PAPER

A Joint Coverage Constrained Task Offloading and Resource Allocation Method in MEC

Daxiu ZHANG[†a)], **Xianwei LI**[††,†††b)], ***Nonmembers***, **Bo WEI**[††††,†††††c)], ***Member***, ***and*** **Yukun SHI**[††††††d)], ***Nonmember***

SUMMARY With the increase of the number of Mobile User Equipments (MUEs), numerous tasks that with high requirements of resources are generated. However, the MUEs have limited computational resources, computing power and storage space. In this paper, a joint coverage constrained task offloading and resource allocation method based on deep reinforcement learning is proposed. The aim is offloading the tasks that cannot be processed locally to the edge servers to alleviate the conflict between the resource constraints of MUEs and the high performance task processing. The studied problem considers the dynamic variability and complexity of the system model, coverage, offloading decisions, communication relationships and resource constraints. An entropy weight method is used to optimize the resource allocation process and balance the energy consumption and execution time. The results of the study show that the number of tasks and MUEs affects the execution time and energy consumption of the task offloading and resource allocation processes in the interest of the service provider, and enhances the user experience.

key words: *Mobile Edge Computing, coverage constraint, deep reinforcement learning, task offloading, resource allocation*

1. Introduction

With the rapid development of IoT communication technologies [1], the Mobile User Equipment (MUE) is everywhere in people's daily life and can be used for sensing, computing and communication. MUEs will generate numerous data tasks that with high requirements in terms of computational resources, computing power and battery life. Due to the limited computational resources, computing power and storage space of MUEs, the data tasks can be offloaded to data center in public clouds for processing [2]. However, the distance of public cloud data centers is far from MUEs in traditional network systems leading to latency [3] and security issues [4] for some real-time processing applications, which impacts the mobile user experience and reduces the potential benefits of the public cloud [5].

To overcome these problems, Mobile Edge Computing (MEC) is considered as a promising technology [6]. MEC allows MEUs to offload tasks to the edge cloud for processing, relieving congested waiting in the data center public cloud [7]. However, it is difficult for MUEs to select the appropriate computational node based on the resource requirements of the task. Considerable efforts have been devoted to employing Deep Reinforcement Learning (DRL) algorithm to study task offloading and resource allocation in the MEC system [8], [9].

In this study, we apply the DRL joint task coverage constraint method in the MEC environment to offload tasks generated by MUEs to the appropriate MEC-S server for processing by jointly allocating computation, memory, CPU, and bandwidth resources. We use the DRL joint task coverage constraint method to select the appropriate computational nodes for the tasks in the current environment to complete the process of task offloading and resource allocation, which is described as MDP. This method optimizes the execution time and energy consumption during the task offloading and resource allocation processes in the MEC system, improving the quality of user's life.

The main contributions of this study are as follows.

• Task coverage constraints are considered when building the problem model to ensure that the tasks generated by MUEs can only be offloaded in the nodes within their sensing range, which improves the reliability of the task offloading process.

• This study proposes a task offloading and resource allocation method in the MEC system based on DRL, named Joint Task Coverage Constraint (JC-TORA), which considers the coverage constraints, adaptively selects appropriate computing nodes for the tasks, and improves the reliability of task offloading transmission.

• To obtain better experimental results, the interests between the user equipment and the server, the mobility of the equipment are considered, and the entropy weight and gradient descent methods are used to obtain the reward value of the computational environment for the next step of feedback learning to obtain the optimal TORA strategies.

Manuscript received November 5, 2023.
Manuscript revised January 24, 2024.
Manuscript publicized March 6, 2024.

[†]School of Information Technology, Quanzhou Vocational College of Economics and Business, Quanzhou, 362000, China.
[††]School of Computer and Information Engineering, Bengbu University, Bengbu, 233000 China.
[†††]Anhui Engineering Research Center for Intelligent Applications and Security of Industrial Internet, Anhui University of Technology, Ma'anshan, Anhui, 243032 China.
[††††]Department of Systems Innovation, School of Engineering, The University of Tokyo, Tokyo, 113-0033 Japan.
[†††††]Japan Science and Technology Agency (JST), PRESTO, Kawaguchi-shi, 332-0012 Japan.
[††††††]School of Cyberspace Security Academy, Hangzhou Dianzi University, Hangzhou, 310018 China.
a) E-mail: zhangdaxiu2021@163.com
b) E-mail: lixianwei163@163.com
c) E-mail: weibo@g.ecc.u-tokyo.ac.jp
d) E-mail: yukun.shi@hdu.edu.cn
DOI: 10.1587/transfun.2023EAP1139

2. Related Work

There are many approaches to study task offloading and resource allocation in multi-user and multi-server MEC systems, among which convex optimization and Lyapunov methods are commonly used to solve the task offloading and resource allocation problems [10]–[12]. This section reviews some related studies on task offloading and DRL in MEC systems.

