

RESEARCH ARTICLE

An Efficient Multi-Edge Server Coalition Computation Offloading Scheme of Sensor-Edge-Cloud

DING YIN^{1,2}, (Member, IEEE), LI CHEN^{1,2,3,4}, JUN YANG^{2,3,4}, AND KUN DENG^{1,2,3,4}¹School of Computer Science and Technology (School of Artificial Intelligence), Zhejiang Sci-Tech University, Hangzhou 310018, China²College of Information Science and Engineering, Jiaying University, Jiaying 314001, China³Key Laboratory of Medical Electronics and Digital Health of Zhejiang Province, Jiaying University, Jiaying 31400, China⁴Engineering Research Center of Intelligent Human Health Situation Awareness of Zhejiang Province, Jiaying University, Jiaying 314001, China

Corresponding author: Li Chen (chenli20040415@163.com)

This work was supported in part by the Humanity and Social Science Research (College Counselors) Project of the Ministry of Education of China under Grant 22JDSZ3023, in part by the Natural Science Foundation of Zhejiang Province under Grant LGG22F020021, and in part by the Top-Level Talent Project of Zhejiang Province.

ABSTRACT The high latency and high energy consumption of wireless body areas networks (WBANs) for computing-intensive tasks is intolerable, especially for remote interventional surgery. In this paper, a multi-mobile edge server collaborative computation offloading scheme is proposed, which enables tasks to choose a server and offload a certain proportion of computation to efficiently handle computing-intensive services for massive users. More specifically, we formulate the problem of minimizing system latency and energy consumption, and then model the task offloading and resource allocation process as a Markov decision process (MDP). We have developed a scheme called m4m-PDQN to optimize offloading decisions, aiming to minimize the weighted sum of latency and energy consumption. Compared to existing single-server offloading schemes, it is more effective in utilizing computing resources and reducing waiting time and energy consumption for computing tasks in the multiple-server collaborative computing scenarios. The experimental results show that it outperforms other algorithms in terms of performance and efficiency, significantly improving the quality of service (QoS) for wearable wireless body area networks for medical applications.

INDEX TERMS Multi-access edge computing (MEC), reinforcement learning, telemedicine, wireless body area networks.

I. INTRODUCTION


With the development of artificial intelligence, advanced robotics, and cloud computing technology in recent years, wireless body area networks (WBANs) have emerged as an important way to achieve intelligent and remote medical care [1].

WBAN is a wireless communication network designed to connect sensor devices worn on the human body to facilitate personal health monitoring and medical services. It establishes a communication environment in close proximity to the human body, utilizing wireless technology and sensor devices to monitor and collect physiological parameters and

motion data relevant to the body [1]. However, due to energy limitation, limited bandwidth, and narrow coverage range, the critical quality of service (QoS) requirements of WBANs cannot be effectively guaranteed in the resource-constrained environment of health monitoring. These limitations have become a bottleneck limiting WBANs' application [2], [3].

To address the low QoS issue in WBANs, a new computing model, mobile edge computing (MEC), has emerged in recent years. MEC provides computing services on network edge devices and moves computing tasks from sensor devices to remote resources to achieve computing resource sharing and network optimization [4].

Task offloading refers to the process of transferring computational tasks from mobile devices to more powerful computing resources for processing, aiming to enhance

The associate editor coordinating the review of this manuscript and approving it for publication was Nitin Gupta .

performance and save energy consumption on the mobile devices. By offloading computational tasks to MEC nodes, mobile devices can shift the heavy computational workload to the network edge, alleviating the computing burden on the devices themselves, reducing latency and energy consumption, and improving user experience [5], [6], [7].

In recent years, an increasing amount of research has focused on improving the QoS of WBAN networks, with the use of MEC task offloading technology being a widely explored solution [8], [9], [10], [11], [12], [13], [14], [15]. Specifically, joint cost and energy-efficient task offloading in health care systems enabled by MEC are investigated in [8], with incentive measures designed for WBAN users to reduce their task offloading, significantly reducing the energy cost of edge servers. A resource management scheme was proposed to minimize the energy consumption of edge servers without affecting the WBAN user's quality of experience (QoE) in [9]. Reference [10] presents a task offloading framework that considers WBAN user task priorities, multiple tasks, resources, availability, different delays for computing offloading, network connectivity, and processing devices to improve end-to-end latency. In [11], a new control scheme is designed to effectively share the limited computing and communication resources in the MEC-assisted WBAN (M-W) platform, achieving substantial performance improvement compared to the comparison scheme. Reference [12] proposes a task offloading framework that combines cellular, WiFi networks, and device-to-device communication to fully utilize the communication and computing resources in the WBAN scenario, enhancing the system's reliability. In [13], the aim is to enhance UE's computing capability by utilizing small Coordinator-based Mobile Edge Computing (C-MEC) servers, allowing UE to effectively execute delay-sensitive or computation-intensive tasks. In [14], the authors propose a medical monitoring framework based on hospital/home and a task offloading mechanism supported by wireless relays composed of a network model and a computation model, with the performance evaluation of various metrics for all relays in different scenarios. In [15], a two-stage potential game-based task offloading strategy (TPOS) that considers both WBAN task priorities and user priorities to optimize resource allocation is proposed.

Reinforcement learning is a machine learning technique that optimizes behaviour by trial and error learning. In MEC, reinforcement learning can be applied to optimize task offloading strategies by learning task allocation policies through interaction with the environment to maximize system performance. The reinforcement learning environment in MEC includes mobile devices, edge servers, and communication links. Mobile devices can act as agents and take different actions to choose the server for offloading tasks or to execute local computing. Edge servers can be part of the environment, provide offloading services and send feedback information to mobile devices. Communication links can also be part of the environment, affecting the efficiency and latency of task allocation.

