Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2022.022606
Article

Tech Science Press

Multi-Classification and Distributed Reinforcement Learning-Based Inspection Swarm Offloading Strategy

Yuping Deng[1], Tao Wu[1], Xi Chen[2,*] and Amir Homayoon Ashrafzadeh[3]

[1]Chengdu University of Information and Technology, Chengdu, 610255, China
[2]Southwest Minzu University, Chengdu, 610041, China
[3]RMIT University, Melbourne, 3058, Australia
*Corresponding Author: Xi Chen. Email: cx@swun.edu.cn
Received: 12 August 2021; Accepted: 08 November 2021

Abstract: In meteorological and electric power Internet of Things scenarios, in order to extend the service life of relevant facilities and reduce the cost of emergency repair, the intelligent inspection swarm is introduced to cooperate with monitoring tasks, which collect and process the current scene data through a variety of sensors and cameras, and complete tasks such as emergency handling and fault inspection. Due to the limitation of computing resources and battery life of patrol inspection equipment, it will cause problems such as slow response in emergency and long time for fault location. Mobile Edge Computing is a promising technology, which can improve the quality of service of the swarm by offloading the computing task of the inspection equipment to the edge server nearby the network. In this paper, we study the problem of computing offloading of multi-devices multi-tasks and multi-servers in the intelligent patrol inspection swarm under the condition of a dynamic network environment and limited resources of servers and inspection equipment. An effective adaptive learning offloading strategy based on distributed reinforcement learning and multi-classification is proposed to reduce the task processing delay and energy consumption of the intelligent inspection swarm and improve the service quality. Numerical experimental results demonstrate that the proposed strategy is superior to other offloading strategies in terms of time delay, energy consumption and quality of service.

Keywords: Mobile edge computing; deep reinforcement learning; multi-classification; time delay; energy consumption; intelligent inspection swarm; swarm head

1 Introduction

At present, the production and life of human beings are closely linked with weather forecasts, power grids and communication networks. The accuracy of weather forecasts, the safe and stable operation of power grids and communication networks always depend on the normal operation of all kinds of meteorological observation facilities, power facilities and network facilities respectively. Due to the complexity and dynamics of the environment in which various facilities are located (pest tracking) and

the diversity of monitoring objects (facility monitoring and biological monitoring), the simple introduction of fixed sensors and cameras may not be able to fully meet the monitoring requirements. Therefore, long-term and effective dynamic monitoring equipment is an important mean to maintain its continuous operation. Intelligent patrol inspection swarm is one of the effective schemes of mobile all-sides monitoring, through the use of multiple low-cost mobile devices (e.g., lightweight drones, smart cars) to construct a unified control, efficient collaboration and dynamic inspection swarm. The swarm nodes dynamically cooperate to complete the monitoring task, transmitting all monitoring data to the cloud, and then returning the result to the control center after processing. Finally, the control center will maintain the equipment according to the result of data processing [1].

However, there are also some problems in the intelligent inspection swarm. Inspection devices (unmanned aerial vehicles, intelligent cars, etc.) still have defects in battery life and computing power due to the strict constraints of strong mobility, small size and production cost. At the same time, the types of data collected by inspection devices (such as UAVs, intelligent cars, etc.) are diversified with the development of hardware and Internet technology, and the data processing and storage capacity also are enhanced. The data processing mode with cloud computing as the core still has limitations in transmission delay, energy consumption, data security and other aspects [2,3]. Therefore, how to ensure the transmission and processing of data at the minimum cost (delay, energy consumption), and at the same time, maximize swarm's monitoring time, and then realize the goal to improve the quality of experience in a centralized control the equipment monitoring system. It is one of the key issues to be solved in the intelligent inspection swarm.

Multi-Access Mobile Edge Computing (MEC) [2] is the key technology to solve this problem. At present, it is widely used in the industrial Internet of Things, autonomous driving, blockchain [4] and other fields.

A large number of computing tasks are transferred from Mobile wireless devices to nearby access points and executed through connected servers [2], so as to reduce the energy consumption of devices and the delay and energy consumption of data transmission to the cloud, and effectively ensure data security [5,6].

However, not all the computing tasks of inspection devices can be offloaded to the MEC server for the following reasons. Firstly, there are some computing tasks that must be processed in real-time with the resources of patrol inspection devices. Secondly, due to the limited bandwidth resources, if a task is offloaded to a MEC server, the wireless channel of the upload link will be severely congested. Further, it will result in the task transmission time greater than the time cost of local processing tasks, which is contrary to the purpose of reducing the delay. Thirdly, the MEC server itself has limited computing performance and resource storage capacity. As one of the key technologies of MEC, computational offloading technology provides a solution to solve the above problems. Computational offloading technology mainly includes two aspects: offloading strategy and resource allocation. Among them, the offloading strategy refers to the decision of how to confirm the size of the offloading data and the content of the offloading. The results of the offloading decision can be divided into three cases: local execution, partial offloading and total offloading. Resource allocation refers to the allocation of computing and communication resources for the offloaded tasks, that is, the selection of appropriate offloading nodes, balance the utilization of computing resources and communication resources in the system, so as to achieve the goal of minimizing the delay, energy consumption and improving the overall performance of the network. In this paper, we aim to minimize the time delay and energy consumption of a MEC system with multi-edge nodes at different positions and an intelligent inspection swarm. It is a complex problem of multi-task multi-user and multi-edge nodes MEC system.

Computational offloading is a nondeterministic polynomial NP-hard problem. Therefore, solutions based on heuristic or approximate algorithms have been widely adopted in recent years. However, such

algorithms require quite a lot of iterations to find the local optimal solution, which is not suitable for complex and dynamic MEC systems [7–9]. With the breakthrough of artificial intelligent technology, there are various solutions to the computational offloading problem of edge computing, and after the addition of artificial intelligent technology, a calculation scheme of offloading is more efficient than a traditional algorithm. And it also solved the problem of a large amount of data input, of which the most outstanding method is reinforcement learning. By this method, the computing problem will be modeled as an MDP process because this technology can adaptively learn the offloading decision and calculate the execution cost in the process of task offloading in the dynamic environment, realize the decision ability of intelligent offloading of computing tasks, and optimize the execution efficiency of computing tasks. However, they are not suitable for simultaneous offloading users and computational unload problems with many tasks and large action space because the search time of action space and storage capacity of the device is limited.