At present, there are three main task offloading methods, namely, local computing, server computing and collaborative computing. If the local resources meet the demand of the task, the task is directly computed locally; If the local resources are insufficient, the task is offloaded to the nearest server node for computing. These two cases are collectively referred to as the binary offloading decisions [13]. Collaborative computing occurs when local resources are insufficient. Some tasks are processed locally, others are offloaded to the edge server [14]. In task offloading and resource allocation problems, the optimization objectives are divided into two main categories, that are single-objective problems that reduce latency and energy consumption, and multi-objective problems that balance latency and energy consumption.

Some related work only considered the minimization of either energy consumption or delay. Due to the limitation of power stability of local MUEs, Li used the divide-and-conquer strategy and integer linear programming algorithm based on Lyapunov optimization to solve the offload allocation, improve the CPU utilization of MEC server and reduce task latency [10]. When computational resources are limited, Feng proposed a vehicular edge-computing-based reverse offloading framework to further reduce system latency by making full use of vehicle computational resources [15]. Chen constructed an energy-saving resource allocation scheme while considering the constraints of delay, channel quality and transmission power, which aims to minimize the energy consumption of task offloading [16].

Some related work considered the trade-off between energy consumption and delay. To balance the energy consumption and delay for task offloading and resource allocated, Mao proposed an online task offloading algorithm to optimize the power consumption and task delay in MEC system, which improved the user's QoE [17]. But it didn't consider mobility and overall cost. Combined with the double deep Q-network, Tong built a novel task offloading algorithm (DDTMOA) with an integrated trust assessment mechanism to reduce the average task response time and total system energy consumption, effectively [18]. Elgendy proposed a joint task cache offload allocation method, which only considered the bandwidth, deadline and offload constraints, but didn't consider the dynamic movement of mobile devices to different BSs [19].

Different from previous works, in this study, we consider task offloading under the coverage, bandwidth, deadline, memory and CPU resource constraints. We develop a model that balances execution time and energy consumption and use the DRL method to update the policy to select the optimal compute node for the task.

3. System Model

The objective of this study is to optimize task offloading and resource allocation by combining task coverage to minimize total offloading and allocation costs, including energy, calculation and delay, which selects the appropriate computational nodes for each task and completes data transfer processing. Assuming that a MEC system contains multiple MUEs and multiple MEC-S servers, the MUEs are randomly deployed in the monitoring area to acquire and manage the task offloaded to the corresponding MEC-S servers for processing in Fig. 1.

In this study, the problem input is the number of MEC-S servers, BSs, MUEs and tasks, and the output is the optimal offloading and resource allocation strategies.

Definition: The notations j, m, u and n represent the number of MEC-S servers, BSs, MUEs and tasks, respectively. In addition, the MEC-S servers, BSs and MUEs are collectively referred to as node k. Table 1 lists the necessary symbols in JC-TORA.

MUE is a set of multi-user equipments that can be formalized as $U = \{u_1, u_2, \ldots, u_U\}$ whose scale is U. Each MUE generates N independent tasks, and it has an attribute set, represented by Task. Task = $\{ID_{ui}, A_i, D_{l_i}, l_i, B_i, MEM_i, CPU_i\}$, $i \in (1, 2, \ldots, \mathrm{N})$, where $l_i = A_i * CPI$. Each task has a submission time and a deadline, the tasks are queued according to the submission time, deadline and communication relationships, and is processed according to the first come first served rule. Due to the mobility, the MUE may be near different BSs at different times, so tasks generated by the MUE at different times may be offloaded to MEC-S servers on different BSs for execution. Each server has multiple task buffers that simultaneously store tasks offloaded by multiple users but not yet processed. But the CPU of each MEC-S server can be allocated to at most one user to perform processing tasks. The binary variable x_{ki}=1, indicating that the i-th task is processed and offloaded to edge MEC-S server, otherwise it is processed locally.

