Task Offloading via Prioritized Experience-Based Double Dueling DQN in Edge-Assisted IIoT

Jiancheng Chi, *Member, IEEE*, Xiaobo Zhou, *Senior Member, IEEE*, Fu Xiao, *Senior Member, IEEE*, Yuto Lim, *Member, IEEE*, and Tie Qiu, *Senior Member, IEEE*

***Abstract*—In the Industrial Internet of Things (IIoT), Multi-access Edge Computing (MEC) emerges as a transformative paradigm for managing computation-intensive tasks, where task offloading plays an important role. However, due to the complex environment of IIoT, existing deep reinforcement learning-based schemes suffer from significant shortcomings in accuracy and convergence speed during model training when addressing the issue of task offloading. In this paper, to solve this problem, we propose an online task offloading scheme based on reinforcement learning, leveraging the double deep Q network (DQN) and dueling DQN with a prioritized experience replay mechanism, called the Prioritized experience-based Double Dueling DQN task offloading scheme (P-D3QN). P-D3QN enhances action selection accuracy using double DQN and mitigates Q-value overestimation by decomposing state and advantage using dueling DQN. Additionally, we adopt the prioritized experience replay mechanism to enhance the convergence speed of model training by selecting transitions that induce a higher training error between the evaluation network and the target network. Experimental results demonstrate that P-D3QN outperforms several state-of-the-art schemes, achieving a reduction of 21.0% in the average cost of the task and improving the completion rate of the task by 19.5%.**

***Index Terms*—Deep reinforcement learning, Industrial Internet of Things, multi-access edge computing, online task offloading.**

I. Introduction

As a specialized field within the scope of the Internet of Things (IoT), the Industrial Internet of Things (IIoT) integrates a variety of industrial equipment, machinery, production lines, and systems with the Internet through sensors, software, and other technological advancements, facilitating more efficient, automated, and intelligent operations [1]. In IIoT, a large number of sensors and devices are continuously generating significant amounts of data in real-time within the industrial operations [2]. The demand for immediate data analysis is critical to improve operational efficiency and support decision-making processes. However, transmitting all data to cloud-based systems would require extensive bandwidth and lead to considerable data transfer expenses. Multi-access Edge Computing (MEC), as an emerging data processing paradigm, enables processing closer to data sources, ensuring that only essential data is sent to cloud platforms for further analysis or decision-making optimization [3]. MEC facilitates faster data analysis and prompt feedback, which are essential for time-sensitive industrial applications, and also helps reduce bandwidth cost and alleviate traffic congestion [4].

In the MEC-supported IIoT environment, such as smart factories, offloading computational tasks to nearby MEC servers significantly enhances operational efficiency by reducing delay and conserving energy. For instance, in quality control, high-resolution cameras capture images along production lines, which are then processed using machine learning (ML) models on MEC servers to detect defects in real-time [5]. In predictive maintenance, sensors on machinery continuously send data to MEC servers where ML models analyze patterns to predict equipment failures before they occur, preventing costly downtimes [6]. Similarly, for automated robotics and automated guided vehicle (AGV) systems, navigation and obstacle avoidance updating tasks are offloaded to MEC servers, allowing for seamless operation and efficient handling, ensuring that robots and AGVs operate efficiently within the dynamic environment [7]. By offloading these critical computational tasks, smart factories can maintain high throughput and quality standards while optimizing the use of energy and reducing operational costs.

The crucial issue of task offloading involves determining whether to execute computational tasks on local devices (such as sensors, industrial robots or controllers) or to offload them to a suitable server selected from the available MEC servers [8]. This decision is influenced by various factors, including the computational capacity of the local devices and servers, the time and bandwidth costs of data transmission, the energy efficiency requirements of the tasks, sensitivity to delay, etc. In the dynamic IIoT environment, an appropriate task offloading strategy is crucial to optimize system performance, response time, and resource utilization efficiency.

Some existing studies tried to address the issue of task offloading using traditional approaches such as linear programming and game theory [9], [10], [11], [12]. However, due to the high

Received 26 May 2024; revised 22 July 2024; accepted 14 August 2024. Date of publication 30 August 2024; date of current version 5 November 2024. This work was supported in part by the National Science Fund for Distinguished Young Scholars of China under Grant 62325208 and Grant 62125203, in part by Joint Funds of the National Natural Science Foundation of China under Grant U2001204, and in part by National Natural Science Foundation of China under Grant 62272339 and Grant 62072330. Recommended for acceptance by A. Garcia-Saavedra. *(Corresponding authors: Xiaobo Zhou; Tie Qiu.)*

Jiancheng Chi, Xiaobo Zhou, and Tie Qiu are with the College of Intelligence and Computing, Tianjin University, Tianjin 300350, China (e-mail: chijiancheng@tju.edu.cn; xiaobo.zhou@tju.edu.cn; qiutie@ieee.org).

Fu Xiao is with the School of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210023, China (e-mail: xiaof@njupt.edu.cn).

Yuto Lim is with the Graduate School of Advanced Science and Technology, Japan Advanced Institute of Science and Technology, Nomi 923-1292, Japan (e-mail: ylim@jaist.ac.jp).

Digital Object Identifier 10.1109/TMC.2024.3452502

computational burden and the challenge of rapidly adapting to dynamic changes in the network, these task offloading methods based on game theory and optimization may encounter limitations in the IIoT environment. Machine learning (ML) can extract features from extensive data and adjust model parameters through training, establishing a task-offloading model in complex environments [13]. Some supervised machine learning approaches have been adopted to solve the task offloading problem [14], [15], [16], [17], but they require large amounts of labeled data for training and can struggle to adapt to dynamic environmental changes. In contrast, deep reinforcement learning (DRL) can learn through interaction with the environment without relying on prelabeled data, and is capable of adapting to dynamic environmental changes, making it a better choice to solve task offloading problems in a dynamic and complex environment [18], [19]. As a result, DRL-based task offloading solutions have attracted a lot of attentions [20], [21], [22], [23], [24], [25], [26], [27], [28], [29].

However, existing DRL-based solutions exhibit considerable shortcomings that prevent practical deployment in real-world environments. First, existing DRL approaches struggle to maintain high accuracy in complex environments. In smart factories, task offloading decisions need to be highly accurate to ensure the continuity of the production process and the quality of products. Nevertheless, current methods experience a notable decrease in decision accuracy when confronted with diverse environments and uncertain task loads due to model drawbacks such as low estimation precision, thereby increasing the task completion cost and reducing task completion rate. Second, many existing DRL solutions often converge too slowly in complex and dynamic IIoT environments [28]. The slow convergence rate results in the system's inability to provide reliable decisions during the initial deployment phase, affecting production efficiency and equipment utilization. For example, in smart factories, when equipment fails or production demands change suddenly, the system may not offload tasks in a timely manner, leading to task backlogs and production interruptions. Additionally, the continuously changing states of production lines and equipment exacerbate the issue, as slow convergence rates prevent the system from promptly adapting to these unexpected conditions. In IIoT, only by comprehensively improving the convergence speed and accuracy of DRL solutions can the overall efficiency of the system be effectively enhanced.

In this paper, we propose an online task offloading scheme for IIoT support by MEC that utilizes the double deep Q-network (DQN) and the dueling DQN with a prioritized experience replay (PER) mechanism, called **P**rioritized experience-based **D**ouble **D**ueling **DQN** task offloading scheme (P-D3QN). Unlike other DRL-based methods, in P-D3QN, we have comprehensively considered the trade-off between task completion delay and device energy consumption, and we utilized double DQN to reduce the overestimation issue in Q-value estimation. To enhance the adaptability of the trained model to dynamic environments, we introduce dueling DQN to increase the accuracy of estimating the action values for task offloading decisions, thereby accelerating the convergence speed of the task offloading model, which are particularly pivotal in the IIoT environment characterized by stringent task delay requirements and significant dynamism. Furthermore, to further accelerate the model convergence speed, we have adopted the PER mechanism for P-D3QN. This mechanism effectively retains previous task offloading information by the model, thereby enhancing the adaptability of the trained model to the complex environment. The main contributions of this paper are summarized as follows.

- We establish a system model and formulated the task offloading issue as an optimization problem, considering the tradeoff between task delay and energy consumption constraints in the IIoT environment support by MEC.
- We propose P-D3QN, a deep reinforcement learning-based online task offloading scheme for IIoT support by MEC. In P-D3QN, we combine double DQN and dueling DQN to reduce the issue of overestimation in estimating the Q-value, improving the task completion rate and decreasing the average task cost. Moreover, prioritized experience replay mechanism is adopted to further accelerate the convergence speed of trained task offloading model.
- The effectiveness of proposed P-D3QN is demonstrated by experimental results, which show an obvious advantage in both the average task cost and the completion rate of tasks, achieving a 21.0% reduction in average task cost and enhancing the task completion rate by 19.5% compared with the state-of-the-art scheme.

The rest of this paper is organized as follows. Section II presents a review of the literature on DRL-based task offloading schemes. Section III introduces the system model and formulates the optimization problem. In Section IV, we propose the BRL-based task offloading scheme P-D3QN. In Section V, we discuss the experimental results. Finally, we conclude this paper in Section VI.

II. Related Work

DRL is an advanced machine learning method that combines the capabilities of deep learning and reinforcement learning. It utilizes deep neural networks as function approximators to learn and represent the optimal policy or value function in a reinforcement learning setup. In a typical DRL structure, there are several hidden layers located between the input layer and the output layer, all of which contain numerous neuron nodes. This multi-layer neural network structure allows DRL to effectively learn the nonlinear relationship between states and actions, and thus derive optimal policies in complex environments [30]. Given sufficient data, DRL can train models that excel in decision-making problems, such as task offloading. The main strengths of DRL lie in its powerful learning capacity and high precision in specific scenarios. However, DRL requires considerable computational resources, and building models can be time-consuming. The performance of the DRL is also largely dependent on the volume and quality of the interaction data it receives. With abundant and diverse interaction data, a DRL model can achieve high accuracy in policy optimization. Otherwise, performance could suffer significantly [31].

Numerous DRL-based task offloading schemes have recently been proposed in the literature. For instance, Huang et al. [21]

proposed a deep reinforcement learning-based framework for task offloading and resource allocation under event channel conditions. However, the model assumes stable wireless power transfer and continuous energy availability, which may limit the deployment in more unstable contexts. Bi et al. [22] proposed LyDROO with Lyapunov optimization to solve per-frame MINLP problems with low computational complexity. While the Lyapunov optimization technique aids in maintaining system stability, it may not optimally handle unexpected rapid fluctuations in network conditions. Peng et al. [23] proposed an innovative computation offloading strategy for IIoT using a DRL-based approach with a novel metric Age of Information (AoI). However, the computational overhead associated with maintaining AoI metrics in real-time may limit its deployment in environments with limited computational resources. Fan et al. [24] proposed a sophisticated joint task offloading and resource allocation scheme tailored for accuracy-aware machine-learning-based IIoT applications in an edge–cloud network architecture. The biggest disadvantage of this work is that the complexity of managing the inference accuracy alongside task offloading could lead to significant computational overhead. In order to improve the adaptability of existing reinforcement learning schemes to new environments, Wang et al. [25] designed a task offloading method based on meta-reinforcement learning, which can quickly adapt to new environments with a small number of gradient updates and transitions. However, meta-reinforcement learning demands a higher degree of diversity in task data, which may be constrained by data privacy issues in IIoT environments.

Some studies focus on addressing specific technical challenges and optimizing performance in IIoT environments by utilizing multi-agent or distributed deep reinforcement learning techniques to solve the task offloading problem. Ren et al. [26] proposed a computation offloading strategy for the IIoT enabled by fog computing, utilizing a multi-agent DRL approach to optimize energy consumption across a network of IIoT devices and fog access points. For this work, the reliance on offline training might introduce challenges in environments where model re-training is constrained. Zhang et al. [27] proposed a DRL-based framework based on an improved soft actor-critic algorithm for cooperative partial task offloading and resource allocation in IIoT environments. Nevertheless, this work faces possible scalability issues when applied to extensive network architectures or when operated under stringent time constraints. Tang et al. [28] proposed a distributed algorithm based on deep reinforcement learning to solve the problem of minimizing the expected long-term cost of task offloading, but the reliance on accurate load predictions and the need for frequent communication for updating learning models could pose challenges in dynamic and unstable network conditions. Zhou et al. [29] proposed an online learning framework called LFSC to guide task offloading with a small cell network performance guarantee, but the physical limitations of 5G high frequency bands, such as millimeter-wave channel sparsity and beamforming technique, constrain the number of connections each small cell node can manage.