In this article, we consider a MEC network with multiple edge computing nodes and an intelligent inspection swarm consisting of multi-inspection devices. In the intelligent patrol MEC system, the offloading task mainly includes the processing of the monitoring data (image, video and sensor data) generated by the patrol devices during the mobile monitoring process. Assume that in slot t, multi-tasks are generated simultaneously by intelligent inspection terminals respectively, and the tasks follow binary offloading decision [10], the offloading decision is determined by swarm head for all terminal tasks. Specifically, we send the task size generated by the mobile terminal and the distance between the terminal and each edge node to the swarm head, and the swarm head will jointly determine the offloading decision of all tasks according to the above data, so as to minimize the energy consumption and delay of MEC network processing. We propose a distributed DRL-based approach to select an appropriate compute node for all the tasks generated by the swarm nodes within a time slot t. The algorithm has an excellent performance in optimizing the minimum tasks' response time and total energy consumption of the intelligent patrol inspection MEC system, which reduces the energy consumption of the system and improves the quality of experience of the control center user.

The rest of this paper is arranged as follows: Section 2 summarizes the relevant work; Section 3 introduces the system model, proposed problem model; Section 4 describes the solution in detail. Section 5 records the experimental simulation results and result's analysis; Section 6 records the conclusion and future work.

2 Related Work

There have been many pieces of research on the task offloading of the MEC system. Firstly, the traditional optimization iterative method is widely used in the optimization of MEC networks. Tong et al. [8] designed a hierarchical arc edge computing architecture based on the distance between the MEC server and the user, and proposed an optimized offloading scheme to minimize the task duration by using a heuristic algorithm. A coordinate descent (CD) [11] method is proposed to search along one variable dimension at a time. Mao et al. [12] proposed a dynamic computational offloading (LODCO) algorithm based on Lyapunov optimization, which determines offloading strategy by minimizing the cost of task execution.

In contains multiple mobile devices network using binary offloading decision-making, the task cannot be further divided, only the overall execution locally or offloaded to MEC servers. By enumerating the possible values for all tasks is computationally impossible. Because, it is a typical mixed integer nonlinear programming problem and generally a non-deterministic polynomial hard (NP-hard) problem, which is very difficult to solve [13]. Therefore, many researchers propose improved algorithms based on game theory [14] to solve the optimal computational offloading strategy. Chen et al. [13] studied the multi-user computing offloading problem of MEC in a multi-wireless channel environment, and designed

a distributed computing offloading algorithm to look for the Nash equilibrium of the game. Guo et al. [15] proposed the algorithm MECC based on game theory, which fully considered the computation-offloading problem of cloud-edge-end collaboration. Considering the price of MEC servers and the SDN environment, Mitsis et al. [16] proposed a reinforcement learning-based, iterative but low-complexity algorithm to realize the selection process of MEC server, and to determine the price of the computing service of the optimal MEC server and the optimal data offloading amount for the end-user based on game theory and optimization techniques. Li et al. [17] designed the distributed algorithm of ADMM (alternating direction multiplier) to control the total energy consumption of the MEC system. In addition, Yang et al. [18] proposed to allocate resources in advance and save the calculation time of loading decision by predicting user location on the premise of fully considering user mobility. However, the above algorithms are limited by the balance between optimality and computational complexity, so they are not suitable for real-time computing offloading in MEC networks with a real-time environment.

The development of deep learning has brought a great breakthrough to solve the offloading strategy optimization problem of high-dimensional state and action space in a MEC environment, among which the deep reinforcement learning method is particularly outstanding. The improved DQN and DQN [19–21] algorithms are usually applied to the joint optimization of computational offloading and resource allocation in the discrete action space of a single agent. Xiong et al. [21] proposed that the use of multiple playback memories to store fewer interaction experiences is superior to the original DQN method in terms of convergence and resource allocation. In addition, DDPG [22,23] algorithm performs better than DQN in high and continuous state and action spaces. Chen et al. [23] designed a dynamic resource management method DDRM based on DDPG, which solved the problems of joint power control of mechatronics and dynamic resource management of computing resource allocation in the industrial Internet of Things. Qian et al. [24] proposed a joint push and cache strategy based on hierarchical reinforcement learning (HRL). By dividing the total problem into two sub-problems, different basic reinforcement learning methods are adopted to solve the sub-problems.

In general, these works mainly considered the offloading decisions and resource allocation schemes of computation-intensive tasks in the MEC network when a single mobile device offloads tasks to one or more MEC servers [25], or when multiple mobile devices offload tasks to the same MEC server [10,26]. In the centrally controlled intelligent inspection cluster, considering the high moving frequency of the inspection equipment and the channel state between it and the MEC server, it is necessary to study the computational offloading and resource allocation of all the slave tasks in the cluster determined by a single patrol device.

3 System Model and Problem Formulation

3.1 System Model

As shown in Fig. 1, we consider an intelligent inspection MEC system, which consists of W edge computing nodes and an intelligent inspection cluster composed of N inspection devices, $N = \{1, 2, 3, 4 \ldots n\}$, $W = \{1, 2, 3, 4 \ldots w\}$. The inspection equipment is a terminal with computing power and wireless transmission function and has mobility. An intelligent inspection cluster is composed of N inspection devices connected by WIFI, which contains a swarm head node. Maintenance personnel control cluster head nodes through remote command, and then control all cluster slave devices to work together in order to complete the dynamic inspection task. The cluster can independently elect a new cluster head node to ensure the effectiveness of the cluster's work when the cluster head node is unable to work. The edge node is the MEC server, which is composed of the base station (BS) and the server. In the system, the BS and the server are fixed, the former for receiving data signals, the latter for processing data. It is connected through optical fiber, and the transmission delay can be ignored. Each inspection wireless

device in the cluster has I independent and indivisible pending tasks, which is denoted by the set $I = \{1, 2, 3, 4..., i\}$. In addition, the workload of the *i-th* task of the cluster device n can be expressed as $R_{ni}(t) = (C_{ni}(t), S_{ni}(t))$, with a tuple, where the CPU workload required by $C_{ni}(t)$ to execute task $R_{ni}(t)$, and $S_{ni}(t)$ represents the data required to calculate task $R_{ni}(t)$. Since the behavior of the devices in swarm N are controlled by the swarm head and all the devices are networked, in order to maximize the working life of the swarm and make full use of the computing capacity of mobile devices, the computing and offloading decision of all the tasks in the swarm is completed by the swarm head centrally. After that, offloading results are distributed to other slave devices for execution. In this paper, a quasi-static scenario is considered, that is, the inspection equipment will move in different time slots t. Namely, the positions and tasks of the inspection equipment do not change in the time slot t. The cluster head completes task offloading decisions, processes local tasks, and transmits tasks to the MEC server based on the offloading decision.