The specific workflow of the system model is:

STEP 1: u_u (u=1,2,...,U) generates N tasks, and it has at-

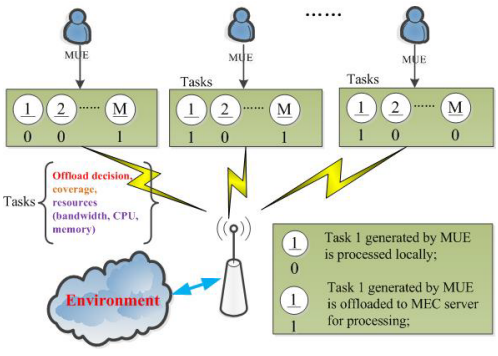


Fig. 1 The MEC system model.

Table 1 Common term symbols.

Symbol	Description
ID_{ui}	The ID of MUE u generate task i
A_i	The task data size (bit) of task i
D_{l_i}	Latest deadline of task i
l_i	the total CPU cycles of task i
CPI	CPU cycles to compute 1-bit data of task
B_i	Bandwidth resources required for task i calculation (bit/s)
MEM_i	Memory resources required for task i calculation
CPU_i	CPU resources required for task i calculation
x_{ki}	Offloading decision of task i
e^{loc}	Energy consumption 1-bit data processed locally
C_u^{loc}	Computation capability of each MUE (CPU cycles/s)
E_{tot}^{loc}	Total energy consumption of task local computing
T_{ui}^{loc}	Time cost of n tasks of MUE u
T_{tot}^{loc}	Total time for task local computing
$R_{u,m}$	Communication rate of MUE u to BS m
P_u	Transmit power of MUE u
N_0	Noise power spectral density of BS m
$I_{u,m}$	Channel gain between MUE u and BS m
q_{mj}	Energy consumption of each bit task data calculated by server j
F_{mj}	CPU frequency of server j
E_{ui}^{off}	Energy consumption of MUE u offloading task i to server j via BS m
E_{tot}^{off}	Total energy consumption for task offloading to MEC-S server
T_{tot}^{off}	The total time for task offloading to MEC-S server
T^{off}	Total time of the system
E^{off}	Total energy consumption of the system
$y_{i,k}$	The relationship between task i and computational nodes k
$lengtn_i$	Scheduling length of i
γ_{ig}	The relationship between effective perception rate f_{ig}^e and perception rate f_{iug}
δ_i	Coverage of task i

tributes such as generation time, deadline processing time and task data A_i.

STEP 2: u_u estimates the calculation and processing methods of task i based on its scheduling and demand, etc. If the local resource meets the request of task i, goes to STEP 3. Otherwise, goes to STEP 4.

STEP 3: Task i completes the calculation on the local node MUE or BS.

STEP 4: Task i completes the calculation on the MEC-S server node.

STEP 5: The calculation result is returned to u_u.

3.1 Local Computing Model

Since the local MUE has a certain computational capacity to handle appropriate task requests. If the local resources satisfy the request of task i and the computation processing is done directly locally, the time of local computation of task i on MUE u is denoted as.

$$T_{ui}^{loc} = \sum_{i=1}^{N} \frac{l_i}{C_u^{loc}}, \quad i \in (1, 2, \ldots, N) \tag{1}$$

The total time for local processing tasks is defined as Eq. (2).

$$T_{tot}^{loc} = \sum_{k=1}^{K} \sum_{i=1}^{N} T_{ui}^{loc}(1 - x_{ki}) \tag{2}$$

The energy consumption of the task at the local nodes is

$$E_{tot}^{loc} = \sum_{u=1}^{U} \sum_{i=1}^{N} A_i e^{loc}(1 - x_{ki}) \tag{3}$$

3.2 MEC-S Server-Computing Model

Because of the limited local computing resources, some tasks will be offloaded to the MEC-S server for processing. That is divided into two processes: offloading transmission and calculation processing. $R_{u,m}$ [18] is used to calculate energy consumption and time.

$$R_{u,m} = B \log_2(1 + \frac{P_u I_{u,m}}{N_0 B}) \tag{4}$$

where B is the communication bandwidth between MUE u and BS m, $I_{u,m}$ is a random independent identically distributed variable.

MEC-S server task offloading energy consumption includes both offloading energy consumption and calculation energy consumption. The energy consumption of MUE u offloading task i to server j through BS m is:

