IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 18 July 2023, accepted 2 August 2023, date of publication 7 August 2023, date of current version 10 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3302518

RESEARCH ARTICLE

Deep Reinforcement Learning-Based Task Offloading and Resource Allocation for Industrial IoT in MEC Federation System

HUONG MAI DO[1], TUAN PHONG TRAN[1], AND MYUNGSIK YOO[2]
[1]Department of Information Communication Convergence Technology, Soongsil University, Seoul 06978, South Korea
[2]School of Electronic Engineering, Soongsil University, Seoul 06978, South Korea

Corresponding author: Myungsik Yoo (myoo@ssu.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) funded by the Korean Government [Ministry of Science and ICT (MSIT)], South Korea, through the Development of Candidate Element Technology for Intelligent 6G Mobile Core Network under Grant 2022-0-01015; and in part by MSIT under the Information Technology Research Center (ITRC) Support Program Supervised by IITP under Grant IITP-2023-2021-0-02046.

ABSTRACT The rapid growth of the Internet of Things (IoT) has resulted in the development of intelligent industrial systems known as Industrial IoT (IIoT). These systems integrate smart devices, sensors, cameras, and 5G technologies to enable automated data gathering and analysis boost production efficiency and overcome scalability issues. However, IoT devices have limited computer power, memory, and battery capacities. To address these challenges, mobile edge computing (MEC) has been introduced to IIoT systems to reduce the computational burden on the devices. While the dedicated MEC paradigm limits optimal resource utilization and load balancing, the MEC federation can potentially overcome these drawbacks. However, previous studies have relied on idealized assumptions when developing optimal models, raising concerns about their practical applicability. In this study, we investigated the joint decision offloading and resource allocation problem for MEC federation in the IIoT. Specifically, an optimization model was constructed based on all real-world factors influencing system performance. To minimize the total energy delay cost, the original problem was transformed into a Markov decision process. Considering task generation dynamics and continuity, we addressed the Markov decision process using a deep reinforcement learning method. We propose a deep deterministic policy gradient algorithm with prioritized experience replay (DDPG-PER)-based resource allocation that can handle high-dimensional continuity of action and state spaces. The simulation results indicate that the proposed approach effectively minimizes the energy-delay costs associated with tasks.

INDEX TERMS MEC federation, IIoT, task offloading, resource allocation, Markov decision process, deep reinforcement learning.

I. INTRODUCTION

The Internet of Things (IoT) has increased significantly in recent years owing to the rapid growth in smart mobile devices and 5G technologies. The IoT has been used in various domains, such as medicine [1], transportation [2], and the military [3]. In the industrial sector, IoT-enabled industrial systems, also known as Industrial IoT (IIoT), help boost productivity, reduce system deployment costs, and simplify maintenance. According to a recent study by Microsoft, 91% of enterprises have at least one IIoT project in their pipelines, and Million Insights predicts that the IIoT industry will reach 992 billion in worldwide investments by 2025 [4]. By automating smart devices to gather, process, and transfer real-time data inside industrial systems, the IIoT may lead to massive breakthroughs in the industry. However, IIoT systems generate a considerable volume of data, while the devices only limited computing, memory, and

The associate editor coordinating the review of this manuscript and approving it for publication was Chih-Min Yu.

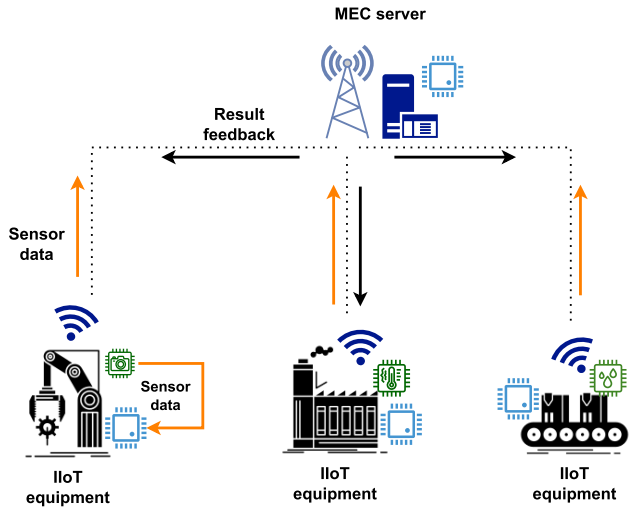


FIGURE 1. The MEC-based IIoT system.

energy capacities. Furthermore, several applications are sensitive to delays, while others require complex processes. Therefore, it is crucial to develop technologies that can improve the performance of IIoT systems.

Numerous studies have used cloud computing to overcome IIoT system issues [5], [6]. Cloud computing provides on-demand computer resources, including services, storage, and processing capabilities. However, cloud servers are often deployed in remote locations by using IoT devices. This increases system delay, which is particularly undesirable for latency-sensitive applications. Mobile edge computing (MEC) has emerged as a viable solution to this problem. Owing to the proximity of the MEC server deployment, computationally demanding tasks are executed with minimal latency. Fig. 1 shows an IIoT system supported by an MEC server. The integration of sensors and processors in industrial equipment facilitates data acquisition and analysis. The IIoT devices have the capability to manage processing tasks independently or alternatively, the tasks can be offloaded to the MEC server. In particular, several studies [7], [8] suggested offloading tasks to MEC servers that are directly connected to the device, the so-called dedicated MEC servers, to minimize system latency. If the dedicated MEC server lacks the necessary capabilities, tasks are redirected to a remote cloud server. This not only introduces higher processing latency but also disregards the utilization of resources available in the nearby MEC servers.

The MEC federation was proposed to address this issue [9], [10], [11], [12]. Instead of exclusively connecting to a distant cloud server, MEC servers inside the MEC federation system have direct connections. This has significant advantages for utilizing system resources and allocating workloads to suitable locations within the system according to the amount of resources required. Recent research [11], [12] focused on reducing the latency of IIoT systems using the MEC federation. Through the optimization of resource allocation and task offloading, the system can minimize service latency. Nevertheless, the formulation of the optimization problem is constrained by certain limitations. Existing studies [13] assumed that IIoT devices lack the ability to execute tasks independently. In addition, several studies [11], [12] neglected to consider the computing capabilities of IIoT devices in their optimal models. In practice, IIoT devices can perform several computational tasks [14], [15]. Moreover, relying solely on the MEC server to perform the tasks, instead of delegating it to the devices themselves, may not be the optimal approach, particularly in systems involving a substantial number of devices. This procedure may impose excessive strain on the bandwidth and processing resources of the MEC servers. However, the optimization models in various studies [10], [16], [17] neglected the influence of queues and storage resources. Several studies [11], [12] utilized $M/M/1$ queues to model the task processing. Technically, this entails assuming that the cache of the MEC server possesses infinite capacity for task queuing and that each MEC server is capable of handling a maximum of one task-processing service. The assumption of unlimited storage capacity of MEC servers poses a challenge in the practical implementation of optimal decision-making, particularly in systems with substantial workload volumes, such as IIoT.

In addition to the latency, the energy consumption of the system is a crucial consideration. This considerably influences the life cycle of IIoT devices, owing to limitations in their battery capacity. To address this problem, several studies [10], [18], [19] have presented strategies for concurrently optimizing the latency of whole task processing and power consumption of IIoT devices. Nevertheless, certain constraints exist in modeling the energy consumption of MEC systems. This study [10] omitted the power consumption of servers and focused solely on enhancing the energy efficiency of IIoT devices during task execution and data transmission. References [18] and [19] conducted research on the energy optimization of servers with a specific focus on computational energy consumption. However, their study neglected the fact that servers consume energy even when they are not actively processing tasks because they must remain in an operational state and be available to transmit and receive data. Insufficient optimization of server power consumption can lead to economic losses due to the charges incurred for power consumption by MEC servers. Consequently, it is crucial to minimize the energy consumption of both IE and MEC servers during task execution.

However, the resource allocation problem in an MEC federation system as a nonlinear integer program has been proven to be an NP problem [11]. Hence, it is not feasible to obtain a globally optimal solution in polynomial time. Several solutions have been proposed to address this problem, including traditional optimization-based [8], [20], [21], [22] and machine-learning (ML)-based methods [18], [19], [23], [24]. The authors of [20] presented a particle swarm optimization to address the resource allocation issue for an IIoT system supported by MEC. The low complexity of heuristic techniques results in a tradeoff in

model performance, particularly for systems with high-dimensional inputs. Hence, ML-based techniques are considered effective solutions to this issue. In recent years, deep reinforcement learning (DRL) models have been increasingly used to optimize MEC system resources. A deep Q-learning strategy was proposed to decrease the latency and energy consumption in MEC-assisted IIoT systems with blockchain integration [19]. The authors of [23] used a deep deterministic policy gradient (DDPG) to construct a framework for the dynamic resource management of an MEC system in an IIoT network. Although Deep Q-Learning is restricted to discrete optimization problems, DDPG can be used to determine optimum decisions in continuous domains, particularly for resource allocation issues. Using the replay buffer, DDPG may retrieve prior data samples, thereby reducing the input correlation. The procedure for sampling data in this buffer has a direct impact on model performance and convergence. Nevertheless, DDPG selects data samples from a buffer with equal weights, disregarding the fact that certain experiences provide higher returns. Thus, the prioritized experience replay (PER) [25] was utilized to enhance the performance of the model by evaluating its significance and selecting appropriate experiences for the DDPG training process. In addition, the importance sampling (IS) approach [26] was used to assist the PER in managing the problem of overfitting.

To address the above mentioned issues, in this study, we investigated the performance and resource optimization problems of the MEC federation system utilized in an IIoT network. In particular, we simultaneously consider latency and energy consumption throughout the entire IIoT system. The available resources of both industrial equipment (IE) and MEC servers, such as computation, communication, storage, and power resources, are also used as inputs for the optimization problems. To address this issue, we propose a DDPG-based model for effective resource allocation and task offloading that is enhanced using a PER buffer. The main contributions of this study are as follows:

- We propose an efficient task offloading and resource allocation framework for IIoT systems that are integrated with MEC federation. By modeling the system in detail with end to end delay and energy consumption of both IIoT devices and MEC servers, we formulate an optimization problem with trade-off between delay and energy consumption. In contrast to prior research, our objective function takes comprehensively into account factors that influence the performance of the system in practical settings. We also use the queuing model as $M/M/c/K$ model to formulate queuing delay.
- To deal with the mixed-integer nonlinear programming (MINLP) characteristics of our optimization problem, the proposed formulation is transformed into a Markov decision process (MDP). We propose the DDPG-PER algorithm-based resource allocation for solving this optimization problem. The algorithm is run as a DRL unit in the MEC Federation controller, collecting and processing all information and having an overview of the whole system.
- We conduct intensive experiments to highlight the performance of the proposed MEC federation system and DDPG-PER-based resource allocation method. The results demonstrate that our proposal outperforms other resource allocation strategies and system schemes for the IIoT system.

The remainder of this paper is organized as follows. Section II presents work related to optimization of MEC systems that has been studied in recent years. Section III describes the proposed system model and defines the optimization problem for resource allocation. In Section IV, a DDPG-PER-based resource allocation algorithm is described to solve this problem. Section V presents simulation results. Finally, conclusions are presented in Section VI.

II. RELATED WORK

The MEC optimization of IIoT systems has gained increasing attention in recent years. Most research [17], [20], [23] has focused on optimizing the performance of a dedicated MEC system in which IE tasks are offloaded to a local MEC server or cloud server. Additionally, several studies [10], [11], [12] have been conducted on the integration and performance optimization of MEC federation systems on IIoT networks. Table 1 summarizes the recent studies related to optimization by task offloading and resource allocation in MEC-enabled IIoT systems. Then, based on Table 1, we summarize and highlight our contributions to the existing works, as shown in Table 2.

A. DEDICATED MEC SYSTEM

The authors of [8] proposed a consolidated stochastic computational offloading (CSCO) framework for MEC-based IIoT systems to reduce service disruptions, particularly for time-critical tasks. In this system, IIoT tasks can be offloaded to the connected MEC server, which is modeled as a Poisson process for an M/M/c/c system. In addition, a method for predicting the possibility of overloading an MEC server was developed. Based on the predicted results, the authors built a QoS-aware offloading framework to minimize the system latency.

The resource allocation problem for an IIoT network in a forest-monitoring system was addressed in [20]. This system comprises IIoT devices deployed across a forest to gather and transmit data to unmanned aerial vehicles (UAVs) that function as MEC servers. Reference [20] proposed a learning-based cooperative particle swarm optimization algorithm combined with a Markov random field-based decomposition approach to reduce the response time of UAVs.