In this paper, we propose a resource-constrained, edge o

In this paper, we propose a resource-constrained, edge offloading model, as shown in Figure 1, which consists of a remote medical center (cloud center), multiple edge cloud servers, and multiple user devices (UEs) in WBAN. UEs can split computing tasks into two parts and offload a part to an edge server, while the remaining portion is locally executed. In addition, to better simulate real-world scenarios, the computing tasks and available computing capabilities of UE are time-varying. Unlike existing DRL policies based on continuous and discrete space decisions, this paper proposes an algorithm based on PDQN, named m4m-PDQN, for task offloading and resource allocation in a mixed action space. The main contributions of this study are presented as follows:

- This study addresses the problem of limited computing resources in a single-edge cloud, which cannot meet the low-latency and low-energy requirements of multiple UE devices responding to intensive computational tasks. To improve the QoS of WBAN, a resource-constrained, multi-to-multi, edge collaborative computing scenario was designed with the goal of minimizing the weighted total cost of system latency and energy consumption.
- The joint optimization problem of minimizing the weighted total cost of latency and energy consumption was modelled as a Markov decision process (MDP). The m4m-PDQN algorithm based on a mixed action space was employed to enable the entire system to learn effective task offloading and resource allocation strategies, maximizing the improvement of users' quality of experience (QoE).
- A simulation experiment was conducted to evaluate the overall performance of the policy learned by the m4m-PDQN algorithm using numerical metrics for user latency and energy consumption. The results also demonstrated the superiority of the m4m-PDQN algorithm based on a mixed action space compared to the Deterministic Policy Gradient (DPG) algorithm based on a continuous action space combined with an experience replay technique, as well as other policies.

The remainder of this paper is organized as follows: Section II provides an overview of related work. The system model is described in Section III, while Section IV presents a novel deep reinforcement learning algorithm named m4m-PDQN. Simulation experiments are conducted in Section V to compare the performance of different strategies. Section VI summarizes the contributions of this paper.

II. RELATED WORK

Numerous prior works have investigated task offloading and resource allocation in Mobile Edge Computing (MEC) using the reinforcement learning method from multiple optimization perspectives.

Delay-sensitive applications require system response times that are optimized to reduce time consumption. To address this, delay-based offloading schemes have been proposed to optimize system response time for delay-sensitive

applications [16], [17], [18], [19]. One approach is to use autonomous partial offloading systems with RL-based offloading policies to optimize delay performance [16]. Another proposal is a Software-Defined Edge Cloud (SDEC) based on reinforcement learning that uses Q-learning and cooperative Q-learning reinforcement learning schemes to optimize task offloading and resource allocation in wireless MEC and reduce total delay [17]. Additionally, a computation offloading algorithm for a UAV-assisted mobile edge computing system has been developed using Deep Deterministic Policy Gradient (DDPG) in reinforcement learning to minimize maximum processing delay while jointly optimizing user scheduling, task offloading rate, drone flight angles, and flight speed [18]. And a distributed computation offloading method is proposed for utilizing surrounding vehicles as a resource pool in scenarios where MEC is not available or sufficient. The proposed method involves splitting complex tasks into smaller sub-tasks and assigning them for optimal execution time using a Deep Q-learning Network [19].

Meanwhile, other researchers have considered both energy and delay factors to improve energy efficiency or reduce costs [20], [21], [22], [23]. One approach is an RL-based computation offloading and energy transfer algorithm that uses joint optimization methods to develop an allocation algorithm that obtains an approximately optimal solution for energy and computation resource allocation [20]. Another method models devices as job shops and uses the Q-learning algorithm to determine the optimal offloading strategy in an MEC environment with Device to Device (D2D) communication, minimizing energy consumption and delay [21]. An RL-based method that uses the Actor-Critic (AC) algorithm has been proposed to address the offloading decision and resource allocation problems in MEC systems with multidimensional continuous and discrete action spaces [22]. An intelligent factory model is constructed, and a mixed-integer nonlinear programming problem is formulated with the objective of minimizing the weighted sum of task delay and energy consumption. Since the problem is NP-hard, the objective function is solved using a deep Q-network (DQN) approach [23]. A computation offloading strategy is proposed for an MEC system consisting of multiple mobile users, considering stochastic task arrivals and wireless channels. The strategy is designed to minimize the long-term average computation cost, including power consumption and buffering delay [24].

In addition, a framework with multiple static and vehicle-assisted edge servers has been designed, and an improved computation offloading method based on deep reinforcement learning has been proposed to minimize the weighted total cost, including transmission and execution cost, energy consumption cost, and communication bandwidth cost [25]. A reinforcement learning method based on Q-Learning and Double Deep Q-Network (DDQN) has been proposed to solve the joint optimization problem of computation offloading and resource allocation in a dynamic multi-user MEC system, with simulation results showing

significant reductions in system energy consumption in different scenarios [26]. A content caching strategy based on DQN is proposed, which considers caching benefit, transmission delay, and backhaul link load. Additionally, a quantum ant colony-based computing offloading strategy is also presented, which takes into account delay, energy consumption, and server cost [27]. A qualified trace actor-critic (AC) algorithm was proposed to improve the revenue of mobile network operators by maximizing the number of offloaded tasks while reducing energy consumption and latency [22]. An online algorithm framework called DROO for wireless-powered MEC networks was proposed, which employs deep reinforcement learning to make binary offloading decisions that adapt to time-varying wireless channel conditions [28].

III. SYSTEM MODEL

As shown in Figure 1, there are multiple edge servers and multiple users in the environment, and the computing resources of edge servers are limited. User equipment (UE) can offload tasks to an edge server for processing in any proportion while also performing local processing on UE and remote processing on edge servers to maximize system service quality. To achieve a more realistic scenario, the local computing capability of UE and the computational tasks to be processed are time-varying. This paper assumes that UE offloads tasks to edge servers via wireless channels, and different servers transmit data to the cloud centre by different feedback paths.

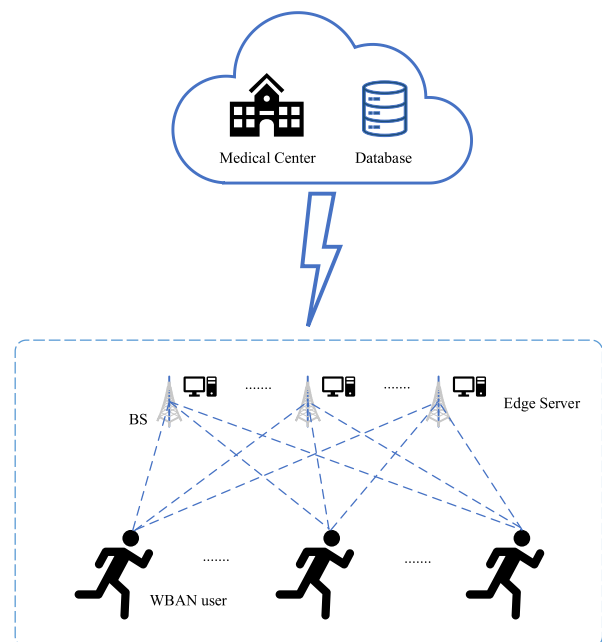


FIGURE 1. The system diagram proposed consists of multiple users, multiple edge servers, and one medical center.

