRM schemes, the DDRM can reduce 8% and 17% of the average delay, respectively.

VI. Conclusion

In this paper, we have investigated the dynamic resource management problem of the joint transmission power control and computing resource allocation in IIoT. In order to minimize the long-term average delay, we have transformed the original optimization problem into a MDP problem. Due to the high dimensional continuity of action space and state space, we propose a DDRM algorithm based on DRL, which jointly optimizes the transmission power and computing resource. Simulation results show that compared with conventional A3C, URM and RRM algorithms, our DDRM algorithm can reduce the task delay effectively.

Acknowledgment

This work is partly supported by the National Natural Science Foundation of China (No. 61902029, No. 61872044), the Excellent Talents Projects of Beijing (No. 9111923401), the Scientific Research Project of Beijing Municipal Education Commission (No. KM202011232015), the Science and Technology Development Fund of Macau SAR under Grant 0162/2019/A3, and the FDCT-MOST Joint Project under Grant 066/2019/AMJ.

References

[1] N. Zhang, X. Fang, Y. Wang, S. Wu, H. Wu, D. Kar, and H. Zhang, "Physical layer authentication for internet of things via wfrft-based gaussian tag embedding," *IEEE Internet of Things Journal*, pp. 1–1, 2020, DOI: 10.1109/JIOT.2020.3001597.

[2] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, Opportunities, and Directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, Nov 2018.

[3] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying fog computing in industrial internet of things and industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4674–4682, Oct 2018.

[4] N. Zhang, R. Wu, S. Yuan, C. Yuan, and D. Chen, "Rav: Relay aided vectorized secure transmission in physical layer security for internet of things under active attacks," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8496–8506, Oct 2019.

[5] Y. Wu, K. Ni, C. Zhang, L. P. Qian, and D. H. K. Tsang, "NOMA-Assisted Multi-Access Mobile Edge Computing: A Joint Optimization of Computation Offloading and Time Allocation," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 12 244–12 258, Dec 2018.

[6] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial Offloading Scheduling and Power Allocation for Mobile Edge Computing Systems," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6774–6785, Aug 2019.

[7] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energy-efficient uav-assisted mobile edge computing: Resource allocation and trajectory optimization," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3424–3438, March 2020.

[8] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Performance Optimization for Blockchain-Enabled Industrial Internet of Things (IIoT) Systems: A Deep Reinforcement Learning Approach," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3559–3570, June 2019.

[9] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 319–333, Feb 2018.

[10] M. Li, J. Gao, L. Zhao, and X. Shen, "Deep reinforcement learning for collaborative edge computing in vehicular networks," *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2020, DOI: 10.1109/TCCN.2020.3003036.

[11] J. Wang, C. Jiang, K. Zhang, X. Hou, Y. Ren, and Y. Qian, "Distributed Q-Learning Aided Heterogeneous Network Association for Energy-Efficient IIoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2756–2764, April 2020.

[12] M. Z. Hasan and H. Al-Rizzo, "Optimization of Sensor Deployment for Industrial Internet of Things Using a Multiswarm Algorithm," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 344–10 362, Dec 2019.

[13] X. Lai, Q. Hu, W. Wang, L. Fei, and Y. Huang, "Adaptive Resource Allocation Method Based on Deep Q Network for Industrial Internet of Things," *IEEE Access*, vol. 8, pp. 27 426–27 434, 2020.

[14] N. Li, M. Xiao, L. Rasmussen, X. Hu, and V. Leung, "On Resource Allocation of Cooperative Multiple Access Strategy in Energy-efficient Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2020.

[15] B. Yang, X. Cao, X. Li, Q. Zhang, and L. Qian, "Mobile-Edge-Computing-Based Hierarchical Machine Learning Tasks Distribution for IIoT," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 2169–2180, March 2020.

[16] Z. Zhao, R. Zhao, J. Xia, X. Lei, D. Li, C. Yuen, and L. Fan, "A Novel Framework of Three-Hierarchical Offloading Optimization for MEC in Industrial IoT Networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5424–5434, Aug 2020.

[17] W. Chen, Z. Zhang, Z. Hong, C. Chen, J. Wu, S. Maharjan, Z. Zheng, and Y. Zhang, "Cooperative and Distributed Computation Offloading for Blockchain-Empowered Industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8433–8446, Oct 2019.

[18] H. Wu, X. Lyu, and H. Tian, "Online Optimization of Wireless Powered Mobile-Edge Computing for Heterogeneous Industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9880–9892, 2019.

[19] H. Zhou, N. Cheng, J. Wang, J. Chen, Q. Yu, and X. Shen, "Toward Dynamic Link Utilization for Efficient Vehicular Edge Content Distribution," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8301–8313, Sep. 2019.