Reference [23] were interested in dedicated MEC systems. The foundation of the system primarily consists of a BS and an MEC server. Each IIoT piece of equipment creates a computing task that must be executed using an MEC server.

TABLE 1. Related works.

Papers	Year	System model					Main objective(s)							Method
			MEC Federation				Energy			Delay				
		Dedicated MEC	MF1	MF2	MF3	IEC	Computing task	Transmitting task & results	Receiving task & result	Computing task	Transmitting task	Transmitting result	Queuing	DRL -based
[18]	2020	✓				✓	✓			✓	✓			✓
[20]	2021	✓				✓				✓				
[19]	2021	✓				✓	✓							✓
[23]	2021	✓				✓				✓	✓			✓
[8]	2022	✓				✓				✓				
[21]	2022	✓				✓		✓						
[16]	2023	✓				✓				✓	✓	✓		✓
[17]	2023	✓				✓				✓	✓			
[10]	2022		✓			✓	✓			✓	✓	✓	✓	
[11]	2022		✓							✓	✓	✓	✓	
[12]	2023			✓						✓	✓		✓	
[13]	2022				✓		✓	✓		✓	✓			✓
Our work	2023			✓		✓	✓	✓	✓	✓	✓	✓	✓	✓

Note:
MF1: Task migrating decision based on Greedy policy
MF2: Task migrating decision based on Optimal policy
MF3: No task migrating between servers
IEC: Consider IIoT equipment computing resource

TABLE 2. Our contributions.

Papers	F1	F2	F3	F4	F5
[18]				✓	
[20]					
[19]				✓	
[23]				✓	
[8]					
[21]					
[16]				✓	
[17]					
[10]			✓		
[11]			✓		
[12]					
[13]				✓	
Our work: MEC Federation for IIoT	✓	✓	✓	✓	✓

F1: MEC federation (Using both Computing resources of IIoT equipment & MEC servers) with task offloading decision based on Optimal policy
F2: Energy model formulation includes Computation, Transmission & Reception
F3: Delay model formulation includes Computation, Task & Result Transmission, Queuing
F4: Using DRL to solve the problem
F5: Formulate queuing model as M/M/c/K

When attempting to manage limited resources, the key problem is reducing delays. The primary contribution of [23] was an examination of dynamic resource management for integrated power control and computational resources. To solve this problem, a DRL-based dynamic resource management (DDRM) algorithm was proposed.

In [16], the authors considered MEC-enabled IIoT networks comprising of an MEC server and multiple user devices. They suggested three strategies for minimizing the system delay: local computing, partial computation offloading, and complete computation offloading. To obtain the optimal decision, the optimization problem was modeled as MDP and then solved by the DDPG algorithm.

The authors of [21] investigated maximizing the long-term utility of IIoT systems. In particular, they created an optimization mathematical model that enhanced experience quality while minimizing energy expenses. The optimization problem was decomposed into subproblems with varying time frames and addressed using a low-complexity heuristic technique based on the alternating direction multiplier method.

Reference [18] considered the optimization in MEC system for IoT networks. The gateway collects data from the IoT users that are processed by the edge server. The proposed computation task offloading scheme includes two steps:1) centralized user clustering using the K-means clustering algorithm, and 2) distributed computation offloading using a Deep Q-Network. Their goal was to reduce long-term system costs by limiting computing, storage, and transmission power resources. Reference [18] performed a relatively good optimization based on the DQL algorithm; however, as mentioned above, one of the two optimal objects is the energy consumption, which has not been fully formulated.

The authors of [19] proposed a framework that enables the integration of the blockchain into MEC-enabled IIoT systems. This 3-layer system consisted of a user layer for the IIoT devices, a controller layer consisting of several MEC servers, and a blockchain layer that provides security and reliability to the system. Simultaneous optimization of the computational overhead and power consumption of the system was modeled as an MDP. The authors propose an algorithm based on Deep Q-learning to solve this problem. Similar to [18] and [19] did not consider the power consumption of servers during data transmission.

In [17], an optimal method for task offloading and resource allocation in an MEC-enabled IIoT system was proposed. Specifically, an IIoT system consists of many devices connected to an MEC server via a 5G wireless network, whereas the MEC server is connected to a cloud server via a fiber-optic network. In contrast to other studies, [17] built an optimization problem that considered system delay and economic loss. Subsequently, an optimal algorithm based on the general Benders decomposition (GBD) technology and a heuristic algorithm were proposed to efficiently solve the problem.

B. MEC FEDERATION SYSTEM

In contrast to a dedicated MEC system, an MEC federation must determine the migration of tasks between the MEC servers within the system. However, [13] failed to consider the transfer of tasks between MEC servers, and [10], [11] used a greedy scheme that only migrates tasks when the MEC server resources are exhausted. However, this may result in inefficient utilization of idle resources. Using the optimal scheme, [12] made task migration decisions based on an optimization process that used the system state as the input.

Reference [11] proposed an MEC-enabled IIoT network architecture integrated with a Software-Defined Network (SDN). In this architecture, the SDN controller was responsible for choosing the MEC server to execute tasks. A greedy approach was proposed to decrease the response time of tasks and balance the load amongst MEC servers.

In [12], a multiarmed bandit-based learning strategy was introduced to minimize the task execution latency in an MEC-enabled IIoT system. In particular, the tasks generated by the user device may be handled by the service device or MEC server, depending on the task co-offloading framework. This strategy was straightforward to implement because it does not require the complete state of the entire system.

In [10], the authors developed a resource allocation framework for an MEC-enabled IIoT network that enables IIoT tasks to be handled locally or offloaded to MEC servers. The system prioritizes the local MEC server for task execution. The tasks were forwarded only to a remote MEC server when the local server is overwhelmed. The authors concurrently investigated the latency and power consumption of the system while formulating the problem and proposed an improved differential evolution algorithm (IDE) and a heuristic algorithm for optimal decision making. However, they neglected to calculate the power consumption of the MEC servers. Moreover,

their resource allocation and task offloading strategies were not optimal when only forwarding tasks were used and the local MEC server was overloaded.

Reference [13] aimed to develop a task offloading framework that considers privacy. MEC servers manage tasks to address the computational challenges of IoT devices. They developed models with the optimal energy costs, response times, and success rates. The authors subsequently proposed a local differential privacy-based deep reinforcement learning (LDP-DRL) strategy for optimal decision-making. In this model, the processing capacity and energy consumption of the IIoT devices as well as the data transmission and reception capabilities of the MEC servers were neglected. In addition, the proposed model is only concerned with determining the best MEC server as opposed to coordinating between servers to make an optimal decision.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. SYSTEM OVERVIEW

Fig. 2 illustrates an MEC-enabled IIoT network composed of multiple IEs, MEC servers, and an MEC federation controller. In this system, IEs are located in geographically fixed industrial zones. Hence, this study does not consider the mobility of IEs. As shown in Fig. 2, multiple IEs can be simultaneously connected to the same MEC server via a wireless connection. These MEC servers are deployed close to IEs. In contrast to the dedicated MEC paradigm, the MEC servers in an MEC federation are directly linked through a backhaul network. This connectivity allows users to interact with and share resources and exchange information. The MEC servers are connected to a centralized MEC federation controller, which collects all system data and determines the optimum resource allocation.

To guarantee services such as data collection, processing, and feedback provision in an IIoT system, IEs must execute a large number of computation-intensive tasks. Owing to constraints in computing power and battery capacity, they may not meet the computational demands. The integration of MEC with the IIoT network enables the offloading of IE workloads to MEC servers. The connected MEC server, also known as a local MEC server, provides better computational and storage capacities for executing tasks that are offloaded by multiple IEs. However, the local MEC server is not the optimal choice if the resource demand exceeds this limit. When a task is too computationally intensive or when excessive IEs are inquiring simultaneously, the local MEC server may shift the task to a remote MEC server with additional resources. The MEC federation controller determines where the computations are performed, depending on the available system resources.

The flowchart of the entire process is illustrated in Fig. 3:

- **Information processing**: ①, The task is generated by IE; ②, the information of resources and task size are transferred to the local MEC server; ③, the information of resources and task size of IE and MEC servers are sent to Controller; ④, the Controller makes the decision based on this information; ⑤ and ⑥, the decision is sent back to the MEC servers and IE.
- **Task processing**: After obtaining the decision in ⑦, the task will be executed by IE; or ⑧ the task will be executed by the local MEC server; or ⑨, the task will be migrated to the remote MEC server; in the case of choosing the remote MEC server ⑩, the result of migrated task will be sent back to the local MEC server; ⑪, finally, the result from the local MEC server will be sent back to the IE.

B. SYSTEM MODEL

In this study, we denote the number of IE and MEC servers as D and S, respectively. We consider a time interval divided into T time slots. In each time slot t, IE d generates $I^d(t)$ tasks by following a uniform distribution [7]. As in [11], we denote the ith task generated by the IE d in time slot t as a tuple

$$w_i^d(t) = \{a_i^d(t), b_i^d(t), c_i^d(t)\}$$

where $a_i^d(t)$, $b_i^d(t)$, and $c_i^d(t)$ denote the data size, result size, and number of CPU cycles required to process a unit of data, respectively. $c_i^d(t)$ depends on the type of application, and can be acquired through offline measurements [7].

As mentioned previously, IEs generate multiple large real-time tasks that can be handled locally on the IE or remotely on the MEC server through computational offloading. The offloading strategy for each task is indicated by variable $x_i^d(t)$. Technically, $x_i^d(t) = 0$ represents the task to be executed locally at the IE, whereas $x_i^d = s(0 < s \leq S)$ indicates that the task is handled by MEC server s.

C. DELAY MODEL

1) COMPUTATION DELAY

Whether a task is executed locally or offloaded to an MEC server, computation always results in a delay. If the IE has sufficient resources, the data transmission cost is reduced. $T_i^d(t)$ represents the latency in executing the ith task of the IE d, which is calculated as:

$$T_i^d(t) = \frac{a_i^d(t)c_i^d(t)}{f_i^d(t)}, \tag{1}$$

where $f_i^d(t)$ denotes the computing resources allocated by IE d to execute task $w_i^d(t)$ in time slot t. Similarly, the delay caused by the computing task $w_i^d(t)$ on MEC server s is determined as follows:

$$T_i^s(t) = \frac{a_i^d(t)c_i^d(t)}{f_i^s(t)}, \tag{2}$$

where $f_i^s(t)$ denotes the computing resources allocated by MEC server s to execute task $w_i^d(t)$ in time slot t.

2) TRANSMISSION DELAY

In an IIoT system, IEs must perform several computationally complex tasks. Because of resource constraints, they may

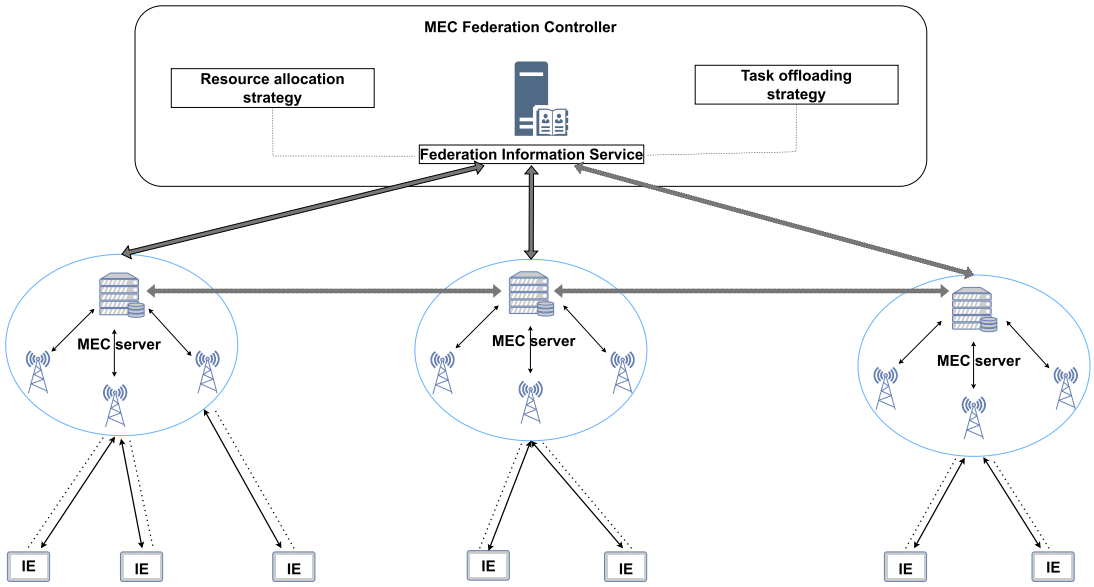


FIGURE 2. System architecture.

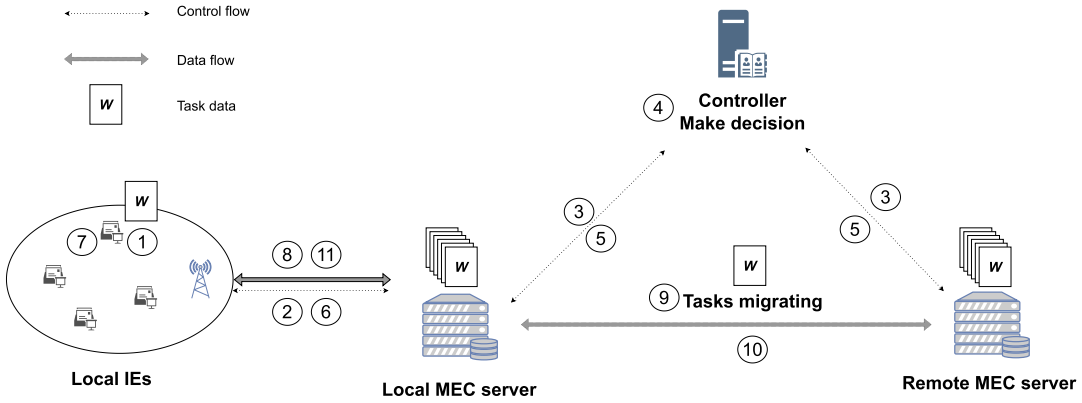


FIGURE 3. Flow of information and task processing.

send these tasks to MEC servers and receive the computation results. This increases system latency. In this study, we consider an MEC-enabled IIoT network that includes two types of communication for data transmission. IEs communicate with MEC servers via wireless connections, whereas MEC servers connect to each other through a wired network. For each type of connection, we defined both uplink and downlink. The uplink is used to send the task data, with the downlink is used to send back the resulting data.

The IEs are connected to the MEC servers via a wireless connection. Let $G_{d,s}$ denote the wireless channel gain [27] between the IE d and MEC server s, which is calculated as:

$$G_{d,s} = 127 + 30\log(dist_{d,s}),$$

where $dist_{d,s}$ denotes the distance between the IE d and MEC server s. Thus, the wireless uplink data rate $R^U_{d,s}(t)$ and downlink data rate $R^D_{d,s}(t)$ of the communication channel between the IE d and MEC servers s in time slot t are obtained by Shannon, which can be formulated as follows [23]:

$$R^U_{d,s}(t) = B^U_{d,s}(t)\log_2\left(1 + \frac{P_d(t)G_{d,s}}{N_0 B^U_{d,s}(t)}\right), \tag{3}$$

$$R^D_{d,s}(t) = B^D_{d,s}(t)\log_2\left(1 + \frac{P_s(t)G_{d,s}}{N_0 B^D_{d,s}(t)}\right), \tag{4}$$

where $B^U_{d,s}(t)$ and $B^D_{d,s}(t)$ denote the uplink and downlink channel bandwidths allocated to the wireless link between the IE d and MEC server s, respectively; $P_d(t)$ and $P_s(t)$ denote the transmission power allocated by the IE d and MEC server s, respectively; and N_0 is the Gaussian noise power spectrum density.

In contrast, the data transfer rate in a wired network corresponds to the link bandwidth. Thus, the uplink data rate $W^U_{s,r}(t)$ and downlink data rate $W^D_{s,r}(t)$ of the wired

TABLE 3. Notation list.

Notation	Definition	Notation	Definition
D	the number of IEs	S	the number of MEC servers
$I^d(t)$	the number of tasks generated by IE d	$a_i^d(t)$	data size of ith task generated by IE d
$b_i^d(t)$	result size of ith task generated by IE d	$c_i^d(t)$	required number of CPU cycles of ith task of IE d
$f_i^d(t)$	computing resource allocated by IE	f_i^s	computing resource allocated by MEC server
$T_i^d(t)$	computing delay of IE	T_i^s	computing delay of MEC server
$G_{d,s}$	wireless channel gain between IE and MEC server	N_0	Gaussian noise power spectrum density
$B_{d,s}^U(t)$	uplink bandwidth allocated between IE and MEC	$B_{d,s}^D(t)$	downlink bandwidth allocated between IE and MEC
$P_d(t)$	transmission power allocated by IE	$P_s(t)$	transmission power allocated by MEC server
$R_{d,s}^U(t)$	uplink data rate between IE and MEC server	$R_{d,s}^D(t)$	downlink data rate between IE and MEC server
$B_{s,r}^U(t)$	uplink bandwidth allocated between MEC servers	$B_{s,r}^D(t)$	downlink bandwidth allocated between MEC servers
$W_{s,r}^U(t)$	uplink data rate between MEC servers	$W_{s,r}^D(t)$	downlink data rate between MEC servers
$T_{d,s}^{U,R}(t)$	uplink transmission delay between IE and MEC	$T_{d,s}^{D,R}(t)$	downlink transmission delay between IE and MEC
$T_{s,r}^{U,W}(t)$	uplink transmission delay between MEC servers	$T_{s,r}^{D,W}(t)$	downlink transmission delay between MEC servers
$SC^s(t)$	storage capacity of MEC server	$SA^s(t)$	available storage of MEC server
$T_s^q(t)$	queuing delay of MEC server	α, β	weights of the delay and energy cost
$p_i^d(t)$	IE energy consumption in one unit of time	$p_i^s(t)$	MEC server energy consumption in one unit of time
$E_i^d(t)$	energy consumption of IE for task execution	$E_i^s(t)$	energy consumption of MEC server for task execution
$E_{d,s}^{U,R}(t)$	uplink energy consumption between IE and MEC server	$E_{s,r}^{U,W}(t)$	uplink energy consumption between MEC servers
$E_{d,s}^{D,R}(t)$	downlink energy consumption between IE and MEC server	$E_{s,r}^{D,W}(t)$	downlink energy consumption between MEC servers
A, B	normalization values of the delay unit and energy unit	$C_i^d(t)$	energy-delay cost of IE execution
$C_i^{loc}(t)$	energy-delay cost of local MEC server execution	$C_i^{rem}(t)$	energy-delay cost of remote MEC server execution
$x_i^d(t)$	offloading variable	$\mathbb{C}_i^d(t)$	energy-delay cost of each task
P_{min}	lowest acceptable power level	P_{max}	highest acceptable power level
B_{max}^U	maximum uplink bandwidth between IE and MEC server	B_{max}^D	maximum downlink bandwidth between IE and MEC server
$FA^s(t)$	available computing resource of MEC server	$EA^s(t)$	available power resource of MEC server
$S_d^s(t)$	allocated storage in buffer MEC s for IE d	$w_i^d(t)$	ith task generated by IE d
ξ	effective switched capacitance	$dist_{d,s}$	distance between IE d and local MEC server s
R	radius of the area covered by each MEC server	$dist_{d,r}$	distance between IE d and remote MEC server r

connection between MEC server s and MEC server r in time slot t are obtained as follows:

$$W_{s,r}^U(t) = B_{s,r}^U(t), \tag{5}$$

$$W_{s,r}^D(t) = B_{s,r}^D(t), \tag{6}$$

where $B_{s,r}^U(t)$ and $B_{s,r}^D(t)$ denote the uplink and downlink channel bandwidths allocated to the wired link between MEC server s and MEC server r, respectively.

The uplinks from the IE to MEC server and from MEC server to MEC server are used to transmit the task data. $T_{d,s}^{U,R}(t)$ represents the wireless uplink transmission delay from IE d to MEC server s, and $T_{s,r}^{U,W}(t)$ denotes the wired uplink transmission delay from MEC server s to MEC server r. Based on (3) and (5), they are calculated as follows:

$$T_{d,s}^{U,R}(t) = \frac{a_i^d(t)}{R_{d,s}^U(t)}, \tag{7}$$

$$T_{s,r}^{U,W}(t) = \frac{a_i^d(t)}{W_{s,r}^U(t)}, \tag{8}$$

where $a_i^d(t)$ denotes the task data size.

Furthermore, downlinks from the MEC server to the IE and from the MEC server to the MEC server are used to transmit the resulting data. $T_{d,s}^{D,R}(t)$ represents the wireless downlink transmission delay from MEC server s to IE d, whereas $T_{s,r}^{D,W}(t)$ denotes the wired downlink transmission delay from MEC server r to MEC server s. Based on (4) and (6), they are calculated as follows:

$$T_{d,s}^{D,R}(t) = \frac{b_i^d(t)}{R_{d,s}^D(t)}, \tag{9}$$

$$T_{s,r}^{D,W}(t) = \frac{b_i^d(t)}{W_{s,r}^D(t)}, \tag{10}$$

where $b_i^d(t)$ denotes the size of the task results.

3) QUEUING DELAY

The MEC server uses a task buffer to temporarily hold the incoming tasks that have not yet been handled. The buffer works on a first-in-first-out basis, and it is usually assumed that its capacity is sufficiently large, such that it can always accommodate all tasks transmitted to the MEC server. When new tasks are offloaded to an edge server, they must be stored in the buffer while waiting for the previous tasks to be completed [10]. For simplicity, queue delay is usually ignored [16], [23]. However, this approach is unreasonable,

particularly in cases of severe congestion. According to [28], it is typically necessary for each server to be equipped with a minimum of four configured cores. It is assumed that the CPU of each server possesses the capability to process four cores, thereby enabling the simultaneous execution of four tasks in parallel. Therefore, $M/M/c/K$ queuing model is probably a more realistic model because it sets the capacity of the buffer at $K = SC^s(t)$ and the number of services is four corresponding 4-core processing of the CPU, as shown in Fig. 4. The arrival distribution of tasks follows a Poisson distribution, and the distribution of service time follows an exponential distribution with $c = 4$ as the number of parallel cores [29]. With λ_s defining the average arrival rate of tasks entering the buffer and β_s defining the service rate of the MEC server [11], given the $M/M/4/SC^s$ model, the probability that i tasks exist in the buffer is defined by P_i [30]. Subsequently, the queue delay $T_s^q(t)$ for each task i in the MEC server s buffer can be expressed mathematically as follows:

$$T_{s,i}^q(t) = \frac{SC^s(t) - SA^s(t)}{\lambda_s(1 - P_{SC^s})}, \tag{11}$$

where $SC^s(t)$ and $SA^s(t)$ correspond to the capacity and available storage of MEC server s in time slot t, then $SC^s(t) - SA^s(t)$ is number of tasks in waiting line for executing, P_{SC^s} is probability that total SC^s tasks in the queuing line.

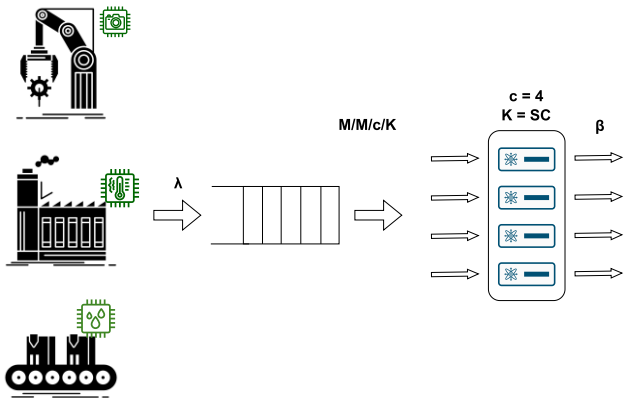


FIGURE 4. Queuing model for IIoT task executing by MEC.

D. ENERGY CONSUMPTION MODEL

1) COMPUTATION ENERGY CONSUMPTION

The devices in the IIoT system are highly energy constrained, whereas the MEC servers that provide computational support for IEs also use billed power. Thus, the reduction in energy consumption cannot be disregarded. The power in each time unit consumed by the IE d and MEC servers s for executing task $w_i^d(t)$ is as follows [31]:

$$p_i^d(t) = \xi[f_i^d(t)]^3,$$
$$p_i^s(t) = \xi[f_i^s(t)]^3,$$

where ξ is the effective switched capacitance which depends on chip architecture. Therefore, the energy consumption of the IE d or MEC server s is calculated as follows:

$$E_i^d(t) = p_i^d(t)T_i^d(t), \tag{12}$$
$$E_i^s(t) = p_i^s(t)T_i^s(t). \tag{13}$$

2) TRANSMISSION AND RECEPTION ENERGY CONSUMPTION

Each IE or MEC server uses the transmission energy to support offloading or migrating tasks. The total transmission power consumption for all connections, including wireless and wired, comprises the energy used by the sender to transmit data and the energy consumed by the receiver to gather data. Hence, $E_{d,s}^{U,R}(t)$ and $E_{s,r}^{U,W}(t)$ are the total transmission energies of the uplink between the IE d - local MEC server s and the local MEC server s - remote MEC server r, which are calculated as:

$$E_{d,s}^{U,R}(t) = T_{d,s}^{U,R}(t)P_d(t) + T_{d,s}^{U,R}(t)P_s(t), \tag{14}$$
$$E_{s,r}^{U,W}(t) = T_{s,r}^{U,W}(t)P_s(t) + T_{s,r}^{U,W}(t)P_r(t). \tag{15}$$

On the other hand, an IE or MEC server require energy to keep the "on" mode, means able to receive tasks or results. $E_{d,s}^{D,R}(t)$ and $E_{s,r}^{D,W}(t)$ are the total transmission energies of the downlink between the IE d - local MEC server s and between local MEC server s - remote MEC server r, respectively. Similar to (14) and (15), $E_{d,s}^{D,R}(t)$ and $E_{s,d}^{D,R}(t)$ are calculated as follows:

$$E_{d,s}^{D,R}(t) = T_{d,s}^{D,R}(t)P_d(t) + T_{d,s}^{D,R}(t)P_s(t), \tag{16}$$
$$E_{s,r}^{D,W}(t) = T_{s,r}^{D,W}(t)P_s(t) + T_{s,r}^{D,W}(t)P_r(t). \tag{17}$$

E. ENERGY-DELAY COST

This study investigates an IIoT network comprising several IoT devices with limited resources for managing significant volumes of data and computationally intensive tasks. This adversely affects the system latency and power consumption. Thus, it is crucial to optimize the latency and power utilization simultaneously. We consider three computation strategies for each IE task: IE computation, local MEC server computation, and remote MEC server computation. Instead of focusing on a "offload or not" offload strategy, we focus on multi-user multi-MEC scenarios, which not only simplifies the choice of "offload or not", but also solves problem "offload to which one". The offloading strategy defines the execution of a task:

$$x_i^d(t) = \begin{cases} x_i^d(t) = 0 & \text{for local IE,} \\ x_i^d(t) = s & \text{for local MEC server s,} \\ x_i^d(t) = r \neq s & \text{for remote MEC server r.} \end{cases} \tag{18}$$

In each case, the latency-energy cost function is defined as follows.

1) IE

In this case, the tasks $w_i^d(t)$ are executed only by IE d itself and are not offloaded to any MEC server ($x_i^d(t) = 0$). The delay of each IE is the total computing time for all tasks,

whereas energy consumption is the energy used for task computation. The energy-delay cost of task $w_i^d(t)$ is calculated based on (1) and (12) as follows:

$$C_i^d(t) = \alpha \cdot A \cdot T_i^d(t) + \beta \cdot B \cdot E_i^d(t). \tag{19}$$

where α and β are the constant weighting parameters corresponding to the delay and energy costs of the task, respectively. A and B are values that normalize the delay and energy units to the cost unit, respectively, [32].

2) LOCAL MEC SERVER

The MEC server s is defined as local when it satisfies $dist_{d,s} \leq R$, where R is the radius of the area covered by each MEC server.

In this case, all IE tasks are offloaded to the local MEC server. Technically, $x_i^d(t) = s$, where s is the local MEC server that connects directly to IE d. We consider the following three types of delays:

- **Transmission delay**: delay of sending the task data and receiving results via connection between IE d and local MEC sever s
- **Computation delay**: delay of task computing by local MEC server s
- **Queuing delay**: delay for waiting in the buffer of local MEC server s.

According to (2), (7), (9), and (11), the total delay for computing task $w_i^d(t)$ in time slot t is determined as follows:

$$T_i^{loc}(t) = T_i^s(t) + T_{d,s}^{U,R}(t) + T_{d,s}^{D,R}(t) + T_s^q(t). \tag{20}$$

We consider energy consumption as:

- **Transmission and reception energy**: energy for sending and receiving task and result data of the IE d and local MEC server s.
- **Computation energy**: energy for computing the task by MEC server s

Based on (13), (14), and (16), the total energy consumption in time slot t can be determined as:

$$E_i^{loc}(t) = E_i^s(t) + E_{d,s}^{U,R}(t) + E_{s,d}^{U,R}(t) + E_{d,s}^{D,R} + E_{s,d}^{D,R}. \tag{21}$$

The energy-delay cost of each IE task in a time slot t is the combination of energy consumption and delay cost when offloading the task to the local MEC server s as follows:

$$C_i^{loc}(t) = \alpha \cdot A \cdot T_i^{loc}(t) + \beta \cdot B \cdot E_i^{loc}(t). \tag{22}$$

3) REMOTE MEC SERVER

The MEC server r is defined as local when satisfy $dist_{d,r} > R$, where R is the radius of the area covered by each MEC server.

In this strategy, a remote MEC server executes the IE task. In this case, $x_i^d(t) = r \neq s$, where r is an MEC server not directly connected to IE d. We consider three types of delays:

- **Transmission delay**: delay of sending and receiving the tasks and results via the connection of IE d - local MEC server s and local MEC server s - remote MEC server r
- **Computation delay**: delay of task computing by remote MEC server
- **Queuing delay**: delay for waiting in the buffer of remote MEC server.

The total delay of the remote MEC server r is calculated as:

$$T_i^{rem}(t) = T_{d,s}^{U,R}(t) + T_{s,r}^{U,W}(t) + T_r^q(t) + T_i^r(t) + T_{s,r}^{D,W}(t) + T_{d,s}^{D,R}(t).$$

In this strategy, we also consider energy consumption as:

- **Transmission and reception energy**: energy for sending and receiving task data and result data between local MEC server s and remote MEC server r
- **Computation energy**: energy for computing the task by remote MEC server r.

The energy consumption for task calculation on a remote MEC server r is calculated as follows:

$$E_i^{rem}(t) = E_{d,s}^{U,R}(t) + E_{s,r}^{U,W}(t) + E_i^r(t) + E_{s,r}^{D,W}(t) + E_{d,s}^{D,R}(t).$$

Therefore, the total energy-delay cost $C_i^{remote}(t)$ for each IE task is calculated as follows:

$$C_i^{rem}(t) = \alpha \cdot A \cdot T_i^{rem}(t) + \beta \cdot B \cdot E_i^{rem}(t). \tag{23}$$

F. PROBLEM FORMULATION

From the described cases, the general equation for the energy-delay cost of each task $w_i^d(t)$ generated by IE d in time slot t, denoted by $\mathbb{C}_i^d(t)$, can be defined as follows:

$$\mathbb{C}_i^d(t) = \begin{cases} C_i^d(t) & \text{for } x_i^d(t) = 0, \\ C_i^{loc}(t) & \text{for } x_i^d(t) = s, \\ C_i^{rem}(t) & \text{for } x_i^d(t) = r \neq s, \end{cases} \tag{24}$$

where s and r denote local and remote MEC servers, respectively.

The objective of this study is to minimize the average energy-delay cost of the entire system in T time slots, including IEs and MEC servers, considering the resource limitations and sensitive delay requirements. The objective function is expressed as:

$$\min_{x_i^d(t)} \sum_{t=1}^{T} \sum_{d=1}^{D} \sum_{i=1}^{I^d} \frac{1}{T} \frac{1}{D} \frac{1}{I^d} \mathbb{C}_i^d(t). \tag{25}$$

We also present the following constraints:

$$C1: \quad 0 \leq x_i^d(t) \leq S, \tag{26}$$

$$C2: \quad P_{min} \leq P_d(t), P_s(t) \leq P_{max}, \tag{27}$$

$$C3: \quad \sum_{d=1}^{D} B_{d,s}^U \leq B_{max}^U, \quad \sum_{d=1}^{D} B_{d,s}^D \leq B_{max}^D, \tag{28}$$

$$C4: \quad SA^s(t) \geq \sum_{d=1}^{D} S_d^s(t)\, S_d^s(t) \geq 0, \tag{29}$$

$$C5: \quad FA^d(t) \geq \sum_{i=1}^{I^d} f_i^d(t) f_i^d(t) \geq 0,$$

$$FA^s(t) \geq \sum_{d=1}^{D} f_i^s(t) f_i^s(t) \geq 0, \tag{30}$$

$$C6: \quad EA^d(t) \geq \sum_{i=1}^{I^d} E_i^d(t) \tag{31}$$

$$C7: \quad \{T_i^d(t), T_i^{loc}(t), T_i^{rem}(t)\} \leq \tau_i^d. \tag{32}$$

Constraint 1 guarantees that the offloading decision can select one of the three options: IE, local MEC server, or migrate to a remote MEC server. Constraint 2 states that the transmission and reception powers of the IEs and MEC servers should not be exceeded; P_{max} and P_{min} are the lowest and highest acceptable powers for the system to operate stably [33], respectively. According to Constraint 3, the total downlink and uplink bandwidths between all the IEs and the local MEC server should not exceed the maximum bandwidth. We denote $S_d^s(t)$ as the storage resource allocated by the MEC server s for IE d in time slot t. Constraint 4 states that the storage resources used for all tasks in time slot t must not exceed the storage buffer's availability, and the storage resources allocated must be positive. We denote $FA^s(t)$ and $FA^d(t)$ as the the available computing resources of the MEC server s and IE d. The total computing resources allocated to all tasks must not exceed the amount of resources presently available, as specified by Constraint 5 [23]. We denote $EA^d(t)$ as the available energy resource of IE d in time slot t, and Constraint 6 indicates that the total energy resource allocated to all tasks must not exceed the number of resources currently available of IE d [34]. The energy resources of MEC servers are unlimited, indicating that there are no constraints on their energy resources of MEC servers. Constraint 7 states that the maximum completion duration for each task is no longer than the required time [23]. In the next section, we demonstrate that optimization problem (25) satisfies the characteristics of a Mixed-Integer Nonlinear Programming (MINLP) problem.

Proposition 1: The optimization problem (25) is MINLP.

Proof: Mathematically, the mixed integer nonlinear programming (MINLP) problem looks like:

Maximize or Minimize:

$$f(x) + d(y)$$

Subject to:

$$g(x) + h(y) \quad \alpha \quad 0,$$
$$L \leq x \leq U,$$
$$y = \{0, 1, 2, \ldots\},$$

where x is a vector of variables that are continuous real number, $f(x) + d(y)$ is the objective function, $g(x) + h(y)$ represents the set of constraints, α is some mixture of $\leq$, $=$, and $\geq$ operators, L and U are vectors of lower and upper bounds on the variables.

To be a MINLP problem, problem (25) needs to have the characteristic properties of non-linearity and the presence of integer variables. Due to the multiplication of delayed cost $\mathbb{C}_i^d(t)$ with other fractions and the subsequent summation, the objective function incorporates non-linear terms. In addition, it should be noted that the offloading decision variable, denoted as $x_i^d(t)$, is an integer type, whereas the resource allocation and power control mechanisms exhibit dynamic variability. Consequently, objective function (25) is classified as a MINLP problem.

The computational complexity of MINLP problems results from the complex process of searching for an optimal solution, which can prove to be a daunting task due to the combination of non-linearity and integer variables, especially given the high latency requirements of IIoT tasks. The aforementioned solution, on the other hand, requires the controller to make judgments after collecting information. Therefore, we propose a DRL technique based on the integration of DDPG and PER buffer to reduce the energy-latency cost of the system. A carefully trained DRL model promises to deliver optimal results in the shortest possible time.

IV. METHODOLOGY

This section introduces the proposed framework for task offloading and resource allocation to an MEC federation system in an IIoT network. The optimization problem (25) is first modeled as an MDP problem. Eventually, the DDPG model is considered a solution to the MDP problem owing to its capacity to operate with large data dimensions and adapt to highly dynamic system states. Nevertheless, DDPG has several issues owing to the complexity of the training phase. Therefore, we propose combining DDPG with PER to enhance the speed and efficiency of DDPG training. Finally, we present the proposed DDPG-PER-based task offloading and resource allocation framework for optimizing the power consumption and delay of an MEC-enabled IIoT network.

A. MDP

To apply DRL to solve the optimization problem, we define five elements of MDP: state space S, action space A, state transition probability P, reward function R, and discount factor γ. As shown in Fig. 5, when time slot t begins, the agent must decide on the action by choosing $a_t \in A$ based on the current system condition $s_t \in S$. After performing this action, the environment updates the next state s_{t+1} of the system and receives a reward as feedback from the environment to evaluate the effectiveness of the action. Based on the interaction between the agent and environment, the agent can develop a policy μ that maps from state to action, μ: $S \rightarrow P(A)$, meaning that $a_t = \mu(s_t)$.

1) STATE SPACE

When time slot t begins, the agent obtains information regarding the state of the current system environment. We denote the

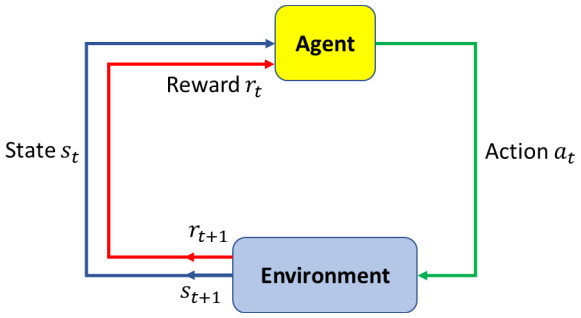


FIGURE 5. Agent-Environment interaction in MDP.

system state in time slot t as follows:

$$s_t = \{K(t), D(t), E(t), F(t), S(t)\},$$

where $K(t)$ and $D(t)$ indicate the location and size of all tasks in the system, respectively, $E(t)$ and $F(t)$ represent the available power and computing resources of all IEs and MEC servers, respectively, and $S(t)$ is the available storage resources of all MEC servers.

2) ACTION SPACE

After considering the current state s_t, the agent makes the following decision:

$$a_t = \{P(t), f(t), B(t), s(t), X(t)\},$$

where $X(t)$ indicates all task offloading decisions, and $P(t)$, $f(t)$, $B(t)$, and $s(t)$ represent the allocated power, computing, bandwidth, and storage resources in each time slot, respectively. Considering that DDPG requires action to be continuous, we represent the action that offloads the decision of each task as

$$\{e_0, e_1, \ldots, e_S\},$$
$$\text{s.t.} \sum_{i=0}^{S} e_i = 1,$$

where $e_{i\neq 0}$ is the probability that the task is offloaded to the MEC server i, and e_0 indicates that the task is executed by the IE itself. Hence, the offloading variable for each task is determined by the index of the largest probability value in $\{e_0, e_1, \ldots, e_S\}$.

3) STATE TRANSITION PROBABILITY

$p(s_{t+1}|s_t, a_t)$ represents the probability of s_{t+1} given s_t and the chosen a_t. Moreover, the agent has no prior knowledge of $p(s_{t+1}|s_t, a_t)$, which is solely determined by the environment [24].

4) REWARD FUNCTION

The reward r_t represents the immediate reward received when performing action a_t in the state s_t. The objective of (25) is to minimize the energy-delay cost. Therefore, we assume that the reward r_t in each time slot t is negative for the average cost of the IE task, as follows:

$$r_t = -\sum_{d=1}^{D}\sum_{i=1}^{I^d} \frac{1}{D}\frac{1}{I^d}\mathbb{C}_i^d(t). \tag{33}$$

Owing to the large dimensionality of the optimization problem, the DRL method is considered an approach for finding the optimal policy that maximizes the long-term reward $R_t = \sum_{t=1}^{T} \gamma^{t-1} r_t$. As mentioned in Section I, since Deep Q-Learning solely focuses on discrete actions, DDPG is able to overcome the limitations of dealing with continuous domains. Owing to the continuity of the state space and actions in the modeled MDP problem, we introduce a DDPG-based approach to solve the resource allocation problem.

B. DDPG ALGORITHM

The DDPG architecture comprised four deep neural networks (DNNs) and four trainable parameters for distinct purposes. The actor network θ^{μ} is responsible for optimizing the policy $\mu(\theta^{\mu})$ that provides the optimal action for a given state, whereas the critic network θ^{Q} critiques the quality of an action as indicated by its Q-value. In addition, the target actor network $\theta^{\mu'}$ and the target critic network $\theta^{Q'}$ are time-delayed versions, allowing the model to maintain more consistent estimated targets. The operation of DDPG is described in detail below.

Each policy μ defines an action-state pair value function $Q^{\mu}(s_t, a_t)$ which indicates the expected return of action a_t execution given state s_t. Based on Bellman's equation [23], Q-value is:

$$Q^{\mu}(s_t, a_t) = E[r_t + \gamma Q(s_{t+1}, \mu(s_{t+1})],$$

whereas the return is determined by

$$R|s_t, \mu_t = r_t + \gamma Q(s_{t+1}, \mu(s_{t+1}).$$

Thus, the outputs of the actor and critic networks are determined as follows:

$$\mu(s_t|\theta^{\mu}) \approx \mu^*(s_t),$$
$$Q(s_t, a_t|\theta^{Q}) \approx Q(s_t, a_t).$$

Additionally, random noise ϱ_t is added to the action output of the actor network at each time step to obtain better action exploration.

Our training sample data is obtained from a replay memory buffer R of finite size V. The tuple $\{s_t, a_t, r_t, s_{t+1}\}$ is saved in R for each time slot t. By randomly selecting a mini-batch $\{s_i, a_i, r_i, s_{i+1}\}$ of data from R, the critic and actor networks update the parameters for each time slot t. Let M represent the amount of data in each mini-batch. The difference between Q-value and target Q-value δ, called temporal difference error (TD-error), is calculated as follows:

$$\delta = Q(s_i, a_i|\theta^{Q}) - y_i. \tag{34}$$

where y_i denotes target Q-value:

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}). \tag{35}$$

Thus, the loss function of the critic network is:

$$L(\theta^Q) = \frac{1}{M}\delta^2, \tag{36}$$

The critic network is updated using the Adam optimizer [35] to minimize the loss function. Reference [36] showed that the policy gradient may assist in obtaining the highest anticipated reward provided that the model parameters are updated using the following gradient equation:

$$\nabla_{\theta^\mu} J_{\mu*} = \frac{1}{M}\sum_{i=1}^{M} \nabla_{a_i}[Q^\mu(s_i, a_i|\theta^Q) \nabla_{\theta^\mu \mu(s_i|\theta^\mu)}]. \tag{37}$$

The DDPG algorithm uses θ^μ and θ^Q of the current actor and critic networks and soft updates them as follows:

$$\begin{cases} \theta^{\mu'} \leftarrow (1-\delta)\theta^{\mu'} + \delta\theta^\mu \\ \theta^{Q'} \leftarrow (1-\delta)\theta^{Q'} + \delta\theta^Q. \end{cases} \tag{38}$$

C. PER

The experience replay buffer is a crucial component of DDPG because it utilizes previous experience to prevent the high correlation of input data [25]. However, DDPG uses uniform experience playback and treats all samples as equally important. This is unreasonable, because it ignores the variance in the results for each sample. The PER-assisted DDPG approach selects a metric with an absolute TD-error to depict the value of each sample accurately. To avoid overfitting, stochastic prioritization was used to select the experiences. In addition, importance-sampling weights have been employed to correct the state visitation bias produced by prioritized sampling [25].

In stochastic prioritization, the probability of the sampling experience j is defined as follows [25]:

$$P(j) = \frac{D_j^\mu}{\sum_i D_i^\mu}, \tag{39}$$

where $D_j = \frac{1}{rank(j)} > 0$ indicates the priority of experience j and $rank(j)$ is the rank of experience j in the replay buffer. The offset value ϵ is added to the priority to prevent case priority from being equal to 0; thus, $D_j = |\delta_j| + \epsilon$. The absolute TD-error of experience is considered as the priority index and is used to demonstrate the importance of experience. However, this model only replayed the experience with the highest TD-error priority, which led to a loss of diversity [37].

Furthermore, the IS weight is introduced to reduce the magnitude of the gradient and thus make the training process more stable. The IS weight [26] was calculated as follows:

$$w_j = \frac{1}{V^v.P(j)^v}, \tag{40}$$

where V is the buffer size and $v \in [0, 1]$. In addition, it normalizes the weights by $\frac{1}{max_i w_i}$ to scale the update downwards [25]. Therefore, the differentiation loss function is transformed as follows:

$$w_j \nabla_{\theta_1^Q} L(\theta_1^Q) = w_j.\delta_j. \nabla_{\theta_1^Q}.Q(s, a|\theta_1^Q).$$

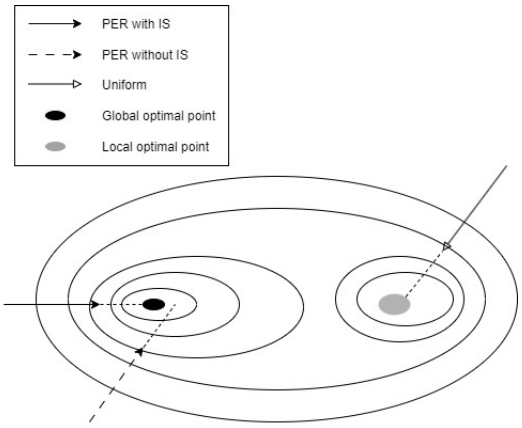


FIGURE 6. Illustration of the performance of different training processes.

Fig. 6 illustrates the convergence of different experience sampling methods such as uniform sampling, PER without IS, and PER with IS. The space curve created by the loss function of all events is indicated by contour lines. The PER with IS weights enables the replay to gravitate toward encounters with higher TD-error magnitudes. Because of the restrictions imposed by the IS weights, the training process was quite steady. Without IS weights, the PER obtains larger step sizes and moves toward a local optimum solution. Moreover, training procedures can fluctuate. The PER offers a more comprehensive and reliable solution to DDPG issues by integrating IS weights to accelerate the training process.

D. DDPG-PER-BASED ALGORITHM

To address the issues of task offloading and resource allocation for the MEC federation system in the IIoT network, we propose a framework based on DDPG-PER. This model is implemented in the controller, which is responsible for receiving the system state and making optimal decisions. As shown in Fig. 7, the proposed framework includes four DNNs: the actor, critic, target actor, and target critic.

The proposed training procedure comprises H episodes. The system state is initialized at the beginning of each episode, and the optimal resource allocation option is determined after the last episode. In each episode, the model is trained for a duration T. At the beginning of each time slot, the MEC controller provides a noise-added output to the agent network. This involves determining where to execute all tasks and the resources allocated to perform them, including computation, transmission, and storage resources. The system then aggregates the delay and energy usage for each task in the IIoT system. The MEC controller calculates the reward for time slot t using (33) at the end of time slot t and gathers the system state for time slot $t+1$. $s(t)$, $a(t)$, $r(t)$, and $s(t+1)$ are stored for future training in replay buffer R. The model updates the neural networks at each time slot after V first time slots. First, the model revises the priority of M experiences, where M is the mini-batch size. Each experience j is sampled with a probability derived from

Algorithm 1 DDPG-PER-Based Algorithm

Input: Tasks size, Tasks location, Resource available
Output: Cost optimization, Task offloading and Resource allocation strategy
Initialize critic $Q(s, a|\theta^Q)$, weight θ^Q and actor $\mu(\theta^\mu)$, weight θ^μ
Initialize target critic Q', weight $\theta^{Q'} \leftarrow \theta^Q$ and target actor μ', weight $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R, size V, mini-batch M
Set priority $D_1 = 1$, exponents u, v
Initialize the number of episodes H
for *episode* $= 1, H$ **do**
    Initialize random process ϱ_t
    Initialize the system environment and receive initial observation state s_1
    for $t = 1, T$ **do**
        Select action $a_t = \mu(s_t|\theta^\mu) + \varrho_t$
        Execute action a_t and observe reward r_t and new state s_{t+1}
        Store experience (s_t, a_t, r_t, s_{t+1}) in R, set maximal priority $D_t = max_{i<t} D_i$
        if $t > V$ **then**
            for $j = 1, M$ **do**
                Sample experience j with probability (39)
                Compute IS weight (40)
                Set target Q-value (35)
                Compute TD-error (34)
                Update the priority of experience j according to absolute TD-error $|\delta_j|$
            end for
            Update critic network by minimizing the loss function (36)
            Update actor network using the sampled policy gradient (37)
            Update target networks (38)
        end if
    end for
end for

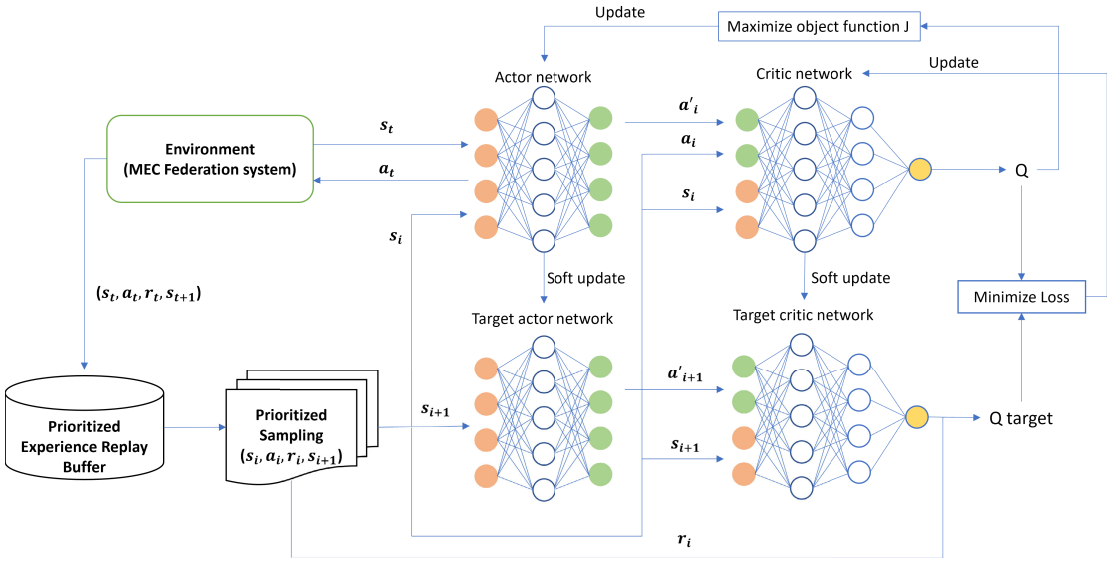


FIGURE 7. MEC federation resource allocation framework based on DDPG-PER.

Equation (39). The IS weight, target Q-value, and TD-error are then calculated using Equations (40), (35), and (34), respectively. Finally, the priority of experience j is computed based on the absolute value of the TD-error. After updating the priority in the priority experience buffer, the model minimizes the loss function (36) using the TD-error calculated above and weights while training the actor network with the sampled policy gradient (37). According to Equation (38), the model updates the target actor and critic networks. Details are presented in Algorithm 1.

V. EXPERIMENTS

In this section, we evaluate the performance of the proposed framework in terms of task offloading and resource allocation. Initially, investigations are conducted to determine the impact of certain parameters such as the learning rate and number of IEs in the system on the training process. In addition, the optimize results of the DDPG-PER model are compared with those of other strategies to demonstrate that the proposed method can simultaneously improve the latency and energy consumption of the IIoT system. Our simulation

is conducted on a PC with a 2.60 GHz Intel(R) Core(TM) i5-11400F processor, 16 GB of RAM, and Windows 11 (64-bit). The DRL methodology under consideration is executed using Python 3.7, Tkinter 8.6 and Tensorflow 2.6. The environment is established in Anaconda.

A. EXPERIMENTAL SETTINGS

This study examined an MEC-enabled IIoT network that allows MEC servers to collaborate. This IIoT system has ten 500-meter-radius industrial zones, each of which is covered by a single MEC server [23]. In each of these areas, the IIoT devices are randomly deployed and connected only to the MEC server of that region, which is called the local MEC server. There are two types of connections between devices in the system: a wireless connection between the IE and local MEC server and a wired connection between the MEC servers. The wireless uplink and downlink have the same bandwidth capacity of 20 MHz [10], while the maximum rate of the wired connections is 150 MBps [38]. Each IE has a battery capacity of 1000 J and a guaranteed power threshold of 20 J [23]. We set the buffer size of each MEC server to 25 GB, according to [39]. Each IIoT device with limited resources has a computational capability of 5 GHz [40], whereas the MEC server has a capacity of 25 GHz [40]. In our experiments, we separated the system operation into 3000 time periods. Because IIoT devices must conduct monitoring-related tasks that require continuous data collection and processing, the length of each time slot is set to 0.1 seconds to guarantee efficient system performance. IEs produce tasks in each time slot in accordance with a uniform distribution generated in the interval [1, 10] [23]. In addition, we assume that the input and output sizes of all tasks are 1 MB and 1 MB [7], respectively. According to [33], the effective transmission power range of a system is [5, 38] dBm. The power spectral density of the Gaussian noise is -174 dBm/Hz [40]. At the conclusion of the episode, the reward value is obtained before the completion of the procedure. The key simulation parameters are presented in Table 4.

In the proposed DDPG-PER model, the objective of the training process is to determine the optimal decision for task offloading and resource allocation for each IE task to minimize the energy-delay cost. Our model comprises four 5-layer networks with an input layer, an output layer, and three hidden layers. Technically, the DDPG-PER continually updates the model throughout system operation. Consequently, the proposed framework adapts to continually changing environments. Additionally, at each step, the exponential moving average (EMA) [41] model is utilized to evaluate the trends of the parameters during training. Based on this information, the model-favoring training parameters are modified. Both the batch and mini-batch size values are 32. Training begins as the sample fills the memory, and the DNN is trained at each step to acquire a sufficient number of fresh data samples. Because DRL is a self-learning method, the dataset, known as experience in DRL, is created by exploring the surroundings. Consequently, the dataset in this study is defined as the mini-batch that is sampled from the replay buffer on a regular basis. The model parameters are presented in Table 5.

TABLE 4. System model parameters.

Parameters	Value
Number of MEC servers	10
Number of IEs	10, 20, 30, 40
Industrial zone radius	500 m [23]
Channel bandwidth between device-MEC server	[0, 20] MHz [10]
Wired transmission rate between 2 MEC servers	[0, 150] MBps [23]
Battery range of device	[20, 1000] J [23]
The buffer size range of MEC server	[0, 25] GB [39]
Computing resource range of IE	[0, 5] GHz [40]
Computing resource range of MEC server	[0, 25] GHz [40]
Task data size	1 MB [7]
Result data size	1 MB [7]
Power range	[5, 38] dBm [33]
Gauss noise power spectrum N_0	-174 dBm/Hz [40]
Required number of CPU cycles for each task	737.5 cycles/bit [40]
The required time to finish 1-bit task	2.10^{-8} s [23]
The effective switched capacitance ξ	10^{-27} [40]

TABLE 5. Training model parameters.

Parameters	Value
Sample batch size V	32
Mini-batch size M	32
Replay memory buffer size	10000
Episode	100
Steps in each episode	3000
Reward discount factor γ	0.1, 0.5, 0.9
Learning rate	0.01, 0.001, 0.0001
Soft replacement τ	0.01

B. PARAMETER ANALYSIS

We first investigate the impact of parameters such as the learning rate and discount factor on the training process of the DDPG-PER-based resource allocation scheme. This evaluation enables the selection of the experimental parameters that best fit the model. In addition, we assess the effect of the number of IEs and the weights of latency and power consumption on the results of the proposed method.

1) IMPACT OF LEARNING RATE

The DDPG-PER model is composed of neural networks to overcome the issue of data dimensionality and boost the performance of system. The learning rate defines the pace at which the model adapts to training data. Thus, selecting an inadequate learning rate affect the convergence or performance of the model. Fig. 8 shows the energy-delay cost incurred during training at various learning rates. The experiments are conducted using specific learning rates of 0.01, 0.001, and 0.0001. In the first 10 epochs, it is evident

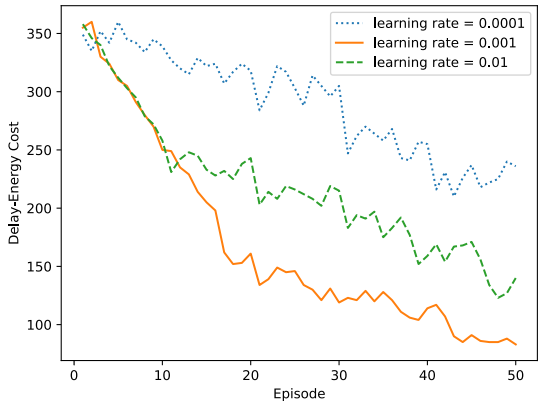


FIGURE 8. Impact of learning rate on training process.

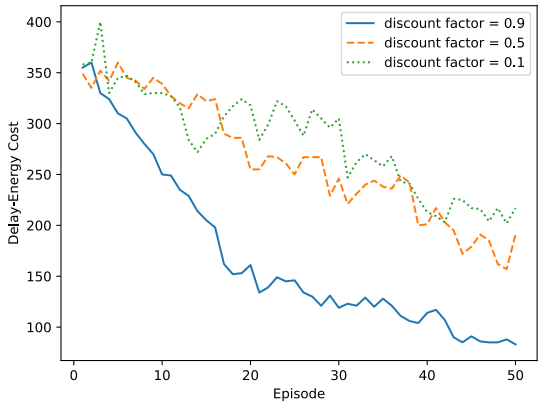


FIGURE 9. Impact of discount factor on training process.

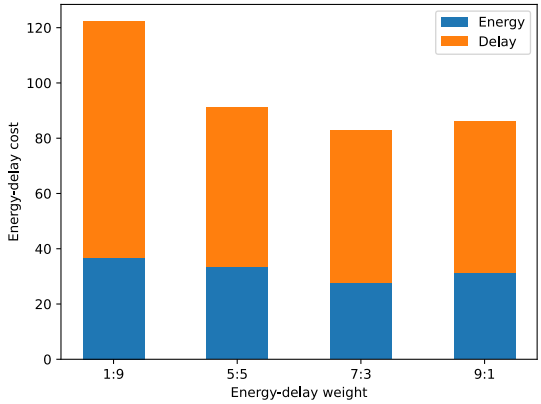


FIGURE 10. Impact of weight on delay and energy consumption.

that the cost significantly decreases with learning rates of 0.01 and 0.001, whereas it decline slightly with a learning rate of 0.0001. In the next 10 epochs, the model with a learning rate of 0.001 reduces the energy-delay cost from approximately 250 to 150. Meanwhile, at epoch 20, the costs corresponding to the learning rates of 0.01 and 0.0001 are 200 and 300, respectively. From epoch 20 onward, the models with varying learning rates exhibit the same declining pattern. As seen in Fig. 8, when the learning rate is 0.001, the average energy-delay cost curve converges to a value of 83 at episode 50. Meanwhile, after 50 episodes with learning rates of 0.01 and 0.0001, the model cannot converge and has cost values of 141 and 236. The experimental results indicate that excessively high or inadequate learning rates negatively affect the convergence of the model. When the learning rate is too low, the model updates a negligible quantity of training data information. Therefore, the training process requires more time for the model to achieve the highest performance. This is demonstrated by the fact that the model with a learning rate of 0.0001 has a lower cost reduction per epoch than the other two cases during the same period. Alternatively, if the learning rate is too high, the model may surpass the optimum point to achieve the maximum performance. In conclusion, the experimental results suggest that 0.001 is the optimal learning rate for the proposed model.

2) IMPACT OF DISCOUNT FACTOR

In reinforcement learning methods, the discount factor is a major factor used to evaluate the influence of future values. Specifically, a discount factor closer to 1 suggests that future gains are more significant than current rewards, whereas a discount factor closer to 0 indicates that the present rewards are favored [23]. Therefore, determining the appropriate discount factor helps to optimize the cumulative reward of the model. We conduct experiments to investigate the convergence of the DDPG-PER model with the discount factors of 0.1, 0.5, and 0.9. In these simulations, we use the previously specified value of 0.001 for the learning rate. As shown in Fig. 9, the model with a discount factor of 0.9 drastically reduces the energy-delay cost after 20 epochs, from 350 to approximately 150. The cost then decreases steadily until the model converges at epoch 50. The cost value of the model declines slightly during training for discount factors of 0.1 and 0.5 and does not converge at epoch 50. Even with a discount factor of 0.1, the training process becomes unstable when the cost value fluctuates significantly from epoch 10 to epoch 40. The rationale for this situation is that the model focuses only on benefits in the immediate stage. This prevents the model from obtaining significantly greater future rewards. For the proposed model, a discount factor of 0.9 is the most appropriate value.

3) IMPACT OF THE ENERGY-DELAY WEIGHT

With IIoT systems comprising many devices with limited resources, it is crucial to minimize both latency and power consumption simultaneously. Nevertheless, the priority of optimum delay or power consumption is determined by the functionality of the system. For systems that do not require stringent processing deadlines for tasks, energy optimization becomes more significant because it contributes to a more stable system operation. In contrast, systems that demand low latency prioritize lowering the task processing time. In this study, we define the energy-delay cost as the cost of executing each IE task, considering both energy consumption and

latency simultaneously, as shown in Equation (24). Specifically, the energy-delay weight is the ratio of the energy consumption to the delay, which has a significant impact on the cost per task. Therefore, we conduct experiments to evaluate the effect of the energy-latency weight on the average latency and energy consumption of each IE task. The examination is conducted based on weight ratios of 1 : 9, 5 : 5, 7 : 3, and 9 : 1. As seen in Fig. 10, it is evident that the 1 : 9 ratio incurs the highest cost of 120.3. Each task in a system with the 5 : 5 ratio costs 91.22, whereas that of the 9 : 1 ratio is 86.1. Compared with the other ratios, the energy-delay cost of a task in the IIoT system using the 7 : 3 ratio is the lowest, reaching only 82.75. Consequently, the weight ratio 7 : 3 is suitable for the proposed model.

C. COMPARISON EXPERIMENTS

This study considers task offloading and resource allocation for IIoT networks enabled by MEC. The optimal objective of the proposed method is to minimize energy consumption and delay concurrently, both of which are crucial for resource-constrained systems. Therefore, we conduct experiments to evaluate the model performance under different scenarios. the comparison is conducted based on two aspects: task offloading and resource allocation.

The following approaches are compared in terms of task execution.

- **IE**: The IE task is handled by IE itself, and MEC servers do not support it.
- **Dedicated MEC**: The IE task is offloaded to the MEC server directly linked to IE, and the MEC servers operate independently.
- **MEC federation with greedy scheme (Greedy MECF)**: The IE tasks can be transferred to local MEC server, and MEC servers collaborate only when the local MEC server is overloaded.
- **MEC Federation with our scheme (Optimal MECF)**: The IE task may be transferred to distant MEC servers, and the MEC servers collaborate to get the optimal energy-delay cost even if the local MEC server has not yet been overloaded.

This investigation compares various optimal decision-making approaches using the current state of the IIoT system as the input. The optimal decision-making process encompasses both resource allocation and task offloading.

- **Random Resource Allocation (RRA)**: To reduce the complexity of the model, resources for IEs and MEC servers are allocated randomly, and then the task offloading decision is given by the DDPG-PER model.
- **Uniform Resource Allocation (URA)**: The resources for the IE and MEC servers are set as the average of the upper and lower bounds.Additionally, the resources allocated for processing each task are distributed equally. The offloading decision is determined by the DDPG-PER algorithm.

TABLE 6. Average delay (s) of task execution comparison.

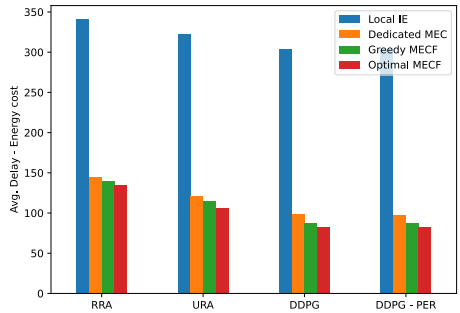
Number of IEs		10	20	30	40
RRA	Local IE	4.78	9.22	14.9	18.5
	Dedicated MEC	1.5	2.45	3.99	5.58
	Greedy MECF	1.47	2.40	3.35	5.26
	Optimal MECF	1.43	2.32	2.78	4.52
URA	Local IE	4.52	8.56	14.28	17.98
	Dedicated MEC	1.3	2.35	3.78	5.51
	Greedy MECF	1.27	2.14	3.56	4.98
	Optimal MECF	1.14	2.11	3.4	4.46
DDPG	Local IE	4.25	8.11	12.18	17.30
	Dedicated MEC	1.03	2.11	3.31	4.19
	Greedy MECF	0.91	1.98	2.97	3.88
	Optimal MECF	0.86	1.75	2.69	3.37
DDPG-PER	Local IE	4.26	8.05	12.06	17.28
	Dedicated MEC	1.01	2.11	3.3	4.12
	Greedy MECF	0.90	1.94	2.94	3.85
	Optimal MECF	**0.86**	**1.73**	**2.68**	**3.37**

- **DDPG**: The DDPG model [16], [23] determines the optimal resource allocation and task offloading decisions.
- **DDPG-PER**: The DDPG-PER model is responsible for making the optimal decision in terms of resource allocation and task offloading.

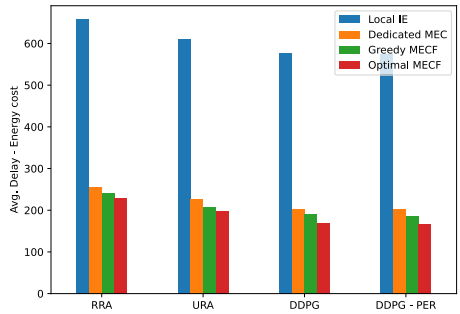
Specifically, we investigate the four decision-making approaches for each of the four task-processing strategies listed above. In addition, studies have been conducted using IEs of 10, 20, 30, and 40. This enables a more comprehensive review of the model performance under various scenarios.

1) PERFORMANCE COMPARISON OF DELAY

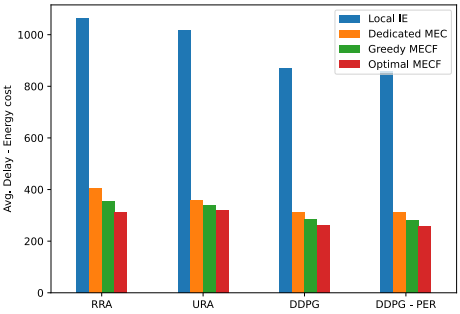
Table 6 presents the comparative results for the average execution latency. It is evident that when the number of IEs in a system increases, the average latency to accomplish a task increases with every algorithm. Owing to limited system resources, increasing the number of IEs exhausts the resources allocated to each task, leading to greater delays. In addition, when using a dedicated MEC and MEC federations, the average latency for each task decreased considerably. In particular, with an IIoT system comprising 10 IEs using DDPG-PER, the latency for a task with our proposed system is 0.86 s, whereas the average latency for a task processed in the IE, Dedicated MEC, and Greedy MECF is 4.26, 1.01 and 0.90 s, respectively. Systems using the DDPG, URA and RRA approaches exhibited similar patterns. With



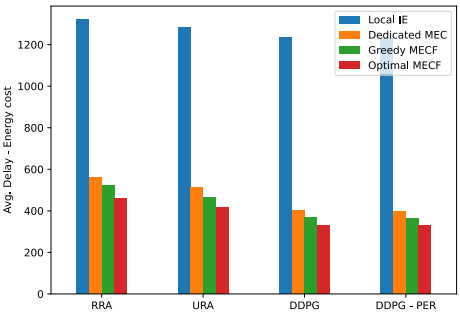
(a)



(b)



(c)



(d)

FIGURE 11. Average energy-delay cost of task execution comparison in IIoT network containing (a) 10 IEs, (b) 20 IEs, (c) 30 IEs, and (d) 40 IEs.

MEC servers, the issue of task management in an IIoT system can be resolved effectively using MEC servers. Furthermore, the Optimal MECF with an optimal policy is the best strategy for reducing the latency in all scenarios.

TABLE 7. Average energy consumption (J) of task execution comparison.

Number of IEs		10	20	30	40
RRA	Local IE	32.5	62.7	100.9	125.9
	Dedicated MEC	45.3	91.6	137.9	187.1
	Greedy MECF	42.8	81.5	131.2	173.8
	Optimal MECF	40.1	75.2	124.8	159.9
URA	Local IE	30.7	58.2	97.1	122.3
	Dedicated MEC	35.0	71.2	107.8	146.4
	Greedy MECF	31.2	66.3	104.3	135.8
	Optimal MECF	30.5	59.2	95.9	123
DDPG	Local IE	29.5	54.6	82.1	117.8
	Dedicated MEC	30.2	63.3	91.8	124.0
	Greedy MECF	27.4	58.2	86.0	109.6
	Optimal MECF	25.9	53.3	81.3	105.7
DDPG-PER	Local IE	29.3	54.7	82.0	117.5
	Dedicated MEC	30.1	63.3	91.7	123.8
	Greedy MECF	27.3	58.0	85.4	108.8
	Optimal MECF	**25.7**	**51.3**	**81.1**	**105.2**

In contrast, DDPG-PER outperforms the other methods in terms of optimizing the processing time for each task. As demonstrated in Table 6, the proposed method consistently produced decreased latency in all settings compared with RRA, URA and DDPG. For a system with 40 IEs, where tasks were handled solely in the IE, the average latencies for RRA, URA, DDPG and DDPG-PER are 18.5, 17.98, 17.30 and 17.28 s, respectively. When the local MEC server handles the processing MEC tasks, the minimal delays for resource allocation via RRA, URA, DDPG and DDPG-PER are 5.58, 5.51, 4.19 and 4.12 s, respectively. With the IIoT system combined with the Optimal MECF, the average latencies of each task with RRA and URA are 4.52 and 4.46 s, respectively, whereas those of DDPG-PER and DDPG are 3.37 and 3.36 s, respectively. This demonstrates that resource allocation using DDPG-PER minimizes the system delay most effectively.

2) PERFORMANCE COMPARISON OF ENERGY CONSUMPTION

In addition to the latency of the system, we conduct studies to determine the performance of power consumption optimization. The experimental results are presented in Table 7. Similar to the delay, energy consumption is correlated with the number of devices in the system. However, in contrast to delay optimization, the energy usage in circumstances in which the IE handled tasks by itself reached a minimal value.

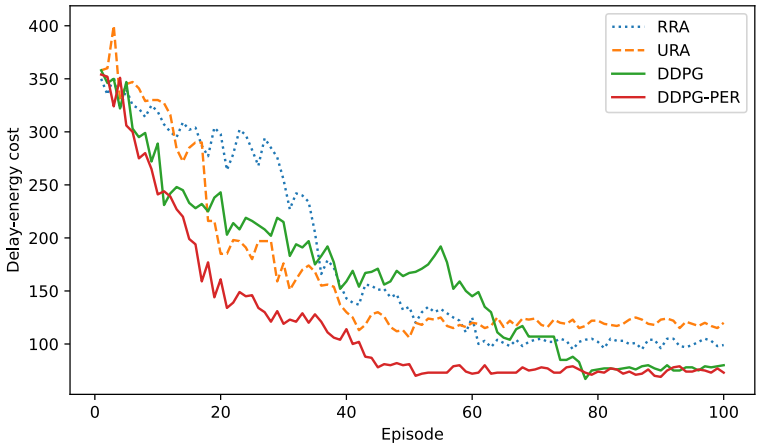


FIGURE 12. Convergence comparison.

In the example of an IIoT system with 10 IEs using RRA, the average power consumption of each task while using simple IE is 32.5 J, whereas the average power consumption of each task when using the Optimal MECF and Greedy MECF is 40.1 and 42.8 J. Dedicated MEC is the least efficient, requiring as much as 45.3 J per task. This pattern is observed as the number of IEs in the system increases. This is due to the task management of each strategy. For a system that simply processes tasks using IE, the energy consumption is only due to computation, whereas data transmission increases the energy consumption when MEC servers are involved. The greedy and optimal MECF use less energy per task than dedicated MEC. This demonstrates that delegating tasks to distant MEC servers is more energy-efficient than using only local MEC servers. However, the utilization of RRA and URA combined with the MEC federation consumes more energy than task processing alone in the IE. This demonstrates the inefficacy of these methods in the context of the MEC federation system.

According to Table 7, DDPG-PER and DDPG allocates resources more effectively than URA and RRA. In the case of the combined IIoT system with the Optimal MECF which includes 10 devices, each task requires an average of 40.1 J with RRA and 30.5 J with URA. Nevertheless, that of DDPG-PER is only 25.7 J. Moreover, unlike RRA and URA, DDPG-PER and DDPG work effectively with the MEC federation approach. With a network of 40 IIoT devices, one task processed by the Optimal MECF consumes only 105.2J on average, compared with 117.5, 123.78, and 108.8 J for one task handled exclusively by the IE, Dedicated MEC and Greedy MECF. In all experimental settings, the proposed DDPG-PER resource allocation strategy consistently outperforms the other methods in terms of energy consumption optimization.