Specifically, $E = \{e_1, e_2, \dots, e_n, \dots, e_N\}$ and $U = \{u_1, u_2, \dots, u_m, \dots, u_M\}$ represent the set

of edge servers and set of users, respectively, and $S = \{s_1, s_2, \dots, s_m, \dots, s_M\}$ represents a set of computationally intensive tasks at the user end, where $s_m = \{d_m, c_m, \tau\}$ represents the data size, number of clock cycles required to process one bit, and maximum delay tolerance. The remaining computing resources of e_n are F_n , and the size of computing resources allocated to the task is $f_{n,m}$. When the remaining computing resources are insufficient, e_n cannot continue to provide services. The task processing includes the following three stages:

- **Task offloading:** In this stage, some parts of the user task are offloaded to the edge cloud. Let $\rho_{m,n}$ represent the offloading ratio of task s_m from u_m to edge server e_n , representing the part that e_n needs to compute. Based on $\rho_{m,n}$, users can flexibly offload part of s_m to edge servers and locally process the other part to fully utilize the computing resources of UE.
- **Task computation:** After collecting the subtasks in the first stage, edge servers and UE compute the received subtasks in parallel, which helps reduce computation delay.
- **Result feedback:** After the task computation is completed, the edge server transmits the computation result to the cloud centre via a wireless channel and then returns it to UE via a downstream link. Since the number of bits in the task processing result is much smaller than that of the initial data, the delay in the result feedback can be disregarded.

A. WIRELESS COMMUNICATION MODEL

When tasks are offloaded by the UE to edge servers via wireless channels, additional time and energy consumption are incurred. In our model, g_m represents the channel gain between a UE device and a wireless base station, p_m is the transmission power used to offload tasks to the base station, W is the bandwidth of the channel between UE and the base station, and N_0 is the noise power spectral density of the channel. The transmission rate of the channel when tasks are offloaded to edge servers is defined as:

$$R_m = W \log_2 \left(1 + \frac{p_m g_m}{N_0 W} \right) \quad (1)$$

The delay and energy consumption during task transmission are defined as:

$$T_m^{tr} = \frac{\rho_{m,n} d_m}{R_m} \quad (2)$$

$$E_m^{tr} = p_m T_m^{tr} = \frac{p_m \rho_{m,n} d_m}{R_m} \quad (3)$$

where $\rho_{m,n} \in [0, 1]$ represents the task offloading ratio of u_m to edge server e_n .

B. LOCAL COMPUTING MODEL

Although the computing capability of UE is quite limited, to fully utilize the existing computing resources and improve the user experience, a portion of the task will be locally

processed. Assuming that the computing capacity of u_m is a fixed value per second over a certain period, denoted as f_m^l , and k represents the energy factor of the CPU chip, the latency and energy consumption of local processing of the task are defined as follows:

$$T_m^l = \frac{(1 - \rho_{m,n}) d_m c_m}{f_m^l} \quad (4)$$

$$E_m^l = (1 - \rho_{m,n}) \left(f_m^l \right)^2 k d_m c_m \quad (5)$$

C. EDGE COMPUTING MODEL

In our model, UE has multiple options for edge offloading targets, but the final offloading target is one of the multiple edge servers. Compared to local computing, edge computing has richer computing resources, which can greatly reduce the computing latency of intensive computing applications.

We define the delay and energy consumption of edge processing as follows:

$$T_m^e = \frac{\rho_{m,n} d_m c_m}{f_{n,m}} \quad (6)$$

$$E_m^e = p_n T_m^e = \frac{p_n \rho_{m,n} d_m c_m}{f_{n,m}} \quad (7)$$

where $f_{n,m}$ is the computing resources assigned to u_m by edge server e_n and p_n is the computing power of e_n .

In addition, the remaining computing resources of edge server e_n are defined as F_n . Therefore, the computing resources allocated to the task by e_n cannot exceed its total limit, that is, $\sum_m^M f_{n,m} \leq F_n$. When the total allocated resources exceed the edge server limit, the system will forcefully offload the task to UE for local computing.

D. PROBLEM MODEL

In this paper, we allow task offloading in a certain proportion to maximize the utilization of the system's computational resources and to reduce the required latency for processing-intensive computing tasks. Furthermore, local and edge computing can be performed in parallel. Therefore, the delay and energy consumption at the mobile device are defined as:

$$T_m = \max \left\{ T_m^{tr} + T_m^e, T_m^l \right\} \quad (8)$$

$$E_m = E_m^{tr} + E_m^e + E_m^l \quad (9)$$

The comprehensive weighted total cost is:

$$C_m = \alpha T_m + (1 - \alpha) E_m \quad (10)$$

where $\alpha \in [0, 1]$ is the weight factor between delay and energy consumption.

The primary optimization objective of this paper is to minimize the total cost of the system while meeting the user's requirements for task processing delay tolerance. The total cost includes time and energy costs. The goal is to minimize the cost of the system by effectively utilizing edge computing resources while meeting user requirements, thus improving

system performance and user experience. The total cost of the system is defined as:

$$C_{total} = \sum_{m=1}^M C_m \quad (11)$$

To ensure successful task execution, two conditions must be met: First, the task processing delay must not exceed the maximum delay tolerance τ . Second, the computational resources allocated by the edge server to the UE must not exceed its remaining computational resources. Assuming that the number of tasks successfully executed by the UE is y , the task execution success rate can be defined as the ratio of the number of tasks successfully executed by the UE to the total number of tasks:

$$SUL = \frac{y}{M} \quad (12)$$

Under the constraints of task processing delay and edge server resource limitations, we need to develop a task offloading decision and resource allocation plan to maximize the task execution success rate and minimize the total cost of all mobile devices. Therefore, the problem is formulated as follows:

$$\min C_{total} \text{ and } \max SUL \quad (13a)$$

$$\text{s.t. } \rho_{m,n} \in [0, 1] \quad (13b)$$

$$0 \leq T_m \leq \tau \quad (13c)$$

$$0 \leq f_{n,m} \leq F_n \quad (13d)$$

$$\sum_{m=1}^M f_{n,m} \leq F_n \quad (13e)$$

In the next section, we develop a DRL-based method to optimize this problem.