Although DRL-based research has been extensively conducted, how to achieve fast convergence speed and improve accuracy in task offloading model training presents significant challenges. Unlike previous DRL-based approaches, we introduce a novel scheme, P-D3QN, in this work to enhance fast convergence and accuracy for task offloading in IIoT supported by MEC. P-D3QN takes advantage of a blend of double DQN and dueling DQN architectures, enhanced by a PER mechanism. This hybrid design benefits from the improved action selection accuracy of double DQN and the capacity to decompose state value from action advantage of dueling DQN, thus refining the decision-making process to better fit IIoT support by MEC systems. Furthermore, the PER mechanism incorporated within P-D3QN selectively revisits transitions that present a significant difference between the evaluation and target networks, thus expediting the convergence of the model. This mechanism retrieval of experiences ensures that our scheme remains responsive to changes in the environment, facilitating a faster and more precise convergence.

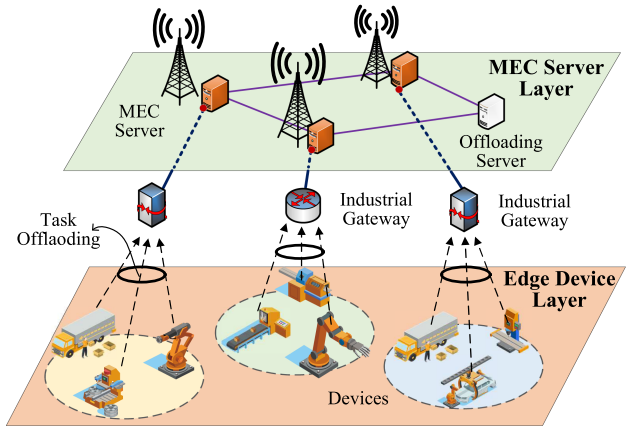

Fig. 1. An illustration of IIoT supported by MEC.

III. System Model and Problem Formulation

A. System Overview

Under the assumption of generality, we envision an IIoT support by the MEC system as shown in Fig. 1. The system consisting of the edge device (ED) layer and the MEC server (MS) layer. In the depicted architecture, the MSs communicate with the EDs through 5G enhanced base stations. The base stations, acting as intermediaries, can augment the network's capability to transmit massive amounts of data generated by EDs, facilitating high-speed, low-latency connections. The system time line is divided into timeslots $\mathcal{T} = \{1, 2, \ldots, t, \ldots \Delta\}$, where t represents the t-th task generating timeslot. Denote the set of EDs is $\mathcal{N} = \{1, 2, \ldots, N\}$.

In our system, we consider a scenario where EDs generate computationally intensive tasks. These tasks can either be processed locally on the ED or offloaded to a proximate MS. The computational capability of each ED, denoted by f_n, is constrained due to limited local resources. The total energy of the ED can be denoted as e_n. When the task is generated, the location of ED n is characterized by a coordinate, $\mathbf{l}_{n,t} = (\mathbf{x}_{n,t}, \mathbf{y}_{n,t})$. We emphasize the premise that the velocities of the EDs are relatively slow [32]. Consequently, the distance

traversed within each timeslot can be considered minimal, following the observations made in the prior literature. Denote the set of MSs is $\mathcal{M} = \{1, 2, \ldots, M\}$, in the envisioned system, the MSs are stationed at constant locations with zero altitudes on the terrestrial plane. The position of MS m is denoted as $\mathbf{l}_m = (\mathbf{x}_m, \mathbf{y}_m)$. The height of the MS antenna is denoted by h_m. Each MS m is endowed with a certain amount of computational resources f_m, enabling MS m to provide offloading services to EDs that are within its communication range. The radius of the communication range of MS m is CR_m. Furthermore, task offloading decision is constrained to align with a long-term energy budget for the MS m, denoted as e_m .

We deploy the P-D3QN scheme on the task offloading server shown in Fig. 1. This server is responsible for collecting global system environment information and making offloading decisions using the P-D3QN-trained model when a task arrives. The offloading decision is then sent to the device that generated the task. The device executes the decision, choosing whether to perform the task locally or transmit it to the designated server for processing. After the decision is executed, the system environment changes, and the server collects the global system environment information again, waiting for the next task to arrive and making decisions. By interactions with the environment, the task-offloading model gradually learns how to get a better performance on decision-making.

B. Task Model

In our task model, delay and reliability are critical for the effective functioning of IIoT systems. The delay in this context encompasses the entire sequence from task transmission through processing to the acquisition of results, which is crucial for managing complex computational tasks such as updates to neural networks. Moreover, our model assumes near-perfect data transmission reliability, offloading tasks in a manner that reflects the high reliability of contemporary edge computing powered by 5G technologies [33], [34]. Consequently, our task model does not consider the scenarios where task transmissions fail and require retransmission.

The task generation probability is a statistical measure that determines how frequently new tasks are generated and introduced into the system. Specifically, if task generation probability is equal to λ, it quantifies that a new task will be generated in timeslots with the probability λ. We denote the generated computation-intensive task at a timeslot t by ED n as $J_{n,t}$. We encapsulate the task $J_{n,t}$ within a three-tuple of parameters $J_{n,t} = (d_{n,t}, c_{n,t}, \tau_{n,t})$. Specifically, $d_{n,t}$ indicates the data volume of task required for the processing. The amount of computation of the task, i.e., the total CPU cycles required to process $J_{n,t}$, is represented by $c_{n,t}$. Lastly, $\tau_{n,t}$ signifies the maximum delay that the task can tolerate.

Each ED n has the option to execute its task locally or offload it to the MS. We define a vector to capture the task-offloading decision from ED to MS. For each ED $n \in \mathcal{N}$ and MS $m \in \mathcal{M}$, the offloading decision of the task $J_{n,t}$ from ED n to MS m is denoted as $v_{m,n,t}$. The collection of these decisions forms a m-dimensional vector $v_{n,t} = \{v_{1,n,t}, v_{2,n,t}, \ldots, v_{m,n,t}\}$. $v_{m,n,t}$ is a binary decision variable denoted as

$$v_{m,n,t} = \begin{cases} 1, & \text{ED } n \text{ offloads } J_{n,t} \text{ to MS } m \text{ in timeslot } t, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Importantly, tasks are atomic and cannot be further divided [35]. We further introduce a binary decision variable, $b_{n,t}$, to capture whether task $J_{n,t}$ is processed locally or on an MS:

$$b_{n,t} = \sum_{m=1}^{M} v_{m,n,t} = \begin{cases} 1, & J_{n,t} \text{ is offloaded to an MS in} \\ & \text{timeslot } t, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

C. Communication Model

Given the potential for significant volumes of computational input data to be transferred from the EDs to the MSs, a considerable amount of wireless spectrum is required. We make the assumption that orthogonal frequency division multiple access (OFDMA) is employed as the uplink multiple access scheme, which is widely recognized in numerous communication standards [35], [36]. for each MS, the bandwidth resources B are subdivided into K equal sub-channels denoted by $\mathcal{K} = \{1, 2, \ldots, K\}$. Each sub-channel has the size of $\bar{B} = B/K$ and can only be allocated to at most one ED.

The OFDMA technique efficiently mitigates intracell channel interference between EDs that transmit data to the same MS. However, when a multitude of EDs employ the same sub-channel frequency of different MSs, the system may be subject to intercell co-channel interference. Denote $\Upsilon_{m,n,k,t}$ as the interference between cells in the co-channel when MS m assign a sub-channel k to ED n for task transmission at timeslot t, which can be calculated by

$$\Upsilon_{m,n,k,t} = \sum_{i \in \mathcal{M} \setminus \{m\}} \sum_{j \in \mathcal{N} \setminus \mathcal{N}_m^t} v_{i,j,t} \cdot P_j^T \cdot H_{i,j,k,t}. \tag{3}$$

Denote $i \in \mathcal{M} \setminus \{m\}$ and $j \in \mathcal{N} \setminus \mathcal{N}_m^t$ as the set of MSs excluding MS m, and the set of EDs not within the signal coverage of MS m, respectively. P_j is the transmission power of ED j and $H_{i,j,k,t}$ is the channel power gain when sub-channel k is allocated from MS i to ED j in the timeslot t. The channel power gain is calculated by

$$H_{i,j,k,t} = \alpha \cdot \left(\sqrt{||\mathbf{l}_i - \mathbf{l}_{j,t}||^2 + h_j^2} \right)^{-\beta} \cdot h_{i,j,k,t}, \tag{4}$$

where α is the path loss coefficient and β is the path loss exponent respectively. $\sqrt{||\mathbf{l}_i - \mathbf{l}_{j,t}||^2 + h_j^2}$ is the distance between the antenna of MS i and ED j in timeslot t. $h_{i,j,k,t}$ is the small-scale fading channel power gain.

Then the definition of the Signal-to-Interference-plus-Noise Ratio (SINR) from ED n to MS m on sub-channel k in timeslot t is as follows [37], where σ^2 is the noise power:

$$SINR_{m,n,k,t} = \frac{P_n^T \cdot H_{m,n,k,t}}{\Upsilon_{m,n,k,t} + \sigma^2}. \tag{5}$$

According to Shannon's theorem [38], we are able to compute the transmission rate between MS m and ED n at timeslot t as

TABLE I
KEY NOTATIONS

Notation	Description	Notation	Description
$\mathcal{M}$	Set of all MSs	$v_{m,n,t}$	Binary variable denotes if $J_{n,t}$ is offloaded to MS m
$\mathcal{N}$	Set of all EDs	$b_{n,t}$	Binary variable denotes if $J_{n,t}$ is offloaded
M	Number of MSs	B	Total bandwidth resources
N	Number of EDs	$\bar{B}$	Bandwidth of sub-channels
m	Index of MSs	$\Upsilon_{m,n,k,t}$	Inter-cell co-channel interference
n	Index of EDs	P_n^T	Transmission power of ED n
$\mathcal{T}$	Set of timeslots	$H_{i,j,k,t}$	Channel power gain
t	Index of timeslots	σ^2	Noise power
f_n	Computational capability of ED n	$SINR_{m,n,k,t}$	Signal-to-interference-plus- noise Ratio
e_n	Total energy of ED n	$R_{m,n,t}$	Transmission rate between MS m and ED n
$\mathbf{l}_{n,t}$	Location of ED n in timeslot t	$T_{m,n,t}^{tr}$	Transmission delay from ED n to the MS m
$\mathbf{l}_m$	Location of MS m	$E_{m,n,t}^{tr}$	Energy consumption of task transmission
h_m	Antenna height of MS m	$E_{n,t}^{lc}$	Energy consumption of task processing locally
e_m	Long-term energy budget for MS m	$T_{n,t}^{lc}$	Total delay of task processing locally
f_m	Total computational capability of MS m	$f_{m,n,t}$	Allocated computational capability for $J_{n,t}$
CR_m	Communication range radius of MS m	φ_m	Occupied computational capability ratio of MS m
$J_{n,t}$	Task generated by ED n at timeslot t	$T_{m,n,t}^{rc}$	Computation delay for $J_{n,t}$ processed on MS m
$d_{n,t}$	Data volume of task $J_{n,t}$	$E_{m,n,t}^{rc}$	Energy consumption of ED n in idle state
$c_{n,t}$	CPU cycles required per bit to process $J_{n,t}$	$C_{n,t}^{lc}$	Weighted-cost of ED n for processing $J_{n,t}$ locally
$\tau_{n,t}$	Maximum delay task $J_{n,t}$ can tolerate	$C_{m,n,t}^{rc}$	Weighted-cost of ED n for offloading $J_{n,t}$ to MS m

follows:

$$R_{m,n,t} = \bar{B} \cdot log_2 \left(1 + SINR_{m,n,k,t}\right). \tag{6}$$

The transmission delay incurred during the task offloading process from ED n to MS m at a given timeslot t can be calculated by

$$T_{m,n,t}^{tr} = d_{n,t}/R_{m,n,t}. \tag{7}$$

Denote the power consumption of ED n in transmission state as P_n^C. The energy consumption of task transmission can be calculated by

$$E_{m,n,t}^{tr} = P_n^T \cdot d_{n,t}/R_{m,n,t}. \tag{8}$$

D. Computation Model

Based on whether ED offloads its task to MSs for processing, there are two modes of computation: local computation and remote computation.