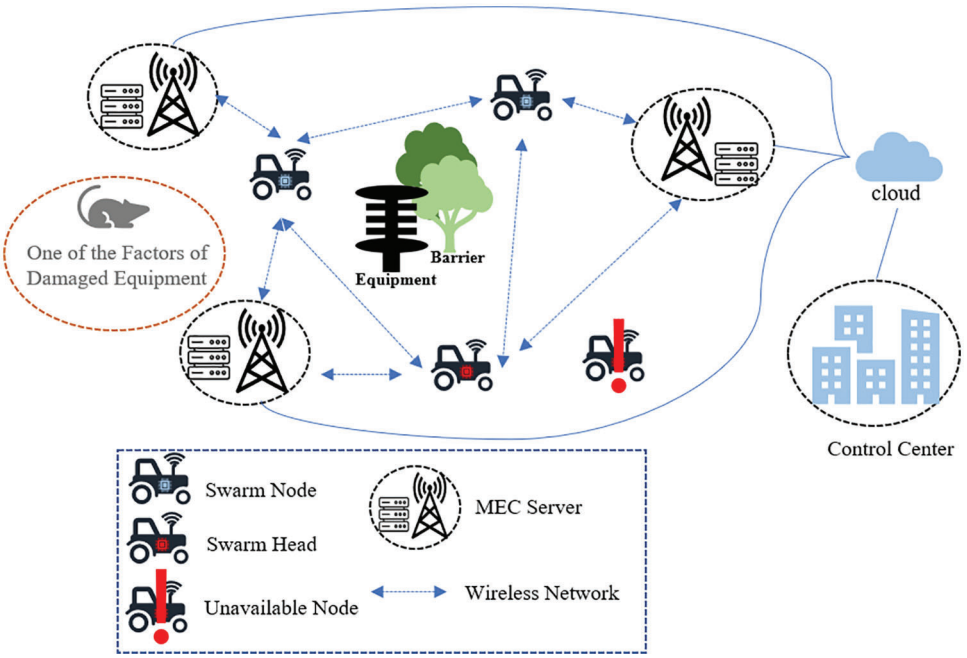

Figure 1: MEC system model of intelligent inspection swarm

In each time slot t, the swarm head determines the offloading decision after collecting related information of each slave node. Next, it distributes the decision, and then each node carries out tasks after responding to the decision. The decision of offloading the *i-th* task of the cluster device n to the edge server w is expressed as $x_{niw}(t) \in \{0, 1\}$, where $w = 0$ represents the local computing node, the corresponding value of the selected computing node is 1, and the corresponding value of other nodes is 0, satisfying

$$\sum_{w \in W} x_{niw}(t) = 1, \quad \forall i \in I. \tag{1}$$

constraints. Due to the mobility of inspection devices, the position of cluster devices will change at different time slots t, and the task and channel quality will also change due to the distance change between the devices and BS, thus the offloading decision also should be changed.

3.2 Problem Formulation

3.2.1 Local Computing Model

The swarm inspection device has a certain computing capacity. If the computing resources of the device are sufficient, or the transmission cost of a task is greater than the execution cost of the device itself, the task should be executed locally. Assuming that the delay of the local computing task is only generated by the CPU, and the maximum CPU frequency of the inspection device is expressed in terms of $f_{n,max}$, the actual computation frequency (In each slot, the execution of different tasks in parallel, and the latency of different tasks is mutually influenced.) is f_n, which is the number of CPU cycles running per second while the inspection device is calculating tasks. The delay of the local execution of task i of data device n can be expressed as:

$$d_{ni}^{local}(t) = \frac{C_{ni}(t)}{f_n(t)}. \tag{2}$$

The energy consumption of a single task executed locally can be expressed as:

$$e_{ni}^{local}(t) = \mathcal{K}(f_n(t))^3 d_{ni}^{local}(t). \tag{3}$$

where $\mathcal{K}$ represents effective switching capacitance in the chip [27].

3.2.2 Edge Computing Model

The patrol inspection equipment resources are insufficient to deal with that the task is computationally intensive or the number of tasks is large. Therefore, the device should offload such tasks to the MEC server and then return the results to the device. Since the results of task processing are directly returned to the control center without the need to return to the inspection equipment. Hence, the return time of task results is not included in the calculation of offloading delay. In general, the delay of offloading the task i of device n to the edge server w $d_{niw}^{edge}(t)$ includes the delay of data transmission required by the device to complete the task $d_{niw}^{trans}(t)$ and the delay of executing the task by the edge server w $d_{niw}^{off}(t)$. It is assumed that the task offloading of the inspection equipment adopts orthogonal frequency division multiple access (OFDMA), and different task transmission occupies different channels without interference. According to [28], the data transmission rate from device n to base station w can be expressed as:

$$r_{nw}(t) = Blog_2\left(1 + \frac{h_{nw}(t)p_n}{pl_{nw}(t)BN_0}\right). \tag{4}$$

where, B represents the communication bandwidth between the device n and the BS w, p_n is the transmission power of the device n, N_0 is the noise power spectral density of the receiving BS, $h_{nw}(t)$ is the channel gain, $pl_{nw}(t)$ the path loss between the device n and the BS w, respectively, where $h_{nw}(t)/pl_{nw}(t) = g_0(Dis_0/Dis_{nw}(t))^{\sigma}$, g_0 is the path loss constant, Dis_0 is the reference distance, Dis_{nw} is the distance between the device n and the BS w, σ is the path loss exponent. Therefore, the transmission time for the task i generated by device n offloading to the edge server w for processing is denoted as:

$$d_{niw}^{trans}(t) = \frac{S_{ni}(t)}{r_{nw}(t)}. \tag{5}$$

Let C be the total number of CPUs of the edge server when computing data. The maximum CPU frequency of the edge server is denoted by $f_{w,max}$, and the actual computing frequency is $f_{nw}(t)$, then the task processing delay in

$$d_{niw}^{off}(t) = \frac{C_{ni}(t)}{f_{nw}(t)}. \tag{6}$$

The energy consumption of the task i generated by device n offloading to the edge server w for processing can be expressed as:

$$e_{niw}^{edge}(t) = p_n d_{niw}^{trans}(t) + S_{ni}(t) q_{w.} \tag{7}$$

Here q_w is the energy consumed per bit of data on the MEC server.

The delay of the *i-th* task of device n $d_{ni}(t)$ is expressed as:

$$d_{ni}(t) = \sum_{w \in W} (x_{niw}(t) d_{ni}^{local}(t) + x_{niw}(t) d_{niw}^{trans}(t) + x_{niw}(t) d_{niw}^{off}(t)). \tag{8}$$

Energy consumption is expressed as:

$$e_{ni}(t) = \sum_{w \in W} (x_{niw}(t) e_{ni}^{local}(t) + x_{niw}(t) e_{niw}^{edge}(t)). \tag{9}$$

Therefore, the total time consumption of the whole system in time slot t is:

$$D^{total}(t) = \max_{n \in N, i \in I} d_{ni}(t). \tag{10}$$

Total energy consumption is:

$$E^{total}(t) = \sum_{n \in N} \sum_{i \in I} e_{ni}(t). \tag{11}$$

In this paper, considering the balance between improving the QoE of intelligent patrol inspection MEC system, the utilization of cluster computing power and the working time of swarm, the problem of optimizing and minimizing the computing overhead (the weighted average sum of delay and energy consumption) is proposed. Within the time slot t, the computational overhead of all tasks in the cluster is:

$$\mathbb{Q}(t) = \alpha D^{total}(t) + \beta E^{total}(t). \tag{12}$$

where α and β respectively represent the weighted parameters of time and energy consumption required for all tasks, $\alpha, \beta \in [0, 1]$ and $\alpha + \beta = 1$. Based on the above formula, at each time slot t, the target optimization problem under the MEC environment of intelligent patrol cluster is expressed as:

$$P1: \min_{\{x\}} \mathbb{Q}(t) \tag{13}$$

$$s.t. \quad C1: x_{niw}(t) \in \{0, 1\} \tag{13a}$$

$$C2: \sum_{w \in W} x_{niw}(t) = 1 \tag{13b}$$

$$C3: \sum_{i \in I} x_{niw}(t) f_n(t) \leq f_{n,max} \tag{13c}$$

$$C4: \sum_{ni \in NI} x_{niw}(t) f_{nw}(t) \leq f_{w,max} \tag{13d}$$

$\forall i \in I,\ n \in N,\ w \in W$

Constraint $C1$ represents the binary offloading mode; constraint $C2$ represents that a task can only be offloaded to one node; constraint $C3$ and $C4$ represent that the inspection equipment and the edge server need to provide sufficient computing resources for the tasks to be processed at this node, respectively.

4 DDRL and Multi-Classification Based Solution

Because Problem P1 is a three-dimensional integer nonlinear programming and NP-hard problem, it is difficult to solve directly. Problem P1 contains two constraints, offloading decision constraints $C1$ and $C2$, and resource constraints $C3$ and $C4$. Therefore, problem P1 can be decomposed into offloading decision subproblem and resource allocation subproblem. We propose a distributed offloading algorithm, MCDDRL, which is based on deep reinforcement learning and multi-classification ideas, to solve the problem of computing offloading decisions and resource allocation in the MEC system with multi-devices multi-task multi-edge nodes offloading simultaneously in a dynamic environment.

Reinforcement learning is one of the paradigms and methodologies of machine learning, which is used to describe and solve the problem that agents learn strategies to maximize returns or achieve specific goals in the process of interacting with the environment. The typical model of reinforcement learning is Markov Decision Process (MDP), which generally includes State Space (S), Action Space (A) and Reward Function (R).

In the intelligent inspection MEC system, the task and position of the inspection equipment are time-varying, so problem P1 can be transformed into an MDP problem.

State space: In time slot t, the state space of the system is expressed as:

$$\boldsymbol{s}(t) = \{\boldsymbol{r}(t),\ \boldsymbol{d}(t)\} \tag{14}$$

where, in time slot t, $\boldsymbol{r}(t)$ represents the task information generated by all inspection devices, and $\boldsymbol{d}(t)$ represents the distance information between all inspection devices and BS.

Action space: By observing the current state of the MEC system $\boldsymbol{s}(t)$, we need to determine the offloading plan for each task:

$$\boldsymbol{a}(t) = \{x_{niw}(t) | x_{niw}(t) \in \{0,\ 1\}\} \tag{15}$$

The reward function is defined as a weighted sum of the energy consumption and delay of all devices to complete the task.

$$\boldsymbol{re}(t) = \mathbb{Q}(\boldsymbol{s}(t),\ \boldsymbol{a}(t)) \tag{16}$$

With the increase of time slot t, the agent will obtain rewards according to state $\boldsymbol{s}(t)$ selection $\boldsymbol{a}(t)$, and the expectation of minimizing rewards, namely, the average calculated cost with the weighted sum of time delay and energy consumption as the index, is expressed as follows:

$$\mathbb{Z}(\boldsymbol{s}(t)) = \mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T} \sum_{T=1}^{T} \boldsymbol{re}(t)\right] \tag{17}$$

In general, the above problem is an MNIP problem. The traditional optimization algorithm has too many iterations, takes too long, and the process of solving the problem is complicated, which cannot adapt to the dynamic changes of the system. In order to solve the problem of dynamic changes in the environment, recently many value-based DRL algorithms have been applied to edge environments, such as DQN and

Double DQN. In the MEC system of this paper, the offloading decisions of all tasks within the time slot t are discrete, which is suitable for the value-based DRL algorithm. However, the size of offloading decision set $\boldsymbol{x}$ is $2^{(w+1)\ IN}$, it increases exponentially as the compute nodes w, equipment number n and the task of each device number i increases, respectively. Therefore, it will lead to action space being too large to find a suitable offloading strategy during time slot t by using the traditional DRL algorithm based on value, which is unfeasible for the mobile inspection system to implement higher frequency optimization.

Therefore, we propose a distributed offloading algorithm, MCDDRL, based on reinforcement learning and multi-classification ideas, for the MEC environment of intelligent inspection swarm. The algorithm structure is shown in Fig. 2 The algorithm consists of two stages: the candidate offloading scheme generation stage and the resource allocation stage. The two stages are completed alternately, and the specific process is as follows.

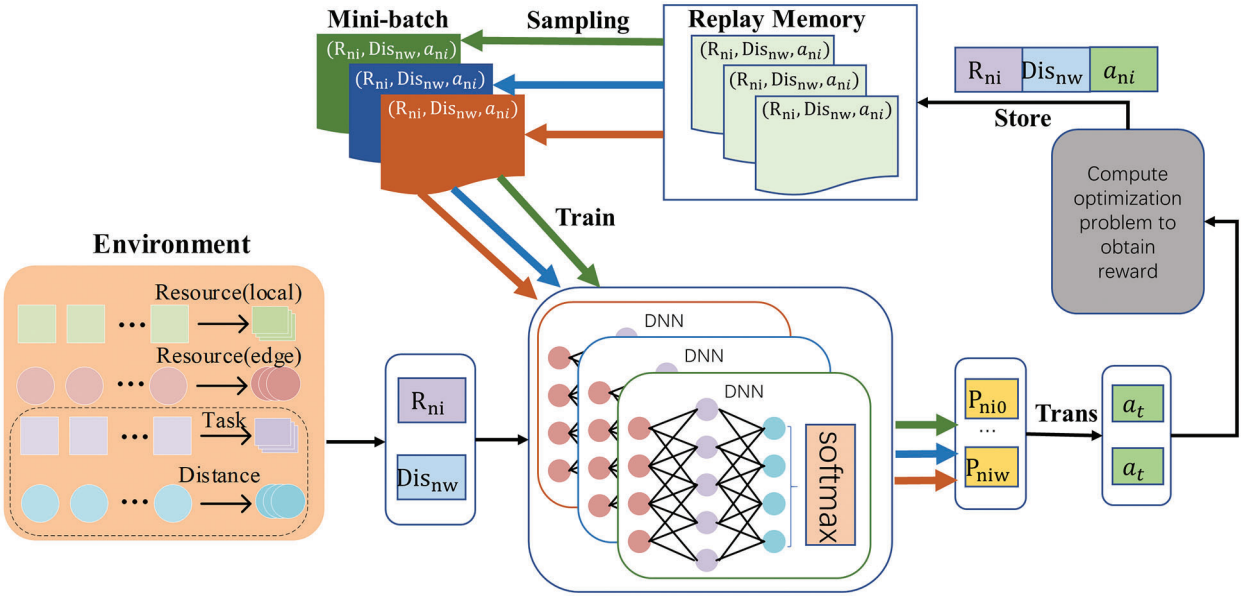

Figure 2: Architecture of MCDDRL algorithm

The algorithm pseudocode is as follows:

Algorithm 1: The MCDDRL Algorithm

Input: Inspection equipment's the task information R_t and the distance information Dis_t between every inspection equipment and every BS at each time slot t, the number of MEC server, task, device W, I, N;

Output: All inspection equipment's tasks action x_t^* and weight sum of time delay and energy consumption $\mathbb{Q}_t^*$;

Initialize k DNNs with random parameters θ_k;

Initialize Memory $\mathcal{M}$;

Set the number of time frames T;

for *1, 2, 3, ..., T* **do**

for *1, 2, 3, ..., N*I* **do**

Match each task information with the corresponding device distance information (R_{ni}, Dis_{nw})

(Continued)

Algorithm 1 (continued)

Generate *K-th* the task offloading to each node's probability P_{niw}

end for

Transform probability P_{niw}*'s* index to *k(W+ 1)* actions

Select best action x_t^* according to computing time delay and energy consumption;

for *1, 2, 3, ..., N*I* **do**

Store (R_{ni}, Dis_{nw}, x_{ni}^*) into memory $\mathcal{M}$

Randomly sample k mini-batches to train *k* DNNs and update parameters θ_k

end for

end for

Generation stage of candidate offloading actions: Firstly, we treat the computable nodes of task *i* as *W+ 1* categories, that is, the task is considered category 0 if it is evaluated locally, and category *w* if it is evaluated on the MEC server *w*. We used a multi-classification model based on DNN with softmax function as the output layer activation function to represent the offloading decision for each task. In each time slot *t*, the swarm head collects inspection equipment's the task information *R* and the distance information between every inspection equipment and every BS (which are related to data transmission rate and energy consumption) *Dis*, namely $\boldsymbol{s}(t)$, which are stored in the memory of the swarm head in the order of information arrival. The above information is sequentially input to *K* DNNs, and *K* DNNs are calculated in parallel. Next, the probability that each task belongs to each category is output, and the category index is arranged in order of probability from most to least until all tasks in the *t* slot are calculated by DNN. Each DNN's output takes the node with the same column number according to the input order of all tasks to form the offloading decision for all tasks during the time slot *t*. Finally, there are *K(W+ 1)* candidate offloading schemes are generated totally.

Resource allocation stage: firstly, we should calculate the result whether each candidate offloading scheme satisfies constraints *C*3 and *C*4 of problem P1; if not, discard that scheme, and then select the one with the smallest value $\boldsymbol{re}(t)$ from the remaining offloading schemes.

After that, the tuple $\left(R_{ni}(t),\ Dis_{ni}(t),\ \sum_{w\in W} x_{niw}(t)\right)$ of each task is stored in a memory pool $\mathcal{M}$. For *K* DNNs, the learning is carried out by random sampling from the same memory pool respectively, so as to reduce the correlation between samples and update the parameters of DNN *k* θ_k by minimizing the cross-entropy loss. In time slot *t+ 1*, the new θ_k can be used to compute the probability of new tasks being offloaded to the edge server or locally.

5 Numerical Experiment and Result Analysis

In this section, we use simulation experiments to evaluate the performance of MCDDRL algorithm. The experimental platform uses Python 3.8 under Linux system and a deep learning framework TensorFlow-2.1.0-gpu. The experimental parameters and super-parameter settings of MCDDRL algorithm are shown in the Tab. 1, and the super-parameter settings are determined through a large number of experiments.