IV. DRL-BASED COLLABORATIVE OPTIMIZATION ALGORITHM

In this section, we propose a DRL-based method for minimizing the system delay and energy cost. Specifically, we use the proposed m4m-PDQN method to learn a task offloading and computing resource allocation strategy from the perspective of the entire system. The strategy selects an action that involves offloading a portion of the UE-intensive computing tasks and allocating the limited computing resources of the edge servers, with the aim of reducing the overall system delay and energy cost. Our discussion is presented in two parts: 1) MDP model formulation and 2) m4m-PDQN solution.

A. MDP MODEL FORMULATION

The Markov decision process (MDP) is a mathematical framework commonly used in reinforcement learning to describe sequential decision-making problems. The MDP involves the interaction between an agent and an environment, where the agent chooses an action at each time slope to influence the environment and receives a reward while

transitioning to a new state. By solving the MDP, the optimal policy is obtained, which selects the best action at each state to maximize the long-term cumulative reward. The optimal policy is represented using value functions or Q-functions, where the former represents the long-term cumulative reward from a given state following the optimal policy, and the latter represents the long-term cumulative reward from taking a specific action in a given state following the optimal policy. The MDP formulation for the optimization problem proposed in this paper is described as follows:

1) DECISION EPOCHS

The period during which a UE device makes a decision and the timing of decisions made by all UE devices are represented by a sequence T :

$$T = \{1, 2, \dots, t, \dots, M\} \quad (14)$$

where M represents the total number of UE devices and a specific decision time slot is described as t , $t \in T$.

2) STATES

States: In the MDP, the state set is a collection of all possible states that describe the environment. The state of the environment affects the decisions made by an intelligent agent. In this context, a state can be represented as:

$$s_t = (d_t, RC_t, f_t^l) \quad (15)$$

where d_t represents the data size of the current UE device's intensive computing task, RC_t is the set of remaining computing resources of all edge servers at time slot t , and f_t^l represents the current UE device's local computing capability. Additionally, RC_t is expressed as

$$RC_t = (rc_t^1, rc_t^2, \dots, rc_t^n, \dots, rc_t^N) \quad (16a)$$

$$rc_t^n = rc_{t-1}^n - f_{n,t-1} \quad (16b)$$

where rc_t^n represents the remaining computing resources of edge server e_n at time slot t and $f_{n,t}$ denotes the computing resources allocated to the current UE device by edge server e_n at time slot t .

3) ACTION

At each time slot, the agent selects an action to execute based on the current state and available actions. The agent's goal is to maximize the long-term reward by choosing the optimal sequence of actions. In our model, we define the action space as follows:

$$A = \left\{ A_t = (a_t, x_t) \mid 1 \leq a_t \leq N, x_t = (x_t^1, x_t^2) \right\} \quad (17a)$$

$$\text{s.t. } a_t \in Z \quad (17b)$$

$$x_t^i \in [-1, 1], i = 1, 2 \quad (17c)$$

where a_t represents the edge server to which the current UE device offloads its task, x_t^1 represents the task offloading ratio of the current UE device, and x_t^2 represents the computing

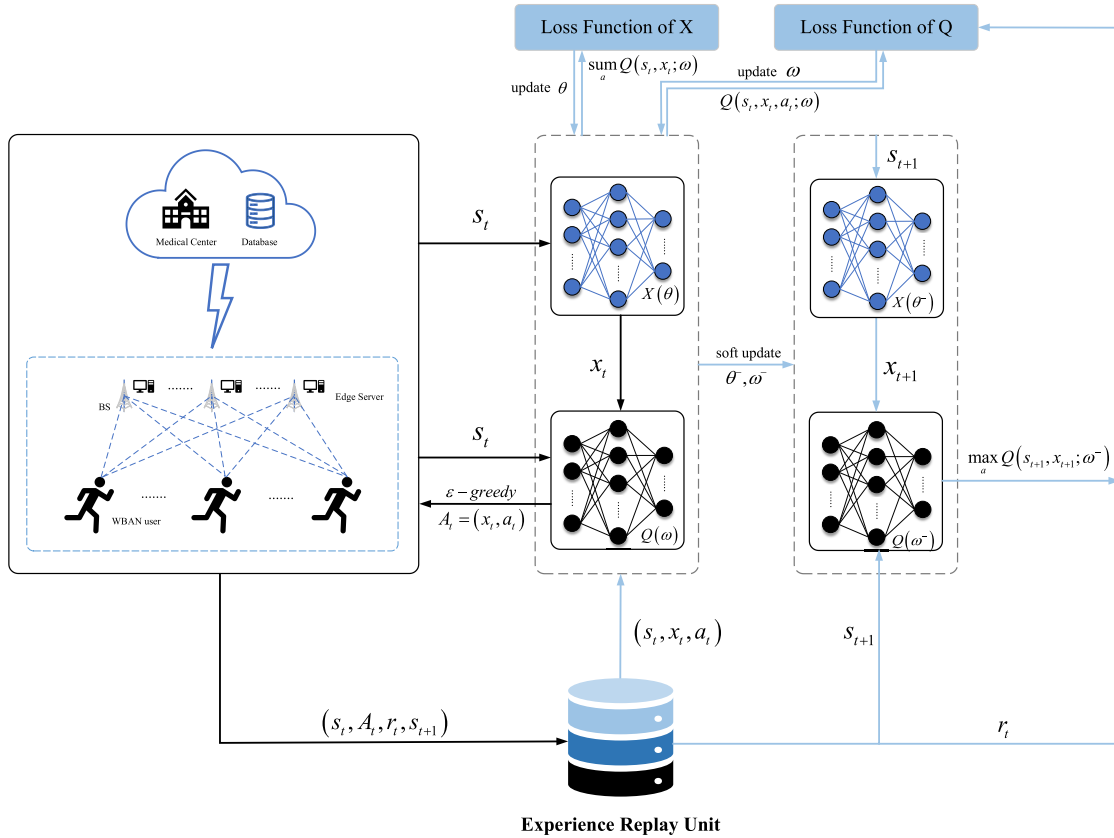


FIGURE 2. Framework of the m4m-PDQN optimization strategy.

resources allocated to the task by the edge server. It can be seen that A is a hybrid action space that simultaneously includes both discrete actions and continuous actions. In addition, N refers to the number of edge servers, and -1 and 1 correspond to the minimum value and maximum value, respectively, of the allocated computational resources and the task offloading ratio.

4) REWARD FUNCTION

In the MDP, the agent receives an immediate reward signal after taking an action, which is utilized to evaluate the quality of the action. In this paper's model, the immediate reward for the agent is composed of the time and energy costs incurred by the offloading decision. Thus, at each time slope t , the reward function for the agent is represented as:

$$r_t = \begin{cases} 1 - \frac{C_t}{C_{max}}, & \text{If the task is successfully executed} \\ -1, & \text{otherwise} \end{cases} \quad (18)$$

where C_{max} represents the maximum weighted total cost of the UE. From the above equation, when the actual total cost C_t exceeds the maximum weighted total cost, the reward is negative, and the smaller C_t is, the greater the reward, and vice versa. In addition, if the task execution fails, i.e., the edge

server exceeds its resource limit, or the task execution delay exceeds the delay tolerance, the reward value is -1 .

B. m4m-PDQN SOLUTION

The Parametrized Deep Q-Network (PDQN) is a reinforcement learning algorithm that combines ideas from the Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG) and can be utilized to solve problems with hybrid action spaces. The PDQN uses a parameterized neural network to represent the Q-function and employs techniques such as experience replay and target networks to improve learning efficiency and stability. Although the network structure of the PDQN resembles that of the DDPG, both using two neural networks and jointly inputting state and action into the second network, the PDQN does not have an actor-critic structure. Instead, the PDQN is divided into a continuous action Q-network and discrete action Q-network.

We apply the PDQN algorithm to task offloading and resource allocation in the multiserver collaboration scenario and refer to it as the m4m-PDQN algorithm. The proposed m4m-PDQN algorithm is shown in Figure 2. First, the current state is input into the X network, which outputs all continuous parameters x in the action space after passing through some fully connected layers and activation functions. Second, the current state, with the action parameter vector output by

the X network in the previous step, are input into the Q network, which passes through several fully connected layers and activation functions to output $Q(s, x, a)$, where a Q-value is produced for each of the N discrete actions in the action space, similar to the DQN. The optimal discrete action is then selected, and the corresponding continuous parameters are chosen accordingly. In addition, the loss function is divided into two parts:

- The discrete Q-network uses a TD error similar to the DQN to optimize, greedily selecting the action and Q-value with the maximum value among N Q-values and then calculating the TD loss for optimization.
- The design of the continuous network in the PDQN is aimed at deterministically providing the optimal continuous parameter values, while the discrete network can act as a “critic” by obtaining N corresponding Q-values by feeding the continuous parameters into the discrete network and then optimizing the continuous Q-network by maximizing the sum of all Q-values.

According to [29], consider an MDP with action space, as shown in (17), where $a \in [N]$ and $x \in \chi$. Let a_t denote the discrete action selected at time t , with associated continuous action parameter x_t . The Bellman equation is written as:

$$Q(s_t, a_t, x_t) = \mathbb{E}_{r_t, s_{t+1}} \left[r_t + \gamma \max_{a \in [N]} \sup_{x \in \chi} Q(s_{t+1}, a, x) \mid s_t = s, A_t = (a_t, x_t) \right] \quad (19)$$

For the right-hand side of the equation, we compute $x^* = \operatorname{argsup}_{x \in \chi} Q(s_{t+1}, a, x)$ for $a \in [N]$ and then choose the maximum Q-value, i.e., $Q(s_{t+1}, a, x^*)$.

Note that for any $s \in S$ and $x \in \chi$, when the Q-function is fixed, we view $\operatorname{argsup}_{x \in \chi} Q(s_{t+1}, a, x)$ as a function $x^Q : S \rightarrow \chi$. Thus, the Bellman equation (19) is rewritten as:

$$Q(s_t, a_t, x_t) = \mathbb{E}_{r_t, s_{t+1}} \left[r_t + \gamma \max_{a \in [N]} Q(s_{t+1}, a, x^Q(s_{t+1})) \mid s_t = s \right] \quad (20)$$

Similar to the DQN approach, we approximate $Q(s, a, x)$ with a deep neural network $Q(s, a, x; \omega)$ and use a deterministic policy network $x(s; \theta) : S \rightarrow \chi$ to approximate x^Q . Given a fixed ω , we obtain a set of θ that satisfies:

$$Q(s, a, x(s; \theta); \omega) \approx \sup_{x \in \chi} Q(s, a, x; \omega) \text{ for each } a \in [N] \quad (21)$$

Let ω_t and θ_t denote the parameters of the value network and deterministic policy network at time t , respectively. For a fixed $n \geq 1$, we define the n -step target y_t as:

$$y_t = \sum_{i=1}^{n-1} \gamma^i r_{t+i} + \gamma^n \max_{a \in [N]} Q(s_{t+n}, a, x(s_{t+n}; \theta_t); \omega_t) \quad (22)$$

We then define two loss functions:

$$L_t^Q(\omega) = \frac{1}{2} [Q(s_t, a_t, x_t; \omega) - y_t]^2 \quad (23a)$$

$$L_t^X(\omega) = - \sum_{a=1}^N Q(s_t, a, x(s_t; \theta); \omega_t) \quad (23b)$$

to optimize the parameters of the two networks.

Algorithm 1 The m4m-PDQN Optimization Algorithm

- 1: Initialize experience replay unit R
 - 2: Randomly initialize θ and ω , let $\theta^- = \theta, \omega^- = \omega$
 - 3: Loop for each episode:
 - 4: Reset system model environment, and initialize $s_1 \in S$
 - 5: Loop for each time slot $t \in [1, M]$:
 - 6: x_t is obtained by inputting s_t into $X(\theta)$
 - 7: Calculate $Q(s_t, x_t; \omega)$, let $a_t = \arg \max_a Q(s_t, x_t; \omega)$
 - 8: Carry out action $A_t = (x_t, a_t)$, and observe the immediate reward r_t and s_{t+1} , then R is updated with (s_t, A_t, r_t, s_{t+1})
 - 9: Randomly sample minibatch from R
 - 10: Set $y_t = \begin{cases} r_t & , t = M \\ r_t + \gamma \max_a Q(s_{t+1}, X(s_{t+1}; \theta^-); \omega^-) & , \text{ otherwise} \end{cases}$
 - 11: Use the loss functions to train $Q(\omega)$ and $X(\theta)$, then update θ and ω
 - 12: Update the target networks by $\theta^- \leftarrow \delta\theta + (1 - \delta)\theta$ and $\omega^- \leftarrow \delta\omega + (1 - \delta)\omega$
 - 13: Let $s_t = s_{t+1}$
 - 14: end for
 - 15: end for
-

V. SIMULATION RESULTS

In this section, we present the numerical results from simulation experiments carried out to investigate the performance of our proposed m4m-PDQN algorithm. First, we introduce the parameter settings of the simulation system. Second, we demonstrate the performance of the proposed algorithm compared to other solutions from several different perspectives.