1) Local Computation: When the task $J_{n,t}$ is computed locally, the predefined system and task model give the CPU cycles required for task $d_{n,t}c_{n,t}$ and the computational capacity of the device f_n. Then the delay incurred during local computation can be expressed as:

$$T_{n,t}^{lc} = d_{n,t}c_{n,t}/f_n. \tag{9}$$

We define the power consumption of ED n in this local computation state as P_n^C. Therefore, the energy consumption of task computation in ED n in timeslot t is as follows:

$$E_{n,t}^{lc} = P_n^C \cdot T_{n,t}^{lc} = P_n^C \cdot d_{n,t}c_{n,t}/f_n. \tag{10}$$

2) Remote Computation: When local computational capacity is insufficient to meet the deadline requirement of a task, it requires offloading the task to a proximate server. Given that MSs typically possess multiple cores that can simultaneously process tasks, this powerful parallel processing capability combined with the maturing technology of virtual containers essentially renders the queuing delay negligible [39], [40].

If the task $J_{n,t}$ generated by ED n at timeslot t need to be offloaded to MS m, the allocation of computational capabilities should first be calculated. Denote the remaining computational capability of MS m as f_m^{re}, then the occupied computational capability ratio φ_m of MS m is

$$\varphi_m = 1 - \frac{f_m^{re}}{f_m}. \tag{11}$$

Based on the occupied computational capability ratio, the computational capability $f_{m,n,t}$ allocated from MS m to task $J_{n,t}$ is

$$f_{m,n,t} = \min\left\{\left(\frac{1}{2e^{\varphi_m}} + 1\right)\frac{d_{n,t}c_{n,t}}{\tau_{n,t} - T_{m,n,t}^{tr}}, f_m^{re}\right\}, \tag{12}$$

where $\frac{d_{n,t}c_{n,t}}{\tau_{n,t} - T_{m,n,t}^{tr}}$ represents the stringent minimal computational capability required. The coefficient $\frac{1}{2e^{\varphi_m}} + 1$ ensures a judicious range for the allocated computational capability.

Leveraging the allocated computational resources, the task $J_{n,t}$ is processed within the MS m. The comprehensive task computation delay for $J_{n,t}$ processed on MS m during the timeslot t is succinctly articulated as follows:

$$T_{m,n,t}^{rc} = \frac{d_{n,t}c_{n,t}}{f_{m,n,t}}. \tag{13}$$

Assuming that ED n remains in an idle state while awaiting the execution results from MS, we define the power consumption of ED n in this idle state as P_n^I. The resultant energy consumption of ED n can be expressed as

$$E_{m,n,t}^{rc} = P_n^I \cdot T_{m,n,t}^{rc} = P_n^I \cdot \frac{d_{n,t}c_{n,t}}{f_{m,n,t}}. \tag{14}$$

E. Problem Formulation

For the remote computation, denote the total remote processing time $T^{rp}_{m,n,t}$ and the remote processing energy consumption $T^{rp}_{m,n,t}$ as follows:

$$T^{rp}_{m,n,t} = T^{rc}_{m,n,t} + T^{tr}_{m,n,t}, \tag{15}$$

$$E^{rp}_{m,n,t} = E^{rc}_{m,n,t} + E^{tr}_{m,n,t}, \tag{16}$$

where $T^{rc}_{m,n,t}$ represents the time taken for remote computation at the MS m, $T^{tr}_{m,n,t}$ represents the transmission time required to send the task data from ED n to MS m, $E^{rc}_{m,n,t}$ denotes the energy consumed for computing the task remotely at the MS m, and $E^{tr}_{m,n,t}$ denotes the energy consumed for transmitting the task data from from ED n to MS m.

Next, we need to perform a weighted sum of energy consumption and processing time, considering both local and remote computation. To ensure that the impact of energy consumption and processing time on the weighted sum is balanced, we apply linear normalization to energy consumption and processing time. For local computation, let $\widetilde{T}^{lc}_{n,t}$ and $\widetilde{E}^{lc}_{n,t}$ denote the normalized processing time and energy consumption of local computation, respectively. Then we have:

$$\widetilde{T}^{lc}_{n,t} = \frac{T^{lc}_{n,t}}{E^{lc}_{n,t} + T^{lc}_{n,t}} = \frac{1}{P^C_n + 1}, \tag{17}$$

$$\widetilde{E}^{lc}_{n,t} = \frac{E^{lc}_{n,t}}{E^{lc}_{n,t} + T^{lc}_{n,t}} = \frac{P^C_n}{P^C_n + 1}. \tag{18}$$

According to previous models, the weighted cost of processing $J_{n,t}$ locally by ED n is:

$$C^{lc}_{n,t} = \alpha_1 \widetilde{T}^{lc}_{n,t} + (1-\alpha_1)\widetilde{E}^{lc}_{n,t}. \tag{19}$$

For remote processing, let $\widetilde{T}^{rp}_{m,n,t}$ and $\widetilde{E}^{rp}_{m,n,t}$ denote the normalized processing time and energy consumption of remote computation, respectively. Then we have:

$$\widetilde{T}^{rp}_{m,n,t} = \frac{T^{rc}_{m,n,t} + T^{tr}_{m,n,t}}{(1+P^I_n)T^{rc}_{m,n,t} + (1+P^T_n)T^{tr}_{m,n,t}}, \tag{20}$$

$$\widetilde{E}^{rp}_{m,n,t} = \frac{P^I_n \cdot T^{rc}_{m,n,t} + P^T_n \cdot T^{tr}_{m,n,t}}{(1+P^I_n)T^{rc}_{m,n,t} + (1+P^T_n)T^{tr}_{m,n,t}}. \tag{21}$$

Then the weighted cost of offloading $J_{n,t}$ to the MS m by ED n is:

$$C^{rc}_{m,n,t} = \alpha_2 \widetilde{T}^{rp}_{m,n,t} + (1-\alpha_2)\widetilde{E}^{rp}_{m,n,t}. \tag{22}$$

The parameters α_1 and α_2 are the weights in the cost functions for local and remote computations, respectively, and are critical for balancing the trade-offs between delay and energy consumption and optimizing system performance. The selection of these weights is strategically aligned with the system's operational priorities. In environments where rapid response is crucial, such as real-time monitoring or emergency scenarios, higher values of α_1 or α_2 are chosen to minimize delay. Conversely, in situations where energy conservation is paramount, particularly in devices with limited power availability, lower values are preferable to enhance energy efficiency. Therefore, the values of α_1 and α_2 require evaluation based on real-time system performance and environmental conditions. Furthermore, if necessary, Dynamic adjustment of α_1 and α_2 can be done to respond to fluctuations in network conditions, workload distribution, and device status, thereby meeting the diverse and evolving demands of IIoT environments.

The total cost for the system can be defined as the aggregated cost, which is determined by the weighted sum of the costs for all EDs associated with different MSs:

$$C_{tot} = \sum_{t=1}^{\Delta}\sum_{n=1}^{N}\sum_{m=1}^{M}\left[b_{n,t}\cdot C^{rc}_{m,n,t} + (1-b_{n,t})\cdot C^{lc}_{n,t}\right]. \tag{23}$$

In Equation (23), $b_{n,t}$ is defined in (2) as a binary decision variable that determines the offloading strategy for task $J_{n,t}$. This variable essentially dictate whether to utilize edge capabilities or leverage centralized computing resources to optimize performance and resource allocation across the network. Our goal is to minimize the total cost and we present the following formal mathematical problem:

$$\begin{aligned}
\mathbf{P}: \min_{\{v_{m,n,t}\}} \quad & C_{tot} \\
s.t. \quad & \mathbf{C1}: v_{m,n,t} \in \{0,1\},\ m \in \mathcal{M},\ n \in \mathcal{N},\ t \in \mathcal{T}, \\
& \mathbf{C2}: (1-b_{n,t})\cdot T^{lc}_{n,t} + b_{n,t}\cdot T^{rp}_{m,n,t} \le \tau_{n,t}, \\
& \mathbf{C3}: \sum_{t\in\mathcal{T}}((1-b_{n,t})E^{lc}_{n,t} + b_{n,t}\cdot E^{rp}_{m,n,t} \le e_n, \\
& \mathbf{C4}: \sum_{n\in S_m(t)} f_{m,n,t} \le f_m.
\end{aligned} \tag{24}$$

In the constraints, **C1** denotes the offloading decision for the task being a binary variable. **C2** ensures that the task processing time, whether locally $T^{lc}_{n,t}$ or remotely $T^{rp}_{m,n,t}$ on MS m, does not exceed the task deadline $\tau_{n,t}$. **C3** denotes that the total energy consumption to process tasks in a certain period for the ED n is less than the available energy e_n. Let $S_m(t)$ be the set of indexes of EDs whose tasks are being processed by MS m at timeslot t, and **C4** asserts that the total computational asserts that the total computational capacity allocated to the tasks being processed on MS m is less than or equal to the total available computational capacity of MS m.

In the context of MEC-enabled IIoT environments, the constraints of our model are particularly crucial. **C2** ensures real-time processing capabilities for IIoT applications such as automated quality control or predictive maintenance, where delays can lead to production downtime and increased costs. **C3** ensures energy efficiency for sustaining long-term operations of IIoT devices that often operate in remote or challenging conditions without frequent maintenance. **C4** manages computational resources effectively, preventing server overloads that can cause data processing bottlenecks in IIoT environments that depend on continuous data flow for operational decision-making.

IV. Deep Reinforcement Learning-Based Task Offloading Scheme (P-D3QN)

The optimization problem **P** involves numerous parameters and variables such as the location, computational capabilities, and energy of the EDs and MSs, resulting in its high complexity. In this section, we propose P-D3QN to efficiently solve the minimization problem **P**. We offer an in-depth illustration of P-D3QN's fundamental mechanism, presenting an efficient scheme for addressing the decision-making process of task offloading in IIoT support by MEC systems.

A. Markov Decision Process of Task Offloading Problem

The problem **P** can be transferred as a Markov Decision Process (MDP) with a 3-tuple of environment state, agent action, and obtained reward.

1) Environment State: The environment state of the task $J_{n,t}$ generated by ED n in timeslot t, denoted as $s_{n,t}$, encompasses the relevant attributes of the task and the arguments related to the system environment. At timeslot $t \in \mathcal{T}$, the task generated by ED n ($n \in N$) is denoted as $J_{n,t} = (d_{n,t}, c_{n,t}, \tau_{n,t})$, where $d_{n,t}$, $c_{n,t}$, and $\tau_{n,t}$ indicates the data volume of task, CPU cycles required, and the maximum tolerable delay, respectively. When a task is generated (that is, at the start of a specific timeslot), the coordinate $\mathbf{l}_{n,t} = (\mathbf{x}_{n,t}, \mathbf{y}_{n,t})$ of the ED n can be obtained. The computational capability of each ED is f_n, is constrained due to limited local resources. The remaining energy of the ED is e_n^{re}.

At the beginning of each step, the coordinate $\mathbf{l}_m = (\mathbf{x}_m, \mathbf{y}_m)$ ($m \in M$) of MS m can be obtained. The remaining computational capability of MS m is f_m^{re}. It is assumed that each ED can discern the position of each MS. The transmission rate $R_{m,n,t}$ between MS m and ED n at timeslot t can be calculated by (6).

Therefore, the environment state of $J_{n,t}$ by ED n in the timeslot t can be specified by $s_{n,t} = \{d_{n,t}, c_{n,t}, \tau_{n,t}, \mathbf{x}_{n,t}, \mathbf{y}_{n,t}, f_n, e_n^{re}, \mathbf{x}_m, \mathbf{y}_m, f_m^{re}, R_{m,n,t}\}$. The set of states is denoted as $\mathcal{S}$.

2) Agent Action: Based on the above elaboration, the agent selects actions according to the current environment state $s_{n,t}$ [41]. We denote the action for the task $J_{n,t}$ in timeslot t as $a_{n,t} = \{b_{n,t}, v_{m,n,t}\}$. Here, $b_{n,t}$ signifies the decision about offloading the current task. More explicitly, if the task is decided to process the task $J_{m,n,t}$ locally, then $b_{n,t} = 0$, otherwise, $b_{n,t} = 1$. Subsequently, the value of $v_{m,n,t}$ delineates the selection of MS m to process the offloaded task. We have $v_{m,n,t} = 1$ if the task $J_{m,n,t}$ is transferred to MS m for processing; otherwise, $v_{m,n,t} = 0$.