Table 1: Simulation experiment parameter and super parameter setting

Parameters	Value
C_{ni} (cycles)	$\boldsymbol{S}_{ni} * 500$
$\boldsymbol{S}_{ni}$ (KB)	Unif (1024, 2048)
$f_{n,max}$ (GHz)	2.0
$f_{w,max}$ (GHz)	10.0
p_n (mW)	100
q_w (J/bit)	$2*10^{-8}$
Dis_{nw} (m)	Randint (1,200)
Dis_0 (m)	100
B(MHz)	2.0
$\mathcal{K}$	10^{-28}
σ	4
g_0 (dB)	−40
N_0 (dBm/Hz)	−174
Number of inspection devices	2
Number of MEC servers	3
Number of tasks of each device	3
Loss function	Sparse categorical cross entropy
Mini-batch size	32
Optimizer	Adam
Learning rate	0.01
Replay memory size	3072
α	0.5
β	0.5

To evaluate the influence of parameter K on MCDDRL algorithm, we set the number of K as 1, 5 and 10 respectively, and compare the weighted sum of delay and energy consumption under the different total numbers of time slots. As shown in Fig. 3, when $K = 10$, the MCDDRL algorithm is superior to $K = 1$, 5 in terms of time delay and energy consumption weighted sum. The reason is that the more DNNs there are, the more offloading solutions are generated, and the easier it is to find a solution that is close to the optimal solution.

In order to verify the performance of the proposed algorithm, the advantages and disadvantages of the MCDDRL algorithm can be evaluated by comparing the weighted sum of energy consumption and delay, energy consumption and delay of MCDDRL with other offloading algorithms under different time frames number. Comparative offloading algorithms are DDLO [29] (a distributed algorithm using binary strategy), DQN (Deep Q Network, a reinforcement learning algorithm), random offloading algorithm (ROA), greedy algorithm (It will enumerate all possible schemes and select the weighted sum of the smallest scheme.) and local computing (All tasks are computed in local.).

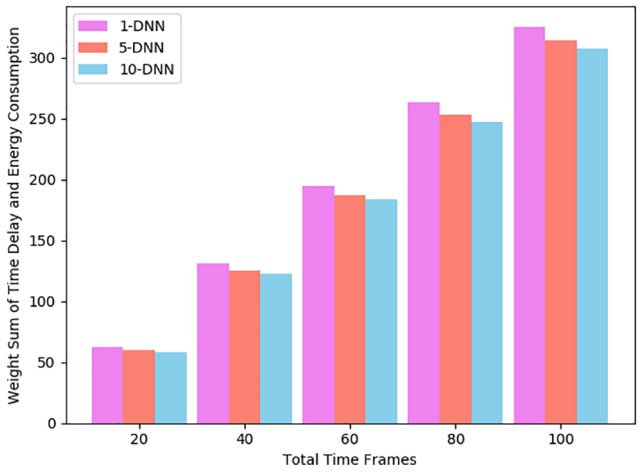

Figure 3: Weight sum of time delay and energy consumption under different time frames number for different DNN number

In this paper, we compare the Time and Energy Weight Sum Reduction Ratio (TEWRR) of MCDDRL, DDLO, ROA and DQN, as shown in Tab. 2. TEWRR represents the value of the difference of the weighted sum of the delay and energy consumption between the above algorithm and local computing and then divided by the weighted sum of local computing. MCDDRL has the highest TEWRR among all algorithms, reaching more than 50%, which is better than DDLO DQN and ROA.

Table 2: TEWRR for different algorithms

Algorithm	TEWRR
ROA	14.41%
DDLO	42.47%
MCDDRL	53.24%
DQN	51.20%

We compared the performance of local computing, ROA, greedy algorithm, DDLO, DQN and MCDDRL in terms of the weighted sum of delay and energy consumption, time delay and energy consumption. The value range of total time slot is [0, 400], with 20 as the step size.

In Figs. 4–6, the time delay, energy consumption and the weighted sum of the two factors show an upward trend with the increase of the number of total time slots, and tasks generated by the MEC system. MCDDRL performs better than DDLO and DQN in the above three aspects and is slightly lower than the greedy algorithm. However, the greedy algorithm is very complicated and time-consuming, so it is not suitable for intelligent the inspection MEC system. At the same time, we found that the energy consumption of the ROA algorithm was lower than that of all other algorithms. Because the amount of task data to be offloading is small result in the total energy consumption of the system is small. Meanwhile, the time delay is far greater than the energy consumption. Therefore, the weighted sum mainly shows the trend of time delay, when α and β are the same. And we can change the dominant position of energy consumption and time delay in the system by adjusting α and β.

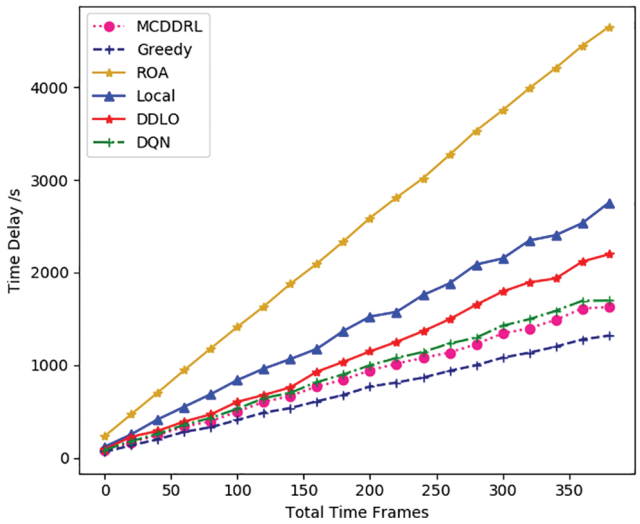

Figure 4: Time delay under different time frames number for different algorithms

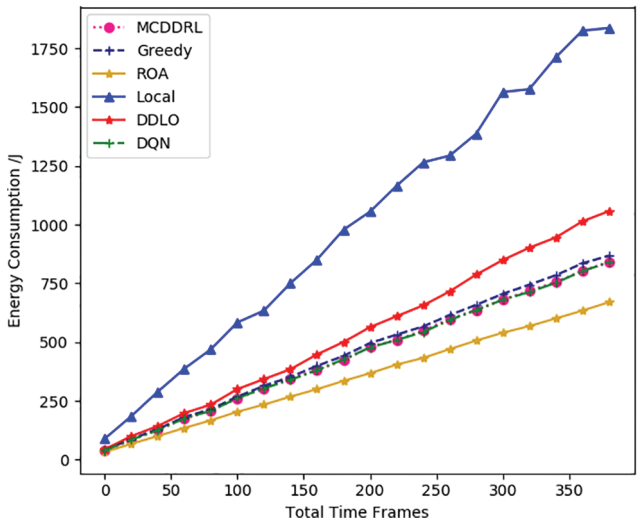

Figure 5: Energy consumption under different time frames number for different algorithms