A. NUMERICAL SETUP

The m4m-PDQN architecture consists of two neural networks, including a Q network and an X network. The Q network is a fully connected neural network with two hidden layers of 256 and 128 neurons and an output layer of dimension N , using the ReLU activation function. The X network has the same hidden layer structure as the Q network but uses the tanh activation function in the output layer. We use the Adam optimizer with adaptive moment estimation [30] to learn the neural network parameters, setting the learning rates for the X network and Q network to 0.00001 and 0.001, respectively. The soft update rate of the target network is set to 0.01. To explore better strategies, we set the X network noise to 0.1, and the Q network uses ϵ -greedy for action exploration. The initial value of ϵ is 1, and it gradually decreases to 0.002 and then remains constant during training. The size of the experience replay buffer is set to 2.5×10^5 . The parameters for the MEC system are shown in Table 1.

B. NUMERICAL SETUP

To better fit the rewards and optimal policy provided by the environment, we generated 1000 tasks based on the task parameters in Table 1. The first 800 tasks were utilized as a

TABLE 1. Simulation parameters.

Symbol	Values
d_m	[3, 5] MB
c_m	500 (Hz/bit)
T	3.8 s
f_m^l	[2, 3] GHz
p_m	2 W
p_n	0.5 W
k	10^{-29}
W	2.0 MHz
$f_{n,m}$	[5, 10] GHz
g_n	50^{-4}
α	0.5
N_0	-174 dBm/Hz
N	3
γ	0.9

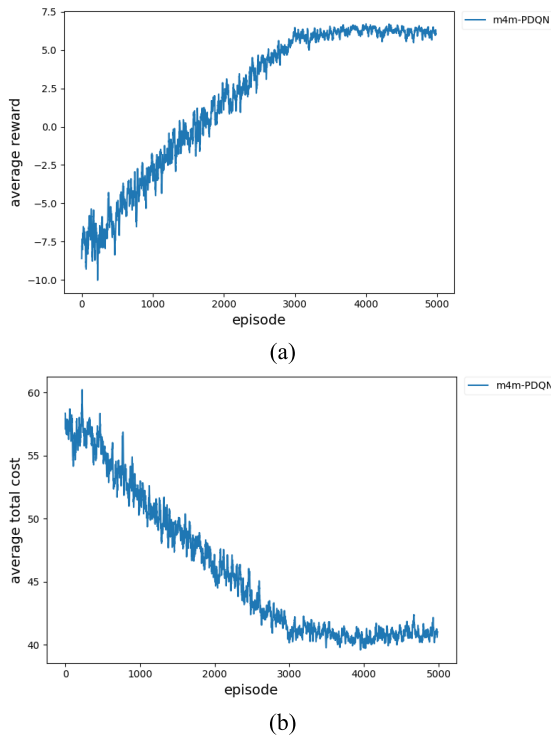


FIGURE 3. Graph of the change in reward and the weighted sum of delay and energy consumption over the training epochs.

training set to train the model, and the remaining 200 tasks were utilized as a test set to evaluate the model’s performance. Figure 3 shows the training process, which indicates that in the early stages of training, the total reward was low due to the high probability of random action selection as the agent was in an exploratory learning stage. With an increase in the number of iterations, the agent transitioned from the exploration learning stage to utilizing experienced states, and the algorithm quickly converged, with the total reward value tending to stabilize.

Next, we conduct comparative experiments on some algorithms from two perspectives: the number of users and the computing resources of edge servers. Four baseline approaches are employed for comparison as follows:

1) ALL-LOCAL

Tasks are locally executed by the user.

2) RANDOM

A random percentage of tasks are offloaded to a randomly selected edge server, and the remaining tasks are locally executed by the user.

3) UNIFORM DISTRIBUTION ALGORITHM (UDA)

Half of the tasks are offloaded to edge servers with sufficient computing resources, the edge servers allocate moderate computing resources to execute the offloaded tasks, and the remaining tasks are locally executed by the user.

4) DETERMINISTIC POLICY GRADIENT (DPG) AND EXPERIENCE REPLAY

A continuous action-based deep reinforcement learning (DRL) algorithm [30] that uses the experience replay mechanism.

The graph in Figure 4 shows the variation in the average total cost and average success rate of execution with respect to the number of users. As shown in Figure 4(a), the average total cost of all algorithms increases with an increase in the number of users. Specifically, when the number of users is

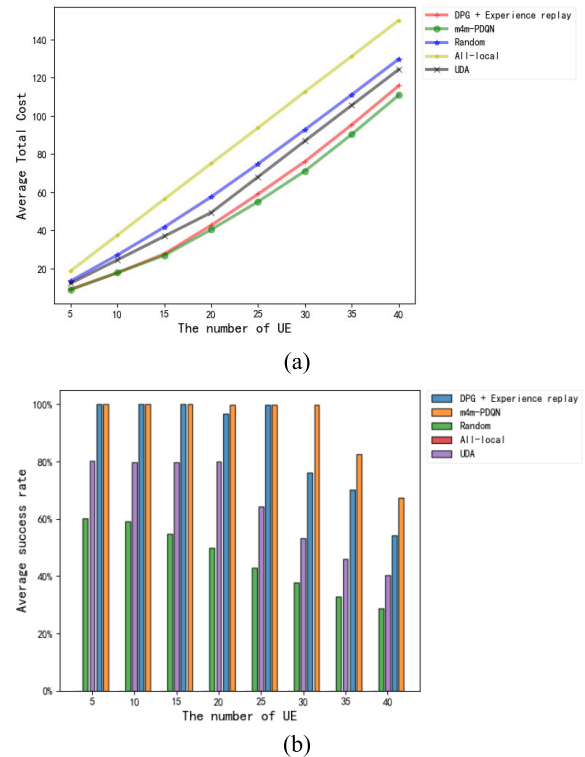


FIGURE 4. Graph of the variation in the average total cost and average success rate of execution with respect to the number of UE devices.

less than or equal to 15, the average total cost of the m4m-PDQN algorithm and the DPG algorithm with experience replay is significantly lower than that of other strategies, with the former being up to 27% lower than the UDA algorithm. When the number of users is greater than 15, the m4m-PDQN algorithm outperforms the other algorithms, with the average total cost being up to 6.5% lower than that of the DPG algorithm with experience replay. Notably, when the number of users is less than or equal to 15, the performances of the m4m-PDQN algorithm and the DPG algorithm with the experience replay mechanism are similar, which indicates that both algorithms can achieve good results via training when MEC computing resources are relatively abundant.