To ensure that these decisions are consistent and error-free, $b_{n,t}$ is defined as the sum of all $v_{m,n,t}$ across MSs for each task at each timeslot, which has been defined in Eq. (2). This definition guarantees that $b_{n,t} = 1$ if and only if one $v_{m,n,t}$ is set to 1, meaning exactly one MS is selected for offloading. If all $v_{m,n,t} = 0$, indicating no MS has been selected, $b_{n,t}$ automatically sets to 0, ensuring the task is processed locally. Denote the set of available actions as $\mathcal{A}$.

3) Obtained Reward: Given the environment state $s_{n,t}$, the agent interacts with the system environment using the offloading scheme $a_{n,t}$, receiving an instant reward [18]. Denote the instant reward for the task $J_{n,t}$ of ED n as $r_{n,t}$. The design of rewards is closely related to cost C_{tot}, as our objective is to minimize the overall cost of IIoT support by the MEC system. More specifically, there is an inverse relationship between cost and reward $r_{n,t}$, which means that higher costs are associated with lower rewards. Denote the instant reward for the task $J_{n,t}$ under the action $a_{n,t}$ as follows, where C is a constant.

$$r_{n,t} = C - \left[b_{n,t} \cdot C_{m,n,t}^{rc} + (1 - b_{n,t}) \cdot C_{n,t}^{lc}\right]. \tag{25}$$

Denote π as a mapping from state to action, that is, π: $\mathcal{S} \rightarrow \mathcal{A}$. According to the Bellman equation, our aim is to find an optimal policy π^* for all tasks so that the long-term expected reward is maximized, i.e.,

$$\pi^* = \arg \max_{\pi} E\left[\sum_{t \in \mathcal{T}} \gamma^{t-1} r_{n,t}\right], \tag{26}$$

where γ is the discount factor.

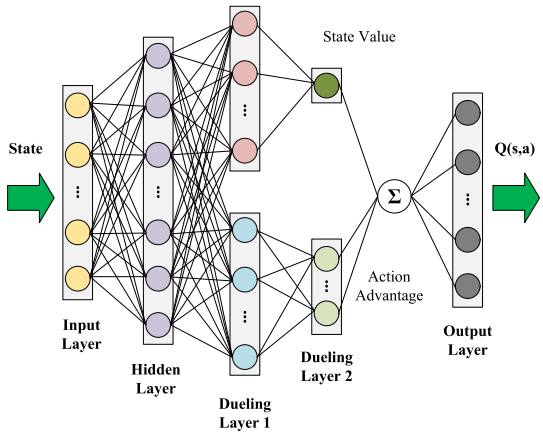

Fig. 2. The neural network structure based on dueling DQN.

B. Neural Network Structure Based on Dueling DQN

In P-D3QN, the main role of the neural network is to utilize the data to fit a mapping, denoted as π previously, from the state $\mathcal{S}$ to the action $\mathcal{A}$. Fig. 2 illustrates the network structure of the Dueling DQN used in our work. Initially, the previously defined state vector is fed into the neural network via the input layer. Then follows the input layer, a fully connected layer, which is used to extract features from the state vector. We adopt the Dueling DQN technique to enhance model learning efficiency by integrating the state value and action advantages. Finally, the Q-values, based on the state values and action advantages, are computed through the output layer. We provide a detailed description of the role of each layer as follows.

The input and output layers constitute two pivotal interfaces, with the former receiving raw data and the latter producing processed results. The input layer, as the initial layer of the neural network, is responsible for receiving the input data and forwarding it to subsequent layers. Each neuron in the input layer typically corresponds to a feature in the input data. In the task offloading problem, the input is the previously defined

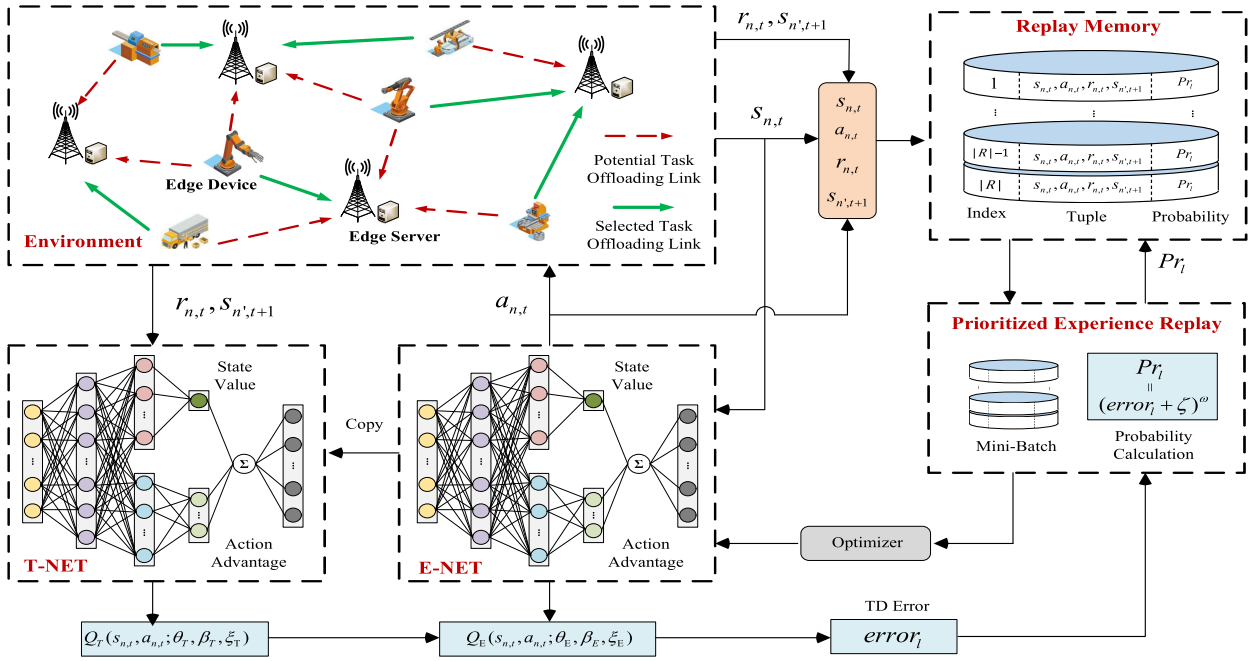

Fig. 3. The framework and procedures of P-D3QN.

state vector $s_{n,t}$, including the information of the ED, MS and the generated task. On the contrary, the output layer, which is the final layer of the neural network, is tasked with generating the final output results. In the task-offloading problem, the output layer is designed to generate Q-values, which are a combination of state values and action advantages.

The hidden layer, as a crucial component of the neural network, functions to process data received from the input layer and extract valuable complex features. The hidden layer contains multiple neurons that process the input data through the activation function and transmit the processed information to the subsequent dueling layers.

The dueling layers are the novel components of the neural network introduced by dueling DQN. In traditional DQN, the Q-values for actions are output separately, whereas dueling DQN produces two distinct outputs: the state value and a vector for action advantage. These outputs are independently learned by the neural network and combined through summation to form the Q-value. By differentiating which part of the Q-value is contributed by the state and which by the action, the calculation of Q-values becomes more robust and efficient. In the context of task offloading problems, such a method of computing Q-values aids the system in concurrently considering the impacts of both state and action on the offloading decisions, thereby facilitating the adoption of more suitable task offloading actions for the system. As shown in Fig. 2, in dueling layers 1 and 2, there are two sub-networks dedicated to learning action advantage and state value, respectively.

In the dueling DQN, denote $A(s,a;\theta,\beta)$ as the action advantage function and $V(s;\theta,\xi)$ as the state value function, where s is the input state, a is the selected action, θ is the common network parameter, β is the action advantage network parameter, and ξ is the state value network parameter. For the generated task $J_{n,t}$, we can get the state $s_{n,t}$. Then, the Q-value $Q(s_{n,t},a_{n,t};\theta,\beta,\xi)$ of the action $a_{n,t}$ can be calculated by

$$Q\left(s_{n,t},a_{n,t};\theta,\beta,\xi\right)=V\left(s_{n,t};\theta,\xi\right)+\left(A\left(s_{n,t},a_{n,t};\theta,\beta\right)-\frac{1}{|\mathcal{A}|}\sum_{a'_{n,t}\in\mathcal{A}}A\left(s_{n,t},a'_{n,t};\theta,\beta\right)\right). \tag{27}$$

After obtaining the Q-values for different actions, we can then select the optimal action corresponding to the state based on our policy.

C. Framework and Procedures of P-D3QN

Based on the neural network structure discussed and the design of state, action and reward of P-D3QN previously, we describe the framework and procedures of P-D3QN in detail, as shown in Fig. 3.

1) Agent Configuration: In the P-D3QN framework, the discussion of state, action and reward about task offloading environment has been illustrated previously. Moreover, the agent configuration is introduced in this sub-section.

In traditional DQN, the use of the max operation can rapidly converge Q-values towards potential optimal targets, but it often leads to overestimation, which refers to the significant bias in the trained model. To address this issue, the agent in P-D3QN comprises two key components: the evaluation network (E-NET) and the target network (T-NET). These components decouple action selection from Q-value computation, effectively

eliminating the problem of overestimation. Both E-NET and T-NET possess identical network structures and share the same network parameters in initialization stage.

E-NET is tasked with real-time learning and offloading decision making. During each training step, it receives the current state $s_{n,t}$ as input and outputs the Q-value $Q_E(s_{n,t}, a_{n,t}; \theta_E, \beta_E, \xi_E)$ for the corresponding actions. Then, E-NET selects the optimal action $a_{n,t}$ according to the maximal Q-value, i.e.,

$$a_{n,t} = \arg\max_{a \in \mathcal{A}} Q_E\left(s_{n,t}, a; \theta_E, \beta_E, \xi_E\right), \tag{28}$$

where $Q_E(s_{n,t}, a; \theta_E, \beta_E, \xi_E)$ represents the Q-value derived from the state-action pair under E-NET parameters θ_E, β_E, ξ_E in P-D3QN. After the action $a_{n,t}$ is selected, this action is transmitted to the environment. Then, the environment returns the next state $a_{n',t+1}$ and the reward $r_{n,t}$ for action $a_{n,t}$. The target Q-value $Q_T(s_{n,t}, a_{n,t}; \theta_T, \beta_T, \xi_T)$ (that is, $y_{n,t}$) can be calculated by T-NET as follows:

$$y_{n,t} = r_{n,t} + \gamma \max_{a \in \mathcal{A}} Q_T\left(s_{n',t+1}, a; \theta_T, \beta_T, \xi_T\right). \tag{29}$$

The evaluation Q-value $Q_E(s_{n,t}, a_{n,t}; \theta_E, \beta_E, \xi_E)$ and the target Q-value $Q_T(s_{n,t}, a_{n,t}; \theta_T, \beta_T, \xi_T)$ are utilized to calculate the loss function to update the weights. Denote $\mathcal{B}$ is the set of mimi-batch size, l is the data transition in $\mathcal{B}$, and $|\mathcal{B}|$ is the total number of data transitions in $\mathcal{B}$. The loss function is defined as follows:

$$L(\theta, \beta, \xi) = \frac{1}{|\mathcal{B}|} \sum_{\forall l \in \mathcal{B}} \left(y_{n,t} - Q_T\left(s_{n,t}, a_{n,t}; \theta_T, \beta_T, \xi_T\right)\right)^2. \tag{30}$$

In P-D3QN, E-NET is continually updated with new training data to adapt to changes in the environment. The update of the E-Net is triggered by the arrival of new tasks. Each new task triggers an update to ensure that the learning model continuously adapts and optimizes its performance based on the most current data. The task generation probability λ, defined in Section III-B, quantifies the rate at which new tasks are generated within the system, directly influencing the frequency of updates.

The role of the T-NET is to provide a stable target Q-value for training the E-NET. Its structure is identical to that of the E-NET, but the updates to its weights are more gradual. The weights of the T-NET are periodically copied from the E-NET. This delayed update mechanism aids in stabilizing the training process, as it reduces the fluctuations in the target Q-values caused by rapid changes in the E-NET. This separation mechanism helps to address the training instability issues caused by the interdependence of targets and estimates.