In Figs. 7–9, we study the average TEWRR of different edge computing nodes $W = [2, 5]$, inspection devices $N = [2, 5]$ and task number of every device $I = [2, 5]$ under different total time slots. The value range of total time slot is [0, 1600], with 200 as the step size. For MCDDRL, the average value of TEWRR tends to be stable as the total number of time slots increases, no matter W, N, I take any value in the range [2, 5]. In addition, when N and I are fixed, the average value of TEWRR increases as the edge computing node W increases within the range of the same total number of time slots based on the results of Figs. 7 and 10. The reason is that the total number of tasks that need to be processed remains the same in the MEC system, but an increase in W will increase computing resources available in the system, resulting in a positive correlation between the average value of TEWRR and the size of edge computing nodes W. On

the contrary, combining the results of Figs. 9 and 10 shows that when W and N are fixed, the average value of TEWRR is inversely proportional to inspection devices I. Because in this case, the available computing resources of the MEC system remain unchanged, and an increase in I will increase the total number of tasks to be processed in each time slot.

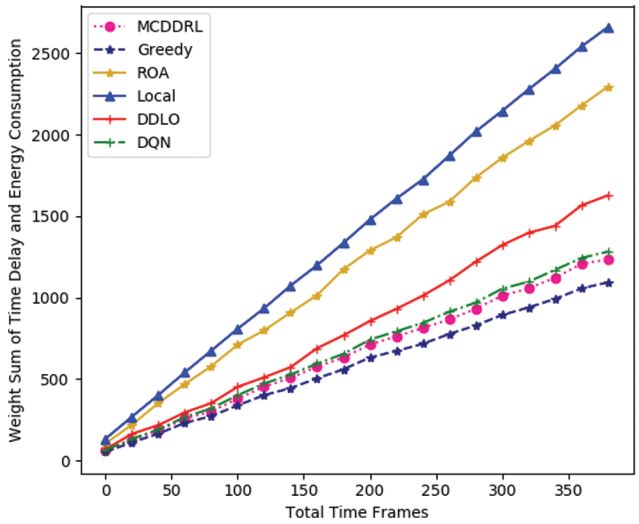

Figure 6: Weight sum of time delay and energy consumption under different time frames number for different algorithms

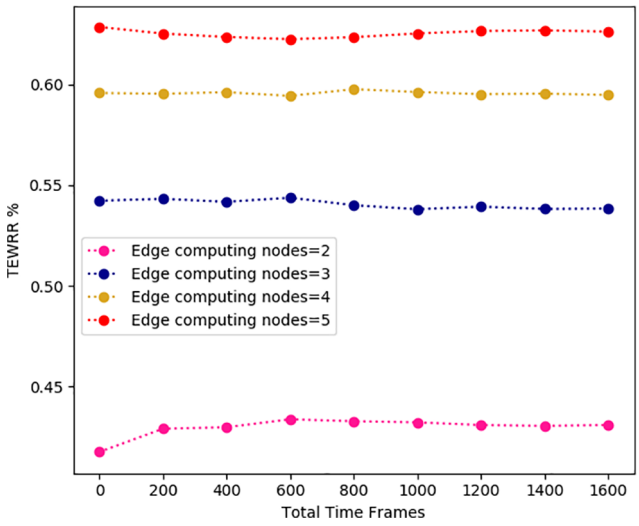

Figure 7: The average TEWRR of different edge computing nodes W under different total time slots

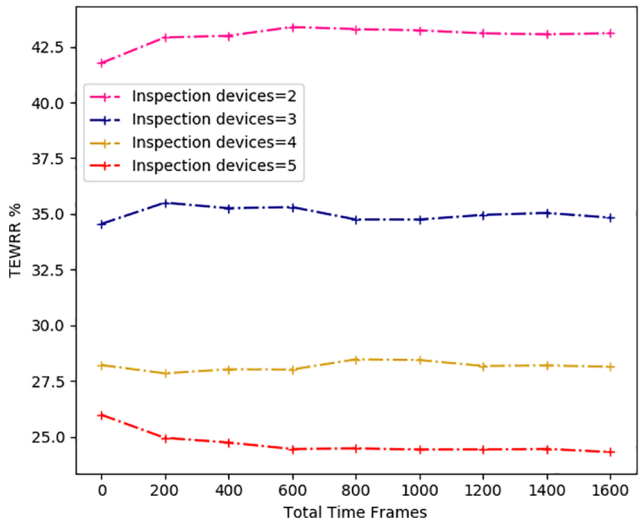

Figure 8: The average TEWRR of different inspection devices N under different total time slots

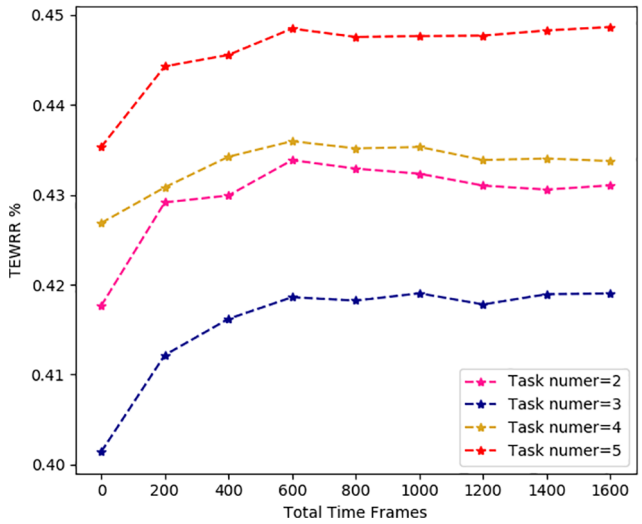

Figure 9: The average TEWRR of different task number of every device I under different total time slots

Finally, combining Figs. 8 and 10, the correlation between the average value of TEWRR and inspection devices N is not obvious in the case of determining W and I, and the range of change is relatively small. The reason is that the total number of computing resources and tasks to be processed in each time slot in the MEC system will increase with the increase of N. At this time, the value of TEWRR is related to the task offloading strategy. The more similar the MCDDRL's offloading strategy is to the local computing, the smaller the value of TEWRR.

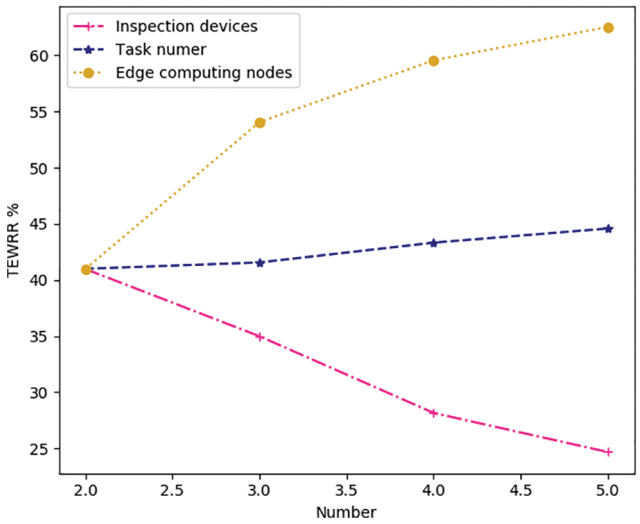

Figure 10: The average TEWRR of different factor under different number in range [2, 5], step size = 1

6 Conclusion

In this paper, we propose a distributed algorithm MCDDRL based on reinforcement learning and multi-classification ideas to minimize the task delay and total energy consumption, and cooperate to solve the task offloading and resource allocation problems in the intelligent inspection cluster MEC network of multi-task multi-inspection devices multi-edge nodes. We transformed the offloading problem of a single task into a multi-classification problem and simultaneously used multiple DNNs to learn the offloading decision from past data. At the expense of the interests of part of the task, we gradually iterative to find the closest solution. Simulation results show that MCDDRL is superior to other baseline algorithms, and it can effectively reduce the energy consumption and delay of the MEC system. In the future work, we will consider the task execution with time limit, priority and order of the intelligent patrol MEC system, as well as the offloading decision and resource allocation problems under wireless interference.

Acknowledgement: We would like to thank all those who participated in the completion of the experiment and this paper.

Funding Statement: This paper is funded by the National Research & Development Project (No. 2018YFC15007005) and Sichuan Research & Development Project (No. 2020YFG0189).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] X. Chen, T. Wu, G. Sun and H. Yu, "Software-defined MANET swarm for mobile monitoring in hydropower plants," *IEEE Access*, vol. 7, pp. 152243–152257, 2019.

[2] C. Esposito, A. Castiglione, F. Pop and K. R. Choo, "Challenges of connecting edge and cloud computing: A security and forensic perspective," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 13–17, 2017.

[3] K. Li, X. Tang and K. Li, "Energy-efficient stochastic task scheduling on heterogeneous computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 2867–2876, 2013.

[4] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng *et al.,* "Consortium blockchain for secure energy trading in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3690–3700, 2017.

[5] D. Sabella, V. Sukhomlinov, L. Trang, S. Kekki, P. Paglierani *et al.,* "Developing software for multi-access edge computing," *ETSI White Paper*, vol. 20, pp. 1–38, 2019.

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

[7] T. Q. Dinh, J. Tang, Q. D. La and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[8] L. Tong, Y. Li and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. IEEE International Conference on Computer Communications*, San Francisco, SF, USA, vol. 2016, pp. 1–9, 2016.

[9] X. Lin, Y. Wang, Q. Xie and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 175–186, 2014.

[10] L. Huang, X. Feng, A. Feng, Y. Huang and L. P. Qian, "Distributed deep learning-based offloading for mobile edge computing networks," *Mobile Networks and Applications*, vol. 23, pp. 1–8, 2018.

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

[12] Y. Mao, J. Zhang and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[13] X. Chen, L. Jiao, W. Li and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[14] S. Long, W. Long, Z. Li, K. Li, Y. Xia *et al.,* "A game-based approach for cost-aware task assignment with QoS constraint in collaborative edge and cloud environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1629–1640, 2020.

[15] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4514–4526, 2018.

[16] G. Mitsis, P. A. Apostolopoulos, E. E. Tsiropoulou and S. Papavassiliou, "Intelligent dynamic data offloading in a competitive mobile edge computing market," *Future Internet*, vol. 11, no. 5, pp. 118, 2019.

[17] Y. Li, X. Wang, W. Fang, F. Xue, H. Jin *et al.,* "A distributed admm approach for collaborative regression learning in edge computing," *Computers, Materials & Continua*, vol. 23, no. 2, pp. 493–508, 2019.

[18] T. Yang, X. Shi, Y. Li, B. Huang, H. Xie *et al.,* "Workload allocation based on user mobility in mobile edge computing," *Journal on Big Data*, vol. 2, no. 3, pp. 105–111, 2020.

[19] Y. Wei, Z. Wang, D. Guo and F. R. Yu, "Deep q-learning based computation offloading strategy for mobile edge computing," *Computers, Materials & Continua*, vol. 59, no. 1, pp. 89–104, 2019.

[20] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji *et al.,* "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2018.

[21] X. Xiong, K. Zheng, L. Lei and L. Hou, "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1133–1146, 2020.

[22] J. Ren, H. Wang, T. Hou, S. Zheng and C. Tang, "Collaborative edge computing and caching with deep reinforcement learning decision agents," *IEEE Access*, vol. 8, pp. 120604–120612, 2020.

[23] Y. Chen, Z. Liu, Y. Zhang, Y. Wu, X. Chen *et al.,* "Deep reinforcement learning-based dynamic resource management for mobile edge computing in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4925–4934, 2020.

[24] Y. Qian, R. Wang, J. Wu, B. Tan and H. Ren, "Reinforcement learning-based optimal computing and caching in mobile edge network," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2343–2355, 2020.

[25] Z. Tong, X. Deng, F. Ye, S. Basodi, X. Xiao *et al.,* "Adaptive computation offloading and resource allocation strategy in a mobile edge computing environment," *Information Sciences*, vol. 537, pp. 116–131, 2020.

[26] L. Huang, S. Bi and Y. -J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2019.

[27] Y. Mao, J. Zhang, S. H. Song and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *Proc. IEEE Global Communication Conf.*, Washington, WA, USA, pp. 1–6, 2016.

[28] Y. Mao, J. Zhang and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Proc. IEEE Wireless Communications & Networking Conf.*, San Francisco, SF, USA, pp. 1–6, 2017.

[29] L. Huang, X. Feng, L. Zhang, L. Qian and L. Wu, "Multi-server multi-user multi-task computation offloading for mobile edge computing networks," *Sensors*, vol. 19, no. 6, pp. 1446, 2019.