As shown in Figure 4(b), the m4m-PDQN algorithm consistently achieves an average execution success rate that is not lower than that of the other algorithms, with a maximum performance improvement of over 20% compared to the DPG algorithm with experience replay. However, when the number of users exceeds 25, the average execution success rate of other algorithms significantly decreases due to the insufficient computing resources of the edge servers to meet the users' offloading demands, with the exception of the m4m-PDQN algorithm. Moreover, the average success rate of all local methods remains 0% as the users' available computing resources are insufficient to satisfy the task's latency tolerance requirements.

Figure 5 shows the variation in the average total cost and average success rate of execution with respect to the sum

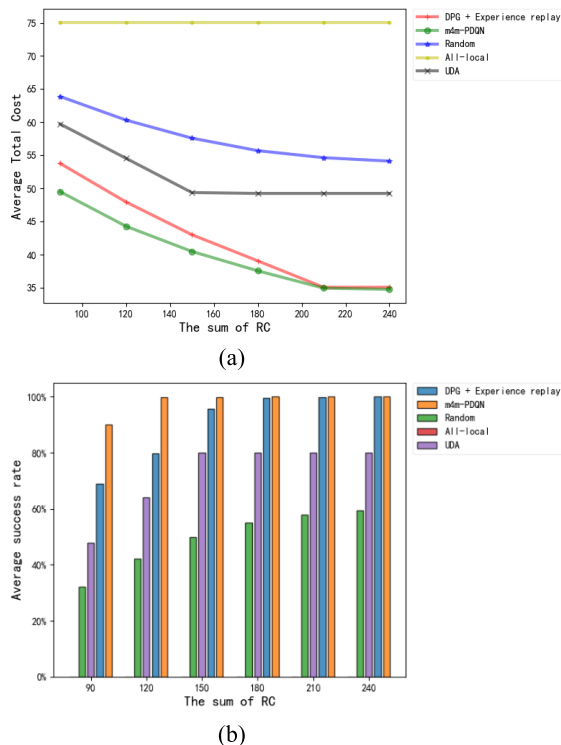


FIGURE 5. Graph of the variation in the average total cost and average success rate of execution with respect to the sum of RC.

of RC. In Figure 5(a), since the All-local strategy locally executes all tasks, the computation resources of the edge server do not affect its average total cost. When the sum of RCs not less than 150, the average total cost of the UDA algorithm is relatively stable as the computation resources of the edge server are sufficient to allocate appropriate computation resources for all offloading tasks. When the sum of RCs not less than 210, the m4m-PDQN algorithm and the DPG algorithm with experience replay have similar performance, indicating that the training effects of these two algorithms are similar when computational resources are sufficiently abundant. When the sum of RCs less than 210, the average total cost of the m4m-PDQN algorithm is much lower than that of the other algorithms, being up to 7.8% lower than that of the DPG algorithm with experience replay.

Figure 5(b) shows that as the sum of RC increases, the average execution success rate of all algorithms, with the exception of the All-local strategy, will keep increasing until stability is reached. However, the average execution success rate of the All-local strategy will remain stable, as it locally executes all tasks and is not affected by changes in the computation resources of the edge server. Overall, the m4m-PDQN algorithm has a significantly higher average execution success rate than other algorithms when computation resources are relatively scarce, with a maximum advantage over the DPG algorithm with experience replay of more than 20%. When computing resources are abundant, the average success rate of the m4m-PDQN algorithm is similar to that of the DPG algorithm with experience replay but significantly higher than the average success rates of the other three algorithms, with a maximum advantage over the UDA algorithm of approximately 20%.

VI. CONCLUSION

This paper considers a resource-constrained MEC system in the WBAN scenario, which involves multiple users and multiple edge servers. The aim is to overcome the limitations of a single-server MEC system in meeting the latency and energy requirements of multiple users. Specifically, the computing tasks and available computing capabilities of UEs are time-varying. We therefore design a weighted sum minimization problem for system latency and energy consumption and propose the m4m-PDQN algorithm. We then conduct simulation experiments and compare our proposed algorithm with four other algorithms, demonstrating its excellent performance in terms of both average total cost and task execution success rate.

REFERENCES

- [1] S. Movassaghi, M. Abolhasan, J. Lipman, D. Smith, and A. Jamalipour, "Wireless body area networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1658–1686, 3rd Quart., 2014.
- [2] G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Artificial intelligence enabled distributed edge computing for Internet of Things applications," in *Proc. 16th Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, May 2020, pp. 450–457.
- [3] T. Leppanen, "Distributed artificial intelligence with multi-agent systems for MEC," in *Proc. 28th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2019, pp. 1–8.