More specifically, the network parameters of E-NET θ_E, β_E, ξ_E are initialized at random. The initial state of T-NET is a duplicate of E-NET, and the parameters of T-NET are denoted by θ_T, β_T, ξ_T. Furthermore, the mini-batch size capacity $|\mathcal{B}|$ and the replay memory capacity $|\mathcal{R}|$ should be predetermined.

2) Action Selection: Once the E-NET is initialized or trained, the action decision-making process can begin. In particular, when E-NET receives the current state $s_{n,t}$ for the task $J_{n,t}$ of the environment in the timeslot t, the agent will give the action selected by the ε-greedy policy:

$$a_{n,t} = \begin{cases} \text{a random action from } \mathcal{A}, & w.p. \varepsilon_t, \\ \arg\max\limits_{a \in \mathcal{A}} Q_E(s_{n,t}, a; \theta_E, \beta_E, \xi_E), & w.p. 1 - \varepsilon_t. \end{cases} \tag{31}$$

The ε -greedy policy means that a random action $a_{n,t}$ is selected with probability (*w.p.*) ε_t, otherwise, the agent will choose the action that has the greatest $Q_E(s_{n,t}, a; \theta_E, \beta_E, \xi_E)$. To enhance the adaptability of the strategy, we employ an ε-decay mechanism, as in Eq. (32). During the early stages of training, a higher value ε encourages P-D3QN to engage in extensive exploration, thus facilitating better understanding and adaptation to the environment. As training progresses, the value of ε is gradually reduced, leading P-D3QN to increasingly rely on the knowledge it has learned for decision making, thus increasing the frequency of exploiting the current optimal strategy.

$$\varepsilon_t = \max\{\varepsilon_{\min}, \varepsilon_{t-1} - \delta\}. \tag{32}$$

The ε-greedy strategy provides an effective mechanism for balancing exploration and exploitation. By exploring with a certain probability of selecting actions randomly, it helps P-D3QN avoid local optima and potentially discover better strategies. By predominantly exploiting the best current action, it ensures the efficiency of the learning process. In the continuously evolving environment of IIoT support by MEC, the ε-greedy strategy enhances overall learning efficacy by adapting to environmental changes through ongoing exploration. Note that other policies can also be adopted within P-D3QN except ε-greedy policy [42], [43].

3) Replay Memory Caching: In P-D3QN, replay memory caching serves as a database for storing and reusing past experiences, encompassing a series of previous task transitions, including states, actions, rewards, and next states. By sampling from these stored experiences for training, replay memory breaks the temporal correlations between transitions, thereby enhancing the stability and efficiency of the learning process. Furthermore, it allows the algorithm to use old data multiple times, improving the data utilization rate, and aids in mitigating the issue of forgetting during prolonged training periods.

Following the execution of the action selection process, the agent chooses the action $a_{n,t}$. Subsequently, the agent conveys this action $a_{n,t}$ to the environment, which, upon execution of the action $a_{n,t}$, transitions to the next state $s_{n',t+1}$. Meanwhile, the environment gives the corresponding instant reward $r_{n,t}$ for the selected action $a_{n,t}$. Then the transition $(s_{n,t}, a_{n,t}, r_{n,t}, s_{n',t+1})$ is stored in the replay memory. In the event that the number of transitions exceeds the capacity of the replay memory, the oldest transition is displaced by the most recent one.

4) Prioritized Experience Replay: To ensure accurate and consistent decision-making by the agent, it is essential to train or update the parameters of E-NET and T-NET in a timely manner. The accuracy and convergence speed of the model is influenced by data, which highlights the importance of carefully designing the training transition preparation process for the proposed P-D3QN.

Sequentially cached transitions in replay memory have a substantial correlation, which may hinder the training of E-NET.

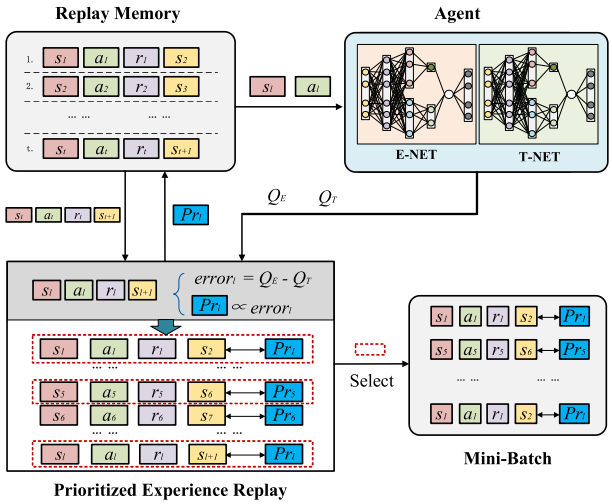

Fig. 4. The prioritized experience replay mechanism of P-D3QN.

The DRL-based task offloading schemes randomly select a batch of transitions from the replay memory to update the model. However, random selection overlooked the relationship between cached data and the model, increasing the model training time. As shown in Fig. 4, we use prioritized experience replay in P-D3QN to accelerate the model convergence speed.

The error of a transition $l = (s_{n,t},\ a_{n,t},\ r_{n,t},\ s_{n',t+1})$ can be defined as the absolute difference between the evaluation Q-value $Q_E(s_{n,t}, a_{n,t}; \theta_E, \beta_E, \xi_E)$ and its corresponding target Q-value $Q_T(s_{n,t}, a_{n,t}; \theta_T, \beta_T, \xi_T)$ (that is, $y_{n,t}$). Mathematically, which is represented as:

$$error_l = \left| y_{n,t} - Q_E\left(s_{n,t}, a_{n,t}; \theta_E, \beta_E, \xi_E\right)\right|. \tag{33}$$

For the T-NET described above, the target Q-value $Q_T(s_{n,t}, a_{n,t}; \theta_E, \beta_E, \xi_E)$ can be calculated by Eq. (29).

We implement prioritized experience replay through proportional prioritization. Initially, the error is translated into priority using the following formula.

$$p_l = \left(error_l + \zeta\right)^{\omega}. \tag{34}$$

Here, ζ is a small positive constant introduced to ensure that no transition is assigned zero priority. ω, which lies in the range $0 \leq \omega \leq 1$, regulates the difference between high and low error. It determines the degree of prioritization used. When ω is equal to zero, it corresponds to a uniform case.

The priority is then converted into a probability that dictates the likelihood of its selection for replay. For a given transition l, the probability of it being selected during the experience replay is determined by the following equation:

$$Pr_l = \frac{p_l}{\sum_k p_k}. \tag{35}$$

In practice, prioritized experience replay chooses a mini-batch of transitions following the aforementioned probability distribution during each learning step and subsequently trains the network on this mini-batch. The only requirement is to store these priorities and sample based on them. Furthermore, if the number of training transitions is inadequate, a high-performance E-NET cannot be realized. Therefore, this selection process should be repeated multiple times to generate a mini-batch of training transitions, denoted as $\mathcal{B} = \{(s_{n,t}, a_{n,t}, r_{n,t}, s_{n',t+1})\}_P$. Next, the selected mini-batch of transitions $\mathcal{B}$ is fed into the agent for E-NET training.

Algorithm 1: P-D3QN.

Input: Observed environment state $\mathcal{S}$
Output: $\pi^* = \arg\ \max_\pi E[\sum_{t\in\mathcal{T}} \gamma^{t-1} r_{n,t}]$, π: $\mathcal{S} \rightarrow \mathcal{A}$
1: Initialize θ_E, β_E, ξ_E, $\theta_T = \theta_E$, $\beta_T = \beta_E$, $\xi_T = \xi_E$;
2: **for** episode $\leftarrow$ 1 to $EPISODES$ **do**
3: Observe the initialize state s_0 from environment;
4: **for** step $\leftarrow$ 0 to STEP **do**
5: E-NET chooses action $a_{n,t}$ by ε-greedy as (31);
6: Observe the next state $s_{n',t+1}$ by action $a_{n,t}$;
7: Obtain reward $r_{n,t}$ from environment;
8: Calculate $Q_E(s_{n,t}, a_{n,t}; \theta_E, \beta_E, \xi_E)$ by E-NET;
9: Calculate $y_{n,t}$ of T-NET by (29);
10: Calculate Q-value error $error_l$ of E-NET and T-NET by (33);
11: Calculate Pr_l of the transition by (35);
12: Store $l = (s_{n,t}, a_{n,t}, r_{n,t}, s_{n',t+1})$ with Pr_l into replay memory;
13: /*Mini-batch $\mathcal{B}$ selection*/
14: **while** Mini-batch $\mathcal{B}$ is not full **do**
15: Select transition l with Prioritized Experience Replay by Pr_l;
16: **end while**
17: /*Take mini-batch $\mathcal{B}$ for training*/
18: **for** $\forall\ l$ in mini-batch $\mathcal{B}$ **do**
19: Accumulate weight change;
20: Update Pr_l for transition l;
21: **end for**
22: Update the parameters of E-NET by Optimizer;
23: **if** step % CopyStepLength $= 0$ **then**
24: Copy parameters $\theta_T = \theta_E$, $\beta_T = \beta_E$, $\xi_T = \xi_E$
25: **end if**
26: **end for**
27: **end for**

5) P-D3QN Procedures: In this sub-section, we introduce the procedures of P-D3QN shown as Algorithm 1 and Fig. 3.

In the initial stage, the E-NET parameters, that is, θ_E, β_E, ξ_E, are randomly initialized and then copied to the T-NET parameters, i.e., $\theta_T = \theta_E$, $\beta_T = \beta_E$, $\xi_T = \xi_E$ (Line 1). An episode is defined as the period from the generation of the first task to the last, culminating in a complete offloading decision scheme for all tasks. Within each episode, the state $s_{n,t}$ is observed in the environment, serving as a starting point for interaction and learning with the environment. The number of steps in model training corresponds to the number of tasks within this period, with each step involving the offloading decision for a task. For a given step, the E-NET employs a ε-greedy policy to select an action $a_{n,t}$ (Line 5). This action $a_{n,t}$ is then inputted into the environment to observe the subsequent state $s_{n',t+1}$ and the associated reward $r_{n,t}$ (Lines 6–7). Concurrently, the Q-value

$Q_E(s_{n,t}, a_{n,t}; \theta_E, \beta_E, \xi_E)$ is calculated in the E-NET based on the action $a_{n,t}$ (Line 8). When the agent receives the next environmental state $s_{n',t+1}$, it enters $s_{n',t+1}$ into the T-NET to obtain the maximum Q-value and calculates $y_{n,t}$ according to Eq. (29) (Line 9). The error between the calculated $y_{n,t}$ and $Q_E(s_{n,t}, a_{n,t}; \theta_E, \beta_E, \xi_E)$ is then calculated and the prioritized probability Pr_l is calculated by (35) (Lines 10–11). This transition $l = (s_{n,t}, a_{n,t}, r_{n,t}, s_{n',t+1})$, along with its corresponding prioritized probability Pr_l, is stored into the replay memory (Line 12).

Subsequently, we are required to select transitions from the replay memory to form a mini-batch for the model updating. When the number of transitions in the mini-batch has not reached its limit, we continuously select the transition l from the replay memory using the prioritized experience replay probability Pr_l (Lines 14–16). Once the mini-batch is constructed, for each transition l within it, we accumulate the weight changes brought by transition l and update its priority Pr_l in the replay memory (Lines 18–21). The E-NET parameters are updated using an optimizer (Line 22). We preset a parameter termed 'CopyStepLength', which is used to define the interval of steps for copying parameters from E-NET to T-NET. Therefore, at every interval of the 'CopyStepLength', we need to copy the parameters of E-NET to T-NET to facilitate the update of T-NET (Lines 23–25).

6) Complexity Analysis: The time complexity of the P-D3QN algorithm is primarily driven by the inner loop, which runs for each step within an episode and involves operations such as action selection, state transition, Q-value calculation, error and priority computation, experience storage, mini-batch sampling, and training. The most computationally intensive operations are mini-batch sampling and training, each with a complexity of $O(|\mathcal{B}| \cdot \log |\mathcal{R}|)$ based on the tree structure searching, where $|\mathcal{B}|$ is the mini-batch size capacity and $|\mathcal{R}|$ is the replay memory capacity. These operations are performed S times per episode with E episodes, thus the overall time complexity of the algorithm is $O(ES|\mathcal{B}| \cdot \log |\mathcal{R}|)$.