[20] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, March 2018.

[21] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 726–738, Sep. 2018.

[22] H. Zhang, J. Guo, L. Yang, X. Li, and H. Ji, "Computation offloading considering fronthaul and backhaul in small-cell networks integrated with MEC," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, May 2017, pp. 115–120.

[23] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[24] H. Ye, G. Y. Li, and B. F. Juang, "Deep Reinforcement Learning Based Resource Allocation for V2V Communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, April 2019.

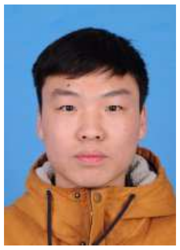
[25] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *2016 International Conference on Learning Representations (ICLR)*, May 2016, pp. 1–14.

[26] F. Meng, P. Chen, and L. Wu, "Power allocation in multi-user cellular networks with deep Q learning approach," in *2019 IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–6.

[27] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *2016 International conference on machine learning (ICML)*, June 2016, pp. 1928–1937.

Ying Chen (S'13-M'19) is an Associate Professor in Computer School, Beijing Information Science and Technology University, China. She received the Ph. D degree from Tsinghua University in 2017, and was a joint Ph. D student in University of Waterloo, Canada from 2016 to 2017. She is the recipient of the Best Paper Award at IEEE SmartIoT 2019, 2016 Google Ph. D Fellowship Award and 2014 Google Anita Borg Award, respectively. She serves/served the TPC member of IEEE HPCC, and PC member of IEEE Cloud, CollaborateCom, IEEE CPSCom, CSS, etc. She is also the reviewer of several journals such as IEEE Wireless Communications Magazine, IEEE Transactions on Dependable and Secure Computing, IEEE Internet of Things Journal, IEEE Transactions on Computers, IEEE Transactions on Cloud Computing, IEEE Transactions on Services Computing, etc. Her current research interests include Internet of Things, mobile edge computing, wireless networks and communications, machine learning, etc.

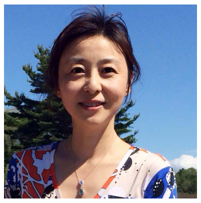
Zhiyong Liu M. S. candidate in the School of Computer Science from Beijing Information Science and Technology University, Beijing, China. His research interests include mobile edge computing and wireless networks.



Yongchao Zhang received the BEng degree in the school of Information and Communication Engineering from Beijing Information Science and Technology University in 2017. He is currently a graduate student in the school of Computer Science, Beijing Information Science and Technology University. His research interests include wireless networks and edge computing.

Yuan Wu (S'08-M'10-SM'16) received the PhD degree in Electronic and Computer Engineering from the Hong Kong University of Science and Technology in 2010. He is currently an Associate Professor with the State Key Laboratory of Internet of Things for Smart City, University of Macau and also with the Department of Computer and Information Science, University of Macau. During 2016-2017, he was a visiting scholar with Department of Electrical and Computer Engineering, University of Waterloo. His research interests include resource management for wireless networks, green communications and computing, mobile edge computing, and smart grids. He was a recipient of the Best Paper Award from the IEEE International Conference on Communications in 2016, and the Best Paper Award from IEEE Technical Committee on Green Communications and Computing in 2017. Dr. Wu served as the guest editors of IEEE Communications Magazine, IEEE Network, IEEE Transactions on Industrial Informatics, and IET Communications. He is currently on the Editorial Boards of IEEE Internet of Things Journal and China Communications.

Xin Chen received the Ph.D. degree in computer science from the Beijing Institute of Technology, Beijing, China. He is currently a Professor with the Computer School, Beijing Information Science and Technology University. His current research interests include performance evaluation of wireless networks. Dr. Chen received the Postdoctoral Fellowship in Computer Architecture from Tsinghua University in 2006. He is a Senior Member of the China Computer Federation (CCF), a member of the CCF Technical Committee of Theoretical Computer Science, and the CCF Technical Committee of Petri Nets.



Lian Zhao (S'99-M'03-SM'06) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada, in 2002. She joined the Department of Electrical, Computer, & Biomedical Engineering at Ryerson University, Toronto, Canada, in 2003 and has been a Professor since 2014. Her research interests are in the areas of wireless communications, resource management, mobile edge computing, caching and communications, and vehicular ad-hoc networks. She has been selected as an IEEE Communication Society (ComSoc) Distinguished Lecturer (DL) for 2020 and 2021; received the Best Land Transportation Paper Award from IEEE Vehicular Technology Society in 2016, Top 15 Editor Award in 2015 for IEEE Transaction on Vehicular Technology, Best Paper Award from the 2013 International Conference on Wireless Communications and Signal Processing (WCSP), and the Canada Foundation for Innovation (CFI) New Opportunity Research Award in 2005.