- [4] Y. Yu, "Mobile edge computing towards 5G: Vision, recent progress, and open challenges," *China Commun.*, vol. 13, no. 2, pp. 89–99, 2016.
- [5] H. Jin, M. A. Gregory, and S. Li, "A review of intelligent computation offloading in multiaccess edge computing," *IEEE Access*, vol. 10, pp. 71481–71495, 2022.
- [6] C. Feng, P. Han, X. Zhang, B. Yang, and Y. Liu, "Computation offloading in mobile edge computing networks: A survey," *J. Netw. Comput. Appl.*, vol. 202, Jun. 2022, Art. no. 103366.
- [7] M. Maray and J. Shuja, "Computation offloading in mobile cloud computing and mobile edge computing: Survey, taxonomy, and open issues," *Mobile Inf. Syst.*, vol. 2022, Jun. 2022, Art. no. 1121822.
- [8] P. K. Bishoyi and S. Misra, "Enabling green mobile-edge computing for 5G-based healthcare applications," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 3, pp. 1623–1631, Sep. 2021.
- [9] P. K. Bishoyi and S. Misra, "Towards energy-and cost-efficient sustainable MEC-assisted healthcare systems," *IEEE Trans. Sustain. Comput.*, vol. 7, no. 4, pp. 958–969, Oct. 2022.
- [10] M. Chowdhury, "FETES: A fast, emergency timeslot allocation, and three-tier energy saving-based task execution strategy for wireless body area network," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 42, no. 3, p. 189, 2023.
- [11] S. Kim, "A new two-stage bargaining game approach for intra- and inter-WBAN management," *Mobile Inf. Syst.*, vol. 2021, Nov. 2021, Art. no. 5798741.
- [12] C. Zhu, J. Ren, H. Wan, and T. Qin, "Wireless body area networks task offloading method combined with multiple communication and computing resources supported by MEC," *IET Commun.*, vol. 17, no. 10, pp. 1188–1198, Jun. 2023.
- [13] Y. Liao, Y. Han, Q. Yu, Q. Ai, Q. Liu, and M. S. Leeson, "Wireless body area network mobility-aware task offloading scheme," *IEEE Access*, vol. 6, pp. 61366–61376, 2018.
- [14] Y. Liao, Q. Yu, Y. Han, and M. S. Leeson, "Relay-enabled task offloading management for wireless body area networks," *Appl. Sci.*, vol. 8, no. 8, p. 1409, Aug. 2018.
- [15] X. Yuan, H. Tian, H. Wang, H. Su, J. Liu, and A. Taherkordi, "Edge-enabled WBANs for efficient QoS provisioning healthcare monitoring: A two-stage potential game-based computation offloading strategy," *IEEE Access*, vol. 8, pp. 92718–92730, 2020.
- [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] N. Kiran, C. Pan, S. Wang, and C. Yin, "Joint resource allocation and computation offloading in mobile edge computing for SDN based wireless networks," *J. Commun. Netw.*, vol. 22, no. 1, pp. 1–11, Feb. 2020.
- [18] Y. Wang, W. Fang, Y. Ding, and N. Xiong, "Computation offloading optimization for UAV-assisted mobile edge computing: A deep deterministic policy gradient approach," *Wireless Netw.*, vol. 27, no. 4, pp. 2991–3006, 2021.
- [19] C. Chen, Y. Zhang, Z. Wang, S. Wan, and Q. Pei, "Distributed computation offloading method based on deep reinforcement learning in ICV," *Appl. Soft Comput.*, vol. 103, May 2021, Art. no. 107108.
- [20] Z. Hu, J. Niu, T. Ren, B. Dai, Q. Li, M. Xu, and S. K. Das, "An efficient online computation offloading approach for large-scale mobile edge computing via deep reinforcement learning," *IEEE Trans. Services Comput.*, vol. 15, no. 2, pp. 669–683, Mar. 2022.
- [21] S. Koo and Y. Lim, "A cluster-based optimal computation offloading decision mechanism using RL in the IIoT field," *Appl. Sci.*, vol. 12, no. 1, p. 384, Dec. 2022.
- [22] 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.
- [23] W. Cheng, X. Liu, X. Wang, and G. Nie, "Task offloading and resource allocation for Industrial Internet of Things: A double-dueling deep Q-network approach," *IEEE Access*, vol. 10, pp. 103111–103120, 2022.
- [24] Z. Chen and X. Wang, "Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach," *EURASIP J. Wireless Commun. Netw.*, vol. 2020, no. 1, pp. 1–21, Dec. 2020.
- [25] J. Wang, H. Ke, X. Liu, and H. Wang, "Optimization for computational offloading in multi-access edge computing: A deep reinforcement learning scheme," *Comput. Netw.*, vol. 204, Feb. 2022, Art. no. 108690.
- [26] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. M. Leung, "Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1517–1530, Jan. 2022.
- [27] C. Li, Y. Zhang, and Y. Luo, "DQN-enabled content caching and quantum ant colony-based computation offloading in MEC," *Appl. Soft Comput.*, vol. 133, Jan. 2023, Art. no. 109900.
- [28] L. Huang, S. Bi, and Y. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.
- [29] J. Xiong, Q. Wang, Z. Yang, P. Sun, L. Han, Y. Zheng, H. Fu, T. Zhang, J. Liu, and H. Liu, "Parametrized deep Q-networks learning: Reinforcement learning with discrete-continuous hybrid action space," 2018, *arXiv:1810.06394*.
- [30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
- [31] D. Silver, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Mach. Learn. (ICML)*, vol. 2014, pp. 387–395.
- [32] Y. Chen, S. Han, G. Chen, J. Yin, K. N. Wang, and J. Cao, "A deep reinforcement learning-based wireless body area network offloading optimization strategy for healthcare services," *Health Inf. Sci. Syst.*, vol. 11, no. 1, p. 8, 2023.
- [33] L. Zhang, X. Yuan, J. Luo, C. Feng, G. Yang, and N. Zhang, "An adaptive resource allocation approach based on user demand forecasting for e-healthcare systems," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2022, pp. 349–354.
- [34] A. I. Awad, M. M. Fouda, M. M. Khashaba, E. R. Mohamed, and K. M. Hosny, "Utilization of mobile edge computing on the Internet of Medical Things: A survey," *ICT Exp.*, vol. 9, no. 3, pp. 473–485, 2022.



DING YIN (Member, IEEE) is currently pursuing the M.S. degree in computer technology with Zhejiang Sci-Tech University, Hangzhou, China. His research interests include wireless body area networks and mobile edge computing.



LI CHEN received the Ph.D. degree from the Harbin Institute of Technology. She is currently an Associate Professor with Jiaying University. Her main research interests include machine learning and mobile communications technology. She is a member of the China Computer Federation.



JUN YANG received the Ph.D. degree from Sun Yat-sen University, Guangzhou. He is currently an Associate Professor with Jiaying University. His current research interests include the areas of image processing, machine learning, artificial intelligence techniques, and robotic control systems.



KUN DENG was born in 1980. He is currently pursuing the Ph.D. degree in engineering. He is an Associate Professor. His research interests include data mining and complex network structure analysis.