V. Experimental Results

In this section, we evaluate the efficiency of P-D3QN with experiments. We begin by describing the simulation setup, followed by presenting the convergence of P-D3QN and comparing P-D3QN with other advanced schemes.

A. Simulation Setup

We performed experiments to evaluate the efficacy of P-D3QN on a machine equipped with four Intel Core i7-7700 CPUs running at 3.60GHz. In this work, the parameter selection is based on previous work [26], [28], and [44]. We deployed 10 MSs and 100 EDs in a $1000\tilde{}m \times 1000\tilde{}m$ area, and the tasks were randomly generated by the EDs in this area with a task generation probability of 0.3. The deployment of MSs follows the principle of maximizing coverage across diverse regions, while the deployment of EDs is conducted based on random generation. Fig. 5 illustrates an example of our deployment of MSs and EDs. The computation resources of MSs are set in the range of 50 to 100GHz. The communication range of each MS was set to $600\tilde{}m$. For the tasks, the data size is generated randomly selected within the range of 1 to 10MB. The task tolerable delay is also randomly generated within a range of 1 s to 3 seconds. The required CPU cycles for task is set to 0.297 gigacycles/MBits. Moreover, the computation resource of each ED is set in the range of 0.5 to 1.0 GHz. The work power of the ED is distributed in the range of 1 Joule/second to 5 Joule/second (Watt). The idle power of the ED is set to 0.5 Joule/second (Watt). For the BRL framework, the replay memory capacity is set to 10,000, and the mini-batch size is 128. The learning rate is set to 0.01. The discount factor γ is set to 0.95. Parameter settings is shown in Table II.

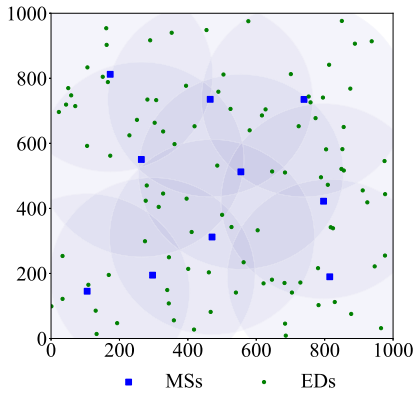

Fig. 5. A deployment example of MSs and EDs.

TABLE II
Experimental Parameter Setup

Parameter	Value
Number of MSs M	10
Number of EDs N	100
Timeslot interval	0.1s
Experimental area	$1000m \times 1000m$
Communication range of MS CR_m	$600m$
Computing capability of MS f_m	[50, 100] GHz
Computing capability of ED f_n	[0.5, 1] GHz
Task data volume $d_{n,t}$	[1, 10] MBits
Task required CPU cycles per bit $c_{n,t}$	0.297 gigacycles/MBits
Task tolerable delay $\tau_{n,t}$	[1, 3] seconds
Channel Bandwidth B	20 MHz
Transmission power P_j	50 mW
Noise power σ^2	–96 dBm
Channel gain per meter β_0	–50 dB

In our experiments, we consider two performance metrics as follows:

- *Average Task Cost:* The objective of P-D3QN is to shorten the total cost for the system. The average task cost of tasks is the ratio of the total cost to the number of completed tasks, which is denoted as ATC, and can be written as

$$ATC = \frac{Total\ Task\ Cost}{Total\ Number\ of\ Completed\ Tasks}. \quad (36)$$

- *Completion Rate:* The ratio of the number of tasks completed within the tolerance delay to the total number of

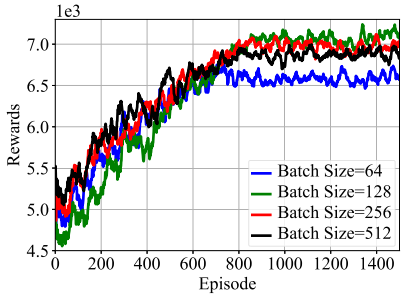

Fig. 6. The rewards with episodes under different batch sizes.

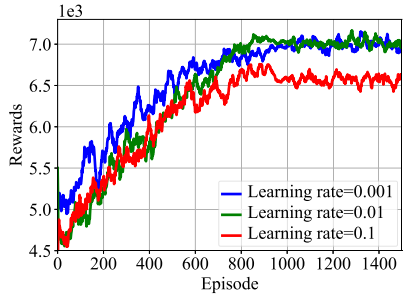

Fig. 7. The rewards with episodes under different learning rate.

tasks, which is denoted as CR and can be written as

$$CR = \frac{Number\ of\ Completed\ Tasks}{Total\ Number\ of\ Tasks}. \tag{37}$$

To comprehensively evaluate the efficiency of the P-D3QN, a detailed comparison is conducted with several benchmarks and relevant schemes. These benchmarks and schemes have been carefully selected to represent a diverse array of approaches within the same domain, allowing for a robust assessment of performance.

A. Benchmarks:

- **RandOff:** Computational tasks of EDs are randomly determined to be either executed locally or offloaded to MSs for processing.
- **NoOff:** The computational tasks of EDs are entirely executed locally, rather than being offloaded to MSs.
- **OptOff:** The computational tasks of EDs are optimally decided to be either executed locally or offloaded to MSs for processing, which can get the best decision.

A. State-of-the-Art schemes::

- **ULOOF:** Estimate the total resource consumption of task execution in terms of time, energy, and bandwidth for both remote and local executions. Then, make an offloading decision based on the comparative assessment of the estimated costs of local and remote executions. [45].
- **DROO:** A scheme maximizes the computation rate in wireless powered MEC networks by automatically optimizing task offloading decisions using DQN. [21].
- **DRLO:** A scheme only considering task delay, and the offloading decision of tasks is achieved by a DRL scheme utilizing Long Short-Term Memory and Double Dueling DQN. [28].

B. Hyperparameters and Convergence

To optimize hyperparameter selection, we conducted convergence experiments with various hyperparameter settings. It is imperative to note that all reward curves have been subjected to robust smoothing techniques; Without such processing, densely packed and highly volatile data points would obscure the discernible trends of the curves.

Fig. 6 shows the trend of cross-episode training rewards for different batch sizes. During the initial stages, the rewards for different batch sizes exhibit significant fluctuations, due to the model's insufficient understanding of the task. As training progresses, the rewards stabilize and gradually increase. The smaller batches (64 and 128) indicate a relatively lower growth of rewards at the initial stage. This is because smaller batches typically contain less training data, introducing more noise and causing the model to study more slowly in the initial stage. After 800-900 episodes, the rewards for all batch sizes reach a steady state. Batch sizes of 128 achieved higher rewards and converged after about 870 episodes. Therefore, we selected 128 as the batch size of P-D3QN for further experiments.

Fig. 7 presents a comparative analysis of the impact of different learning rates on the cumulative rewards achieved by P-D3QN over a sequence of episodes. The general trend across the three curves is congruent. In the initial stage, all three curves exhibit an increasing trend as they learn from the data. As the number of training episodes approaches approximately 800 to 1000, the final rewards values for all three curves tend to stabilize. During the initial learning stage, the rewards for learning rates of 0.1 and 0.01 are lower than those for a learning rate of 0.001. It is attributable to the smaller learning rate extracting more valuable information from the data, thereby enabling better decision-making. However, as the number of training episodes increases, it is observed that the ultimate reward converges to a higher value when the learning rate is set at 0.01. For a learning rate of 0.1, an evident overfitting phenomenon is observed, resulting in a decrease in the final converged reward value after reaching the peak. This decline can be ascribed to the increased learning rate, which could overlook useful information in the data, leading to overfitting of the trained model. Conversely, a much smaller learning rate may prolong the model's learning duration. Thus, we selected a learning rate of 0.01 for P-D3QN

The P-D3QN architecture requires a periodic copy of the network parameters from E-NET to Q-NET, and Fig. 8 delineates the trend of accumulated rewards throughout the training episodes, corresponding to various copy steps. Step sizes of 20, 40, and 60 were evaluated to observe the variation in cumulative rewards with an increasing number of training episodes. With a step size of 20, the initial reward was approximately 5,300. As training progressed, the reward converged to around 6600 by the 900th episode. For a step size of 40, the reward trajectory indicated an ascending trend in the initial 600 episodes, but between 600 and 1000 episodes, a decline in reward magnitude was observed, suggesting potential overfitting within the model. With a step size of 60, a lower initial reward was observed.

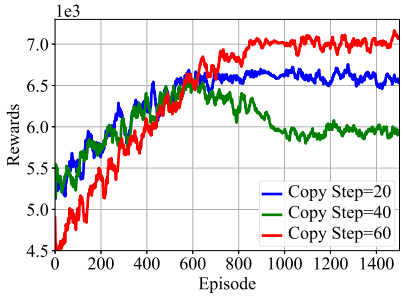

Fig. 8. The rewards with episodes under different copy step.

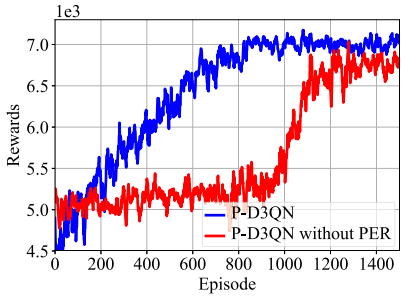

Fig. 9. Comparison of episode rewards for P-D3QN with and without PER.

However, the model exhibited an increase in training epochs, reaching a convergence value surpassing those of step sizes 20 and 40 around the 850th to 900th episode. Consequently, a step size of 60 was chosen as the optimal parameter for our model.

To evaluate the efficiency of the PER mechanism, we present a detailed comparative analysis of the P-D3QN algorithm both with and without the implementation of PER across 1500 episodes, as depicted in Fig. 9. The graph distinctly illustrates that the integration of PER (represented by the blue line) not only leads to consistently higher rewards but also accelerates the convergence process compared to the non-PER variant (red line). On average, the incorporation of PER yields a reward enhancement of approximately 300. Notably, after approximately 800 to 900 episodes, the PER-enhanced P-D3QN converges around a reward value of 7000, whereas the non-PER P-D3QN continues to lag with rewards still below 5500. It is only after 1200 episodes that the P-D3QN without PER finally reaches its convergence value of around 6700. This marked performance disparity emphasizes the crucial role of PER in stabilizing the learning curve and boosting reward convergence. The result substantiates that PER significantly contributes to the optimization of learning efficiencies in P-D3QN scheme.

C. Comparison With Benchmarks

In this sub-section, we compare the performance of P-D3QN with several benchmarks to demonstrate its effectiveness and efficiency.

Fig. 10(a) illustrates the performance of P-D3QN compared to different benchmarks in terms of average task cost across varying numbers of EDs. The results indicate that P-D3QN consistently achieves lower average task costs compared to

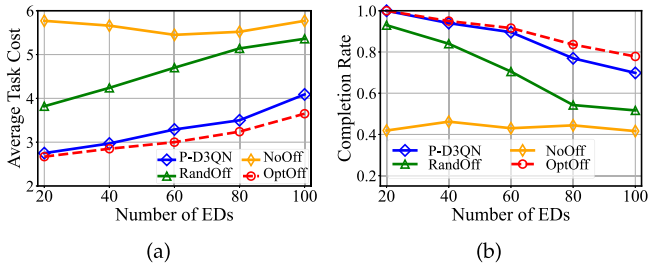

Fig. 10. The completion rate and average task cost for P-D3QN and benchmarks across different numbers of EDs. (a) average task cost; (b) completion rate.

RandOff and NoOff, demonstrating its efficiency and reliability. As the number of EDs increases, all schemes show a trend of rising average task costs. However, the increase for P-D3QN is notably less steep compared to RandOff and NoOff, maintaining a lower average task cost. For example, with 60 EDs, P-D3QN's average task cost is 4.29, which is approximately 24.7% and 33.5% lower than RandOff and NoOff, respectively. On the other hand, OptOff consistently outperforms P-D3QN, with P-D3QN's average task cost being about 7.3% higher than OptOff. When the number of EDs rises to 100, the average task costs for RandOff and NoOff significantly increase, whereas the average task cost of P-D3QN rises more moderately. Specifically, with 100 EDs, P-D3QN shows an improvement of about 20.1% and 24.9% over RandOff and NoOff, respectively, but is 9.5% higher compared to OptOff.