3) PERFORMANCE COMPARISON OF ENERGY-DELAY

By optimizing both the latency and power consumption simultaneously, DDPG-PER for Optimal MECF offers the highest performance. As demonstrated in Fig. 11, the tasks handled by the Optimal MECF consistently have the lowest average energy-delay cost. The values of energy-delay cost optimization by DDPG are also good performance as DDPG-PER. This benefit is realized using the computational capacity and available resources of MEC servers across the entire network, as opposed to merely processing at the IIoT device or local MEC server. In a system with 10 IEs applying RRA, the average energy-delay cost for each task executed by only the IE is 340, compared to 144, 140 and 134 for the Dedicated MEC, Greedy MECF and Optimal MECF, respectively. Similar to RRA, the resource allocation by URA for the 10-IE system resulted in costs of 322, 120, 114 and 105 when executing tasks solely with the IE, Dedicated MEC, Greedy MECF and Optimal MECF, respectively. In addition, DDPG-PER consistently outperforms the other techniques in all experimental scenarios. For 40 IIoT devices in the MEC federation system, resource allocation via RRA and URA incurs average costs of 461 and 418, respectively, whereas DDPG and DDPG-PER cost 329. Similarly, when there are 20 IEs, the energy-delay costs for the systems using RRA, URA, DDPG, and DDPG-PER are 229, 198, 169, and 166, respectively. This deficiency of RRA and URA is due to resource allocation limitations, such as randomization with RRA and equal distribution with URA. By leveraging deep learning models to solve complicated optimization problems, DDPG and DDPG-PER can effectively allocate resources, as demonstrated by the experimental results. Thus, the proposed DDPG-PER approach for task offloading and resource allocation in an MEC federation system is an ideal solution for concurrently optimizing the latency and energy consumption of the IIoT network.

4) PERFORMANCE COMPARISON OF CONVERGENCE

In addition to evaluating the performance of each method, it is crucial to investigate convergence. The results of the comparison of the convergence rates of the strategies are shown in Fig. 12. During training, the energy-delay cost of the

RRA fluctuates significantly, indicating the instability of this model. Owing to the random resource allocation characteristics of RRA, the optimal outcomes of this approach directly correlate with the unpredictability of allocated resources. The URA is substantially more stable than the RRA. However, equal allocation of all tasks by URA does not reflect the actual state of the IIoT system. Thus, the performance of this model is insufficient. It is evident that the energy-delay cost of a system using DDPG-PER is always lower than that of systems employing other approaches. At epoch 50, the DDPG-PER model starts to converge with an energy-delay cost of 83, whereas those of the DDPG, URA, and RRA are greater than 100. After 100 episodes, all methods converge with the values 105.8, 134.8, 82.7, and 82.3 corresponding to RRA, URA, DDPG, and DDPG-PER, respectively. As shown in Fig. 12, the speed requires to obtain the optimal DDPG-PER is higher than DDPG. Hence, DDPG-PER is the most effective method in terms of convergence.

VI. CONCLUSION

In this study, we introduced an MEC federation paradigm for IIoT networks that enables IEs to offload computationally intensive tasks to MEC servers. Instead of merely enabling tasks to be handled on a local MEC server, this system enables tasks to be transferred to distant MEC servers, resulting in more effective system resource utilization. In contrast to previous research, this study incorporates all relevant practical considerations within the framework of an optimal decision-making process. This facilitates the implementation of the system in a practical context. Nonetheless, identifying the appropriate allocation of resources while lowering system latency in the context of a constantly changing state is challenging. This problem was represented as an MDP with realistic restrictive constraints. We proposed a task offloading and resource allocation framework for MEC federation in IIoT systems based on DDPG-PER. Extensive experiments showed that the federation MEC approach is more effective than simply employing an IE or a dedicated MEC server. In addition, investigations have demonstrated that DDPG-PER outperforms other resource allocation strategies by decreasing both power consumption and system delay. In the future, while optimizing IIoT system resources, we may consider the mobility of devices based on this research.

REFERENCES

[1] M. Aledhari, R. Razzak, B. Qolomany, A. Al-Fuqaha, and F. Saeed, "Biomedical IoT: Enabling technologies, architectural elements, challenges, and future directions," *IEEE Access*, vol. 10, pp. 31306–31339, 2022.

[2] N. Aung, S. Dhelim, L. Chen, A. Lakas, W. Zhang, H. Ning, S. Chaib, and M. T. Kechadi, "VeSoNet: Traffic-aware content caching for vehicular social networks using deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 8, pp. 8638–8649, Aug. 2023.

[3] J. J. Kang, W. Yang, G. Dermody, M. Ghasemian, S. Adibi, and P. Haskell-Dowland, "No soldiers left behind: An IoT-based low-power military mobile health system design," *IEEE Access*, vol. 8, pp. 201498–201515, 2020.

[4] J. Okwuibe, J. Haavisto, I. Kovacevic, E. Harjula, I. Ahmad, J. Islam, and M. Ylianttila, "SDN-enabled resource orchestration for industrial IoT in collaborative edge-cloud networks," *IEEE Access*, vol. 9, pp. 115839–115854, 2021.

[5] Y. Yang, Y. Miao, Z. Ying, J. Ning, X. Meng, and K. R. Choo, "Privacy-preserving threshold spatial keyword search in cloud-assisted IIoT," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 16990–17001, Sep. 2022.

[6] S. Qi, W. Wei, J. Cheng, Y. Zheng, Z. Su, J. Zhang, and Y. Qi, "Secure and efficient item traceability for cloud-aided IIoT," *ACM Trans. Sensor Netw.*, vol. 18, no. 4, pp. 1–24, Nov. 2022.

[7] F. Fu, Z. Zhang, F. R. Yu, and Q. Yan, "An actor-critic reinforcement learning-based resource management in mobile edge computing systems," *Int. J. Mach. Learn. Cybern.*, vol. 11, no. 8, pp. 1875–1889, Aug. 2020.

[8] S. Bebortta, D. Senapati, C. R. Panigrahi, and B. Pati, "Adaptive performance modeling framework for QoS-aware offloading in MEC-based IIoT systems," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 10162–10171, Jun. 2022.

[9] A. Naouri, H. Wu, N. A. Nouri, S. Dhelim, and H. Ning, "A novel framework for mobile-edge computing by optimizing task offloading," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 13065–13076, Aug. 2021.

[10] Z. Jin, C. Zhang, Y. Jin, L. Zhang, and J. Su, "A resource allocation scheme for joint optimizing energy consumption and delay in collaborative edge computing-based industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 6236–6243, Sep. 2022.

[11] C. Tang, C. Zhu, N. Zhang, M. Guizani, and J. J. P. C. Rodrigues, "SDN-assisted mobile edge computing for collaborative computation offloading in industrial Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 24253–24263, Dec. 2022.

[12] X. Dai, Z. Xiao, H. Jiang, M. Alazab, J. C. S. Lui, S. Dustdar, and J. Liu, "Task co-offloading for D2D-assisted mobile edge computing in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 480–490, Jan. 2023.

[13] D. Wei, N. Xi, X. Ma, M. Shojafar, S. Kumari, and J. Ma, "Personalized privacy-aware task offloading for edge-cloud-assisted industrial Internet of Things in automated manufacturing," *IEEE Trans. Ind. Informat.*, vol. 18, no. 11, pp. 7935–7945, Nov. 2022.

[14] W. Zhang, W. Guo, X. Liu, Y. Liu, J. Zhou, B. Li, Q. Lu, and S. Yang, "LSTM-based analysis of industrial IoT equipment," *IEEE Access*, vol. 6, pp. 23551–23560, 2018.

[15] L. Zhao, I. B. M. Matsuo, Y. Zhou, and W.-J. Lee, "Design of an industrial IoT-based monitoring system for power substations," *IEEE Trans. Ind. Appl.*, vol. 55, no. 6, pp. 5666–5674, Nov. 2019.

[16] X. Deng, J. Yin, P. Guan, N. N. Xiong, L. Zhang, and S. Mumtaz, "Intelligent delay-aware partial computing task offloading for multiuser industrial Internet of Things through edge computing," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 2954–2966, Feb. 2023.

[17] 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.

[18] X. Liu, J. Yu, J. Wang, and Y. Gao, "Resource allocation with edge computing in IoT networks via machine learning," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3415–3426, Apr. 2020.

[19] L. Yang, M. Li, P. Si, R. Yang, E. Sun, and Y. Zhang, "Energy-efficient resource allocation for blockchain-enabled industrial Internet of Things with deep reinforcement learning," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2318–2329, Feb. 2021.

[20] L. Sun, L. Wan, and X. Wang, "Learning-based resource allocation strategy for industrial IoT in UAV-enabled MEC systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 5031–5040, Jul. 2021.

[21] Y. He, Y. Ren, Z. Zhou, S. Mumtaz, S. Al-Rubaye, A. Tsourdos, and O. A. Dobre, "Two-timescale resource allocation for automated networks in IIoT," *IEEE Trans. Wireless Commun.*, vol. 21, no. 10, pp. 7881–7896, Oct. 2022.

[22] H. Zhou, Z. Wang, G. Min, and H. Zhang, "UAV-aided computation offloading in mobile-edge computing networks: A Stackelberg game approach," *IEEE Internet Things J.*, vol. 10, no. 8, pp. 6622–6633, Apr. 2023.

[23] Y. Chen, Z. Liu, Y. Zhang, Y. Wu, X. Chen, and L. Zhao, "Deep reinforcement learning-based dynamic resource management for mobile edge computing in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4925–4934, Jul. 2021.

[24] H. Zhou, Z. Wang, H. Zheng, S. He, and M. Dong, "Cost minimization-oriented computation offloading and service caching in mobile cloud-edge computing: An A3C-based approach," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 3, pp. 1326–1338, May 2023.
[25] Y. Hou, L. Liu, Q. Wei, X. Xu, and C. Chen, "A novel DDPG method with prioritized experience replay," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2017, pp. 316–321.
[26] A. R. Mahmood, H. P. van Hasselt, and R. S. Sutton, "Weighted importance sampling for off-policy learning with linear function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1–9.
[27] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.
[28] H.-J. Jeong, H.-J. Lee, C. H. Shin, and S.-M. Moon, "IONN: Incremental offloading of neural network computations from mobile devices to edge servers," in *Proc. ACM Symp. Cloud Comput.*, Oct. 2018, pp. 401–411.
[29] D. Bertsekas and R. Gallager, *Data Networks*. Belmont, MA, USA: Athena Scientific, 2021.
[30] S. Jain and J. M. Smith, "Open finite queueing networks with M/M/C/K parallel servers," *Comput. Oper. Res.*, vol. 21, no. 3, pp. 297–317, Mar. 1994.
[31] A. Sarwar, "CMOS power consumption and C_{pd} calculation," in *Proc. Design Considerations Log. Products*, 1997, pp. 1–16.
[32] N. Zhao, C. Roberts, S. Hillmansen, and G. Nicholson, "A multiple train trajectory optimization to minimize energy consumption and delay," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 5, pp. 2363–2372, Oct. 2015.
[33] F. Meng, P. Chen, L. Wu, and J. Cheng, "Power allocation in multi-user cellular networks: Deep reinforcement learning approaches," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6255–6267, Oct. 2020.
[34] O. Karatalay, I. Psaromiligkos, and B. Champagne, "Energy-efficient D2D-aided fog computing under probabilistic time constraints," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2021, pp. 1–7.
[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[36] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 387–395.
[37] D. Mehta, "State-of-the-art reinforcement learning algorithms," *Int. J. Eng. Res.*, vol. V8, no. 12, pp. 717–722, Jan. 2020.
[38] T. Sharma, A. Chehri, and P. Fortier, "Review of optical and wireless backhaul networks and emerging trends of next generation 5G and 6G technologies," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 3, Mar. 2021, Art. no. e4155.
[39] C. Li, C. Qianqian, and Y. Luo, "Low-latency edge cooperation caching based on base station cooperation in SDN based MEC," *Exp. Syst. Appl.*, vol. 191, Apr. 2022, Art. no. 116252.
[40] S. Yang, "A joint optimization scheme for task offloading and resource allocation based on edge computing in 5G communication networks," *Comput. Commun.*, vol. 160, pp. 759–768, Jul. 2020.
[41] S. S. Alotaibi, "Ensemble technique with optimal feature selection for Saudi stock market prediction: A novel hybrid red deer-grey algorithm," *IEEE Access*, vol. 9, pp. 64929–64944, 2021.

HUONG MAI DO received the B.S. degree in electronic and telecommunication from the Hanoi University of Science and Technology, Hanoi, Vietnam, in 2020. She is currently pursuing the master's degree with Soongsil University. Her research interest includes mobile edge computing.

TUAN PHONG TRAN received the B.S. degree in information technology from Le Quy Don Technical University, Vietnam, in 2019. He is currently pursuing the M.S. degree in information communication convergence technology with Soongsil University, South Korea. His current research interests include machine learning, deep learning, and edge computing.

MYUNGSIK YOO received the B.S. and M.S. degrees in electrical engineering from Korea University, Seoul, South Korea, in 1989 and 1991, respectively, and the Ph.D. degree in electrical engineering from The State University of New York at Buffalo, New York, in 2000. He was a Senior Research Engineer with the Nokia Research Center, Burlington, MA, USA. He is currently a full-time Professor with the School of Electronic Engineering, Soongsil University, Seoul. His research interests include visible-light communications, cloud computing, and machine learning.