Fig. 10(b) illustrates the performance of four schemes in terms of task completion rate as the number of EDs varies. P-D3QN consistently demonstrates higher completion rates than RandOff and NoOff, underscoring its robustness and efficiency. Generally, all schemes show a decline in completion rate with an increasing number of EDs. However, P-D3QN's decline is significantly less steep, maintaining superior performance. At 60 EDs, P-D3QN achieves a completion rate of 0.89, which is 27.1% higher than RandOff and 108.2% higher than NoOff. OptOff, however, slightly outperforms P-D3QN, with P-D3QN's completion rate being 2.2% lower. When the number of EDs reaches 100, the completion rates of RandOff and NoOff decrease sharply, while P-D3QN experiences a moderate decline. Specifically, at 100 EDs, P-D3QN's completion rate is 35.1% and 67.7% higher than RandOff and NoOff, respectively, but is 11.4% lower compared to OptOff.

D. Comparison With State-of-the-Art Schemes

In this sub-section, we compare the performance of P-D3QN with some state-of-the-art schemes to demonstrate its effectiveness and efficiency.

Fig. 11(a) shows the comparative performance of four different schemes with respect to the completion rate of tasks under varying task generation probability. It is observed that P-D3QN maintains higher completion rates at all levels of probability of task generation, highlighting its robustness and efficiency. As the probability of task generation increases, a common trend

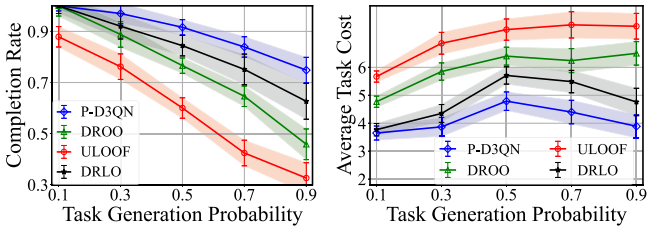

Fig. 11. The completion rate and average task cost with task generation probability under different schemes. (a) completion rate; (b) average task cost.

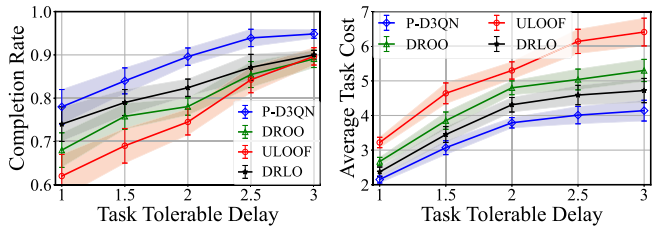

Fig. 12. The completion rate and average task cost with task tolerable delay under different schemes. (a) completion rate; (b) average task cost.

among the schemes is a decrease in the completion rate. However, P-D3QN exhibits a significantly less pronounced decrease, maintaining a higher completion rate relative to other schemes. For instance, at a task generation probability of 0.5, the completion rate of P-D3QN is approximately 0.92, which is higher by 8.5%, 52.6%, and 19.6% than DRLO, ULOOF, and DROO, respectively. As the probability increases to 0.9, the completion rates for ULOOF, DROO, and DRLO experience a substantial decline, while the completion rate of P-D3QN exhibits a much smaller reduction, indicating its capability to manage high efficiency. Quantitatively, compared to DRLO, ULOOF and DROO, P-D3QN shows an improvement of 19.5%, 129.4% and 63.3%, respectively, at the task generation probability of 0.9.

Fig. 11(b) illustrates the average task cost associated with varying task generation probabilities, comparing four different schemes: P-D3QN, ULOOF, DROO, and DRLO. It demonstrates that P-D3QN consistently sustains a lower average task cost across all probabilities, underscoring its cost-effectiveness and operational efficiency. As the probability of task generation increases, all schemes generally experience an increase in average task cost. Nevertheless, P-D3QN shows a comparatively moderate increase, maintaining a cost advantage over competing strategies. Specifically, with a task generation probability of 0.5, P-D3QN achieves an average task cost of approximately 4.79 units, which is 34.8%, 25.2%, and 16.1% lower than that of ULOOF, DROO, and DRLO, respectively. This indicates that P-D3QN not only outperforms in terms of completion rate but also in maintaining cost efficiency under increasing task loads. With the probability rising to 0.9, while the average costs for all the schemes exhibit significant increments, P-D3QN's average cost presents a less substantial elevation. This indicates its superior capability to manage tasks efficiently even in high-task generation scenarios. Quantitatively, with a task generation probability of 0.9, P-D3QN's average task cost is 47.9%, 40.2%, and 18.3% more cost-effective than ULOOF, DROO, and DRLO, respectively. It should be noted that, after the task generation probability exceeds 0.5, there is an observable decrease in the average task cost of DRLO and P-D3QN. This phenomenon can be attributed to the fact that numerous tasks, due to exceeding their time constraints, cannot have their time costs estimated and, consequently, are not included in the cost calculation.

Fig. 12(a) illustrates the comparative performance of four different schemes with respect to task completion rate under various task tolerable delay settings. It is observed that P-D3QN maintains superior completion rates throughout the range of tolerable delays. As the task tolerable delay increases, there is a general trend among the schemes for improvement in the completion rate. However, P-D3QN shows a significantly more pronounced increase, maintaining a higher completion rate relative to other schemes. When the task tolerable delay is set to 1, the performance of P-D3QN is significantly superior to that of other schemes. Compared to DRLO, ULOOF, and DROO, P-D3QN demonstrates an improvement of 5.4%, 25.8%, and 14.7%, respectively. As the task tolerable delay increases to 2, the completion rate of P-D3QN is approximately 0.90, which is higher by 8.7%, 20.3%, and 14.8% compared to DRLO, ULOOF, and DROO, respectively. As the tolerable delay extends to 3, all the schemes have a small difference in completion rate. This is because most tasks can be completed with the task tolerable delay increasing. However, the completion rate of P-D3QN still displays a higher value, indicating its superior capability to accommodate longer delays efficiently.

Fig. 12(b) offers a comparative analysis of the average task cost incurred by four schemes under a range of task tolerable delay settings. As the tolerable delay for tasks is extended, there is a universal increase in the average cost among the schemes, and P-D3QN displays a more gradual ascent in cost, ensuring the advantage over other schemes. DRLO, ULOOF, and DROO demonstrate a relatively steeper increase in average task cost compared to P-D3QN as the task tolerable delay increases. This is attributable to the strategic offloading decisions and optimizations made by P-D3QN, which are tailored according to the varying deadlines, resulting in a reduction in the average task cost. In particular, when the task tolerable delay is established at 2, the average cost of P-D3QN is substantially lower than that of the other schemes, with cost reductions of 50.1%, 28.5%, and 21.0% compared to ULOOF, DROO, and DRLO, respectively. As the tolerable delay increases to 3, P-D3QN's average task cost is 54.2%, 24.5%, and 12.3% more economical than ULOOF, DROO, and DRLO, respectively.

Fig. 13(a) demonstrates the comparative efficacy of four distinct task offloading schemes by measuring the task completion rate with various numbers of EDs. As the number of EDs increases, which typically results in increased computational demand and potential congestion, P-D3QN maintains a higher completion rate and shows a comparatively smaller performance decrease. When the network comprises 20 EDs, the completion

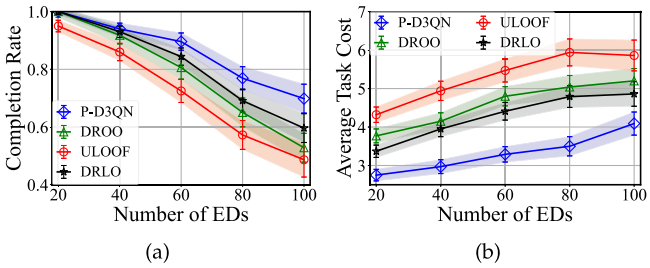

Fig. 13. The completion rate and average task cost with number of EDs under different schemes. (a) completion rate; (b) average task cost.

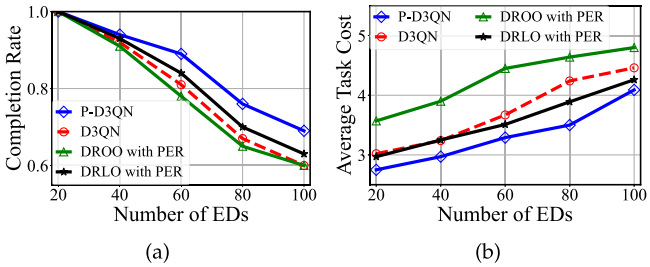

Fig. 14. The completion rate and average task cost with the number of EDs under different schemes with or without PER. (a) completion rate; (b) average task cost.

rates of P-D3QN, DRLO, and ULOOF stand at a near-optimal threshold, substantially exceeding that of the other schemes. For ULOOF, the trend is stable because tasks are processed with a certain level of offloading, and the completion rate gradually decreases as the number of EDs increases. As the ED number reaches 100, the gap in completion rates narrows among DRLO, DROO, and ULOOF. P-D3QN maintains a completion rate of 0.69, which is 17.1%, 43.5%, and 32.2% higher than DRLO, ULOOF, and DROO, respectively. The persistent higher completion rates signify the effectiveness of P-D3QN in accommodating an increasing load of computational tasks efficiently.

Fig. 13(b) provides an evaluation of four schemes with respect to the average task cost in relation to the number of EDs. As the number of EDs increases, it is observed that the average task cost for P-D3QN, ULOOF, DROO, and DRLO shows a persistent upward trajectory. Notably, the P-D3QN scheme demonstrates consistent outperformance in maintaining lower average task costs compared to other schemes. For instance, when the system scales to 80 EDs, the average task cost of P-D3QN is 3.50, which is a decrease of 26.9%, 41.1%, and 30.6% compared to DRLO, ULOOF, and DROO, respectively. Similarly, as the number of EDs increases to 100, the average task cost of P-D3QN rises to 4.09, reflecting decreases of 15.8%, 30.2%, and 21.3% compared to DRLO, ULOOF, and DROO, respectively. This reflects P-D3QN's robust capability to adapt to the system's expansion while minimizing cost implications efficiently.

Fig. 14 shows the impact of PER on completion rate and average task cost for different schemes. The D3QN curve represents the performance of P-D3QN without PER. As shown in Fig. 14(a) and (b), we can see that P-D3QN performs significantly better than D3QN in terms of completion rate and average task cost, indicating that PER contributes significantly to improving the performance of the task offloading model, which is also reflected in Fig. 9. After adding PER to DROO and DRLO, we can see that the resulting schemes still fall short of P-D3QN's performance in completion rate and average task cost. However, compared to DROO and DRLO without PER in Fig. 13, the performance of DROO and DRLO with PER has improved significantly, further demonstrating the performance boost of PER for the task offloading model.

VI. Conclusion

In this paper, we propose an online task offloading scheme based on reinforcement learning for IIoT support by MEC, called P-D3QN. P-D3QN has been designed to address the inherent complexities of IIoT environments, particularly improving on the convergence speed and accuracy deficiencies of existing DRL-based models in task offloading decision-making processes. Integration of double DQN and dueling DQN within the P-D3QN framework enhances the precision of action selection and optimizes the decision-making process by effectively decomposing the state value and the action advantage. Furthermore, incorporation of the PER mechanism accelerates the convergence of model training, selectively focusing on transitions that yield a larger training error, thereby increasing the efficiency of the learning process. The experimental results underscore the efficacy of P-D3QN in reducing task processing cost and maintaining high accuracy, indicating a step forward in the application of DRL in IIoT support by MEC. In the future, we plan to explore the integration of Asynchronous Curriculum Experience Replay (ACER) into our P-D3QN framework to handle increasingly complex tasks in diverse IIoT environments. ACER, with its structured progression through increasingly challenging tasks, offers the potential to significantly enhance the adaptability and efficiency of our task offloading processes by tailoring learning to the specific dynamics of the operational environment.

References

[1] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial Internet of Things: Architecture, advances and challenges," *IEEE Commun. Surv.s Tut.*, vol. 22, no. 4, pp. 2462–2488, Fourth Quarter, 2020.

[2] H. Xu, J. Wu, Q. Pan, X. Guan, and M. Guizani, "A survey on digital twin for industrial Internet of Things: Applications, technologies and tools," *IEEE Commun. Surv. Tut.*, vol. 25, no. 4, pp. 2569–2598, Fourth Quarter, 2023.

[3] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," *IEEE Commun. Surv. Tut.*, vol. 23, no. 4, pp. 2131–2165, Fourth Quarter, 2021.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[5] G. Wang, M. Nixon, and M. Boudreaux, "Toward cloud-assisted industrial IoT platform for large-scale continuous condition monitoring," in *Proc. IEEE*, vol. 107, no. 6, pp. 1193–1205, Jun. 2019.

[6] W. Zhang, D. Yang, Y. Xu, X. Huang, J. Zhang, and M. Gidlund, "DeepHealth: A self-attention based method for instant intelligent predictive maintenance in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5461–5473, Aug. 2021.

[7] Y. Lian et al., "Improved coding landmark-based visual sensor position measurement and planning strategy for multiwarehouse automated guided vehicle," *IEEE Trans. Instrum. Meas.*, vol. 71, 2022, Art. no. 3509316.

[8] B. Kar, W. Yahya, Y.-D. Lin, and A. Ali, "Offloading using traditional optimization and machine learning in federated cloud-edge-fog systems: A survey," *IEEE Commun. Surv. Tut.*, vol. 25, no. 2, pp. 1199–1226, Second Quarter, 2023.
[9] H. Teng, Z. Li, K. Cao, S. Long, S. Guo, and A. Liu, "Game theoretical task offloading for profit maximization in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 9, pp. 5313–5329, Sep. 2023.
[10] J. Ren et al., "An efficient two-layer task offloading scheme for MEC system with multiple services providers," in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 1519–1528.
[11] J. Chi, T. Qiu, F. Xiao, and X. Zhou, "ATOM: Adaptive task offloading with two-stage hybrid matching in MEC-enabled industrial IoT," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4861–4877, May 2024.
[12] J. Chi, C. Xu, T. Qiu, D. Jin, Z. Ning, and M. Daneshmand, "How matching theory enables multi-access edge computing adaptive task scheduling in IIoT," *IEEE Netw.*, vol. 37, no. 3, pp. 126–131, May/Jun. 2023.
[13] K. Sharifani and M. Amini, "Machine learning and deep learning: A review of methods and applications," *World Inf. Technol. Eng. J.*, vol. 10, no. 07, pp. 3897–3904, 2023.
[14] X. Zhao et al., "Deep learning based mobile data offloading in mobile edge computing systems," *Future Gener. Comput. Syst.*, vol. 99, pp. 346–355, 2019.
[15] B. Fan, Z. He, Y. Wu, J. He, Y. Chen, and L. Jiang, "Deep learning empowered traffic offloading in intelligent software defined cellular V2X networks," *IEEE Trans. Veh. Technol*, vol. 69, no. 11, pp. 13 328–13 340, Nov. 2020.
[16] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan./Feb. 2018.
[17] B. Mao, F. Tang, Y. Kawamoto, and N. Kato, "Optimizing computation offloading in satellite-UAV-served 6G IoT: A deep learning approach," *IEEE Netw.*, vol. 35, no. 4, pp. 102–108, Jul./Aug. 2021.
[18] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[19] Y. Li, "Deep reinforcement learning: An overview," 2017, *arXiv: 1701.07274*.
[20] P. Dai, K. Hu, X. Wu, H. Xing, and Z. Yu, "Asynchronous deep reinforcement learning for data-driven task offloading in MEC-empowered vehicular networks," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.
[21] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.
[22] S. Bi, L. Huang, H. Wang, and Y.-J. A. Zhang, "Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7519–7537, Nov. 2021.
[23] K. Peng, P. Xiao, S. Wang, and V. C. M. Leung, "AoI-aware partial computation offloading in IIoT with edge computing: A deep reinforcement learning based approach," *IEEE Trans. Cloud Comput.*, vol. 11, no. 4, pp. 3766–3777, Fourth Quarter, 2023.
[24] W. Fan, S. Li, J. Liu, Y. Su, F. Wu, and Y. Liu, "Joint task offloading and resource allocation for accuracy-aware machine-learning-based IIoT applications," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3305–3321, Feb. 2023.
[25] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 242–253, Jan. 2021.
[26] Y. Ren, Y. Sun, and M. Peng, "Deep reinforcement learning based computation offloading in fog enabled industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4978–4987, Jul. 2021.
[27] F. Zhang, G. Han, L. Liu, M. Martínez-García, and Y. Peng, "Deep reinforcement learning based cooperative partial task offloading and resource allocation for IIoT applications," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 5, pp. 2991–3006, Sep./Oct. 2023.
[28] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1985–1997, Jun. 2022.
[29] R. Zhou et al., "Online task offloading for 5G small cell networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 2103–2115, Jun. 2022.
[30] K. Zheng, G. Jiang, X. Liu, K. Chi, X. Yao, and J. Liu, "DRL-based offloading for computation delay minimization in wireless-powered multi-access edge computing," *IEEE Trans. Commun.*, vol. 71, no. 3, pp. 1755–1770, Mar. 2023.
[31] H. Zhang, H. Wang, Y. Li, K. Long, and A. Nallanathan, "DRL-driven dynamic resource allocation for task-oriented semantic communication," *IEEE Trans. Commun.*, vol. 71, no. 7, pp. 3992–4004, Jul. 2023.
[32] Z. Gao, L. Yang, and Y. Dai, "Large-scale computation offloading using a multi-agent reinforcement learning in heterogeneous multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 6, pp. 3425–3443, Jun. 2023.
[33] 5G Americas, "Communications for automation in vertical domains," Nov. 2018. [Online]. Available: https://www.5gamericas.org/wp-content/uploads/2019/07/5G_Americas_White_Paper_Communications_for_Automation_in_Vertical_Domains_November_2018.pdf
[34] 5GACIA, "5G for Industrial Internet of Things (IIoT): Capabilities, features, and potential," Nov. 2021. [Online]. Available: https://5g-acia.org/whitepapers/5g-for-industrial-internet-of-things/
[35] B. Yang, X. Cao, J. Bassey, X. Li, and L. Qian, "Computation offloading in multi-access edge computing: A multi-task learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 9, pp. 2745–2762, Sep. 2021.
[36] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1985–1997, Jun. 2022.
[37] Y. Jing, X. Yu, and S. ShahbazPanahi, "Interleaved training scheme for multi-user massive MIMO downlink with user SINR constraint," *IEEE Trans. Commun.*, vol. 71, no. 10, pp. 6115–6129, Oct. 2023.
[38] A. J. Jerri, "The shannon sampling theorem—its various extensions and applications: A tutorial review," in *Proc. IEEE*, vol. 65, no. 11, pp. 1565–1596, Nov. 1977.
[39] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.
[40] Z. Liang, Y. Liu, T.-M. Lok, and K. Huang, "Multiuser computation offloading and downloading for edge computing with virtualization," *IEEE Trans. Wireless Commun.*, vol. 18, no. 9, pp. 4298–4311, Sep. 2019.
[41] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 11, pp. 13344–13362, Nov. 2023.
[42] J. Ji, X. Ren, L. Cai, and K. Zhu, "Downlink scheduler for delay guaranteed services using deep reinforcement learning," *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 3376–3390, Apr. 2024.
[43] Y. Fan, J. Ge, S. Zhang, J. Wu, and B. Luo, "Decentralized scheduling for concurrent tasks in mobile edge computing via deep reinforcement learning," *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 2765–2779, Apr. 2024.
[44] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial IoT–edge–cloud computing environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 12, pp. 2759–2774, Dec. 2019.
[45] J. L. D. Neto, S.-Y. Yu, D. F. Macedo, J. M. S. Nogueira, R. Langar, and S. Secci, "ULOOF: A user level online offloading framework for mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 17, no. 11, pp. 2660–2674, Nov. 2018.

Jiancheng Chi (Member, IEEE) received the BEng degree from the Dalian University of Technology, Dalian, China, in 2017, and the MEng degree from Tianjin University, Tianjin, China, in 2020. He is currently working toward the PhD degree in computer science and technology with Tianjin University, Tianjin, China. He is a visiting scholar with Japan Advanced Institute of Science and Technology, Japan in 2023. His research interests include the Industrial Internet of Things, industrial big data, edge computing, and machine learning. He has published some papers on top journals, such as *IEEE Communications Surveys and Tutorials*, *IEEE Transactions on Mobile Computing*, *Journal of Network and Computer Applications*, and *IEEE Network*. He has won many scholarships at Tianjin University, including the Chinese Government Scholarship, the First-Class Academic Scholarship, the Lenovo Scholarship and the Outstanding Youth Scholarship. He is a student member of ACM.

Xiaobo Zhou (Senior Member, IEEE) received the BSc degree in electronic information science and technology from the University of Science and Technology of China, Hefei, China, in 2007, the ME degree in computer application technology from the Graduate University of Chinese Academy of Science, Beijing, China, in 2010, and the PhD degree in information science from the School of Information Science, Japan Advanced Institute of Science and Technology, Nomi, Ishikawa, Japan, in 2013. From April 2014 to March 2015, he was a researcher with the Department of Communications Engineering, University of Oulu, Oulu, Finland. He is currently a professor with College of Intelligence and Computing, Tianjin University, Tianjin, China. His research interests include Internet of Things, cloud computing, edge computing, data center networks, and vehicular networks.

Fu Xiao (Senior Member, IEEE) received the PhD degree in computer science and technology from the Nanjing University of Science and Technology, Nanjing, China, in 2007. He is currently a professor and the PhD supervisor with the School of Computer, Nanjing University of Posts and Telecommunications. He has authored papers in research related international conferences, including INFOCOM, Mobihoc, and ICC, the *IEEE Journal on Selected Areas in Communications*, *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Mobile Computing*, *ACM Transactions on Embedded Computing Systems*, and *IEEE Transactions on Vehicular Technology*. His research interest includes the Internet of Things.

Yuto Lim (Member, IEEE) received the BEng (Hons.) and MInfTech degrees from Universiti Malaysia Sarawak (UNIMAS), Malaysia, in 1998 and 2000, respectively, and the PhD degree in communications and computer engineering from Kyoto University, in 2005. From November 2005 to September 2009, he was an expert researcher with the National Institute of Information and Communications Technology (NICT), Japan. In 2005, he was actively joining the standardization activities of IEEE 802.11 s Mesh Networking. He and his team members have introduced two proposals, which currently adopted in the draft of IEEE 802.11 s D1.04. He also led the RA-OLSR group in resolving a part of the comments. In 2006, he is also actively joining the standardization activities of next-generation home networks at Telecommunication Technology Committee (TTC), Japan. Since October 2009, he has been working with the Japan Advanced Institute of Science and Technology (JAIST), as an associate professor. From 2017 to 2019, he was actively joining the standardization activities of smart cities from the Bridging the Standardization Gap (BSG) Working Group of the TTC, Japan. He is honorably appointed as a TTC representative to report the "Smart City" activities in Asia-Pacific Region in both APT Standardization Program Forum (ASTAP) and APT Telecommunication/ICT Development Forum (ADF). His research interests include smart homes, smart cities, cyber-physical systems, the Internet of Things, future wireless communication and networks, quantum networks, and smart energy distribution. He is a member of IEICE and IPSJ.

Tie Qiu (Senior Member, IEEE) is currently a full professor with the School of Computer Science and Technology, Tianjin University, China. Prior to this position, he held assistant professor in 2008 and associate professor in 2013 with the School of Software, Dalian University of Technology. He was a visiting professor with the department of electrical and computer engineering of Iowa State University in U.S. (2014–2015). He serves as an associate editor of *IEEE/ACM Transactions on Networking* (ToN), *IEEE Transactions on Network Science and Engineering* (TNSE) and *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, area editor of *Ad Hoc Networks* (Elsevier), associate editor of *Computers and Electrical Engineering* (Elsevier), *Human-centric Computing and Information Sciences* (Springer), a guest editor of *Future Generation Computer Systems*. He serves as general chair, program chair, workshop chair, publicity chair, publication chair or TPC member of a number of international conferences. He has authored/co-authored 10 books, more than 200 scientific papers in international journals and conference proceedings, such as *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Industrial Informatics*, *IEEE Communications Surveys & Tutorials*, IEEE Communications, INFOCOM, GLOBECOM etc. There are 18 papers listed as ESI highly cited papers. He has contributed to the development of 6 copyrighted software systems and invented 20 patents. He is a distinguished member of China Computer Federation (CCF) and a senior member of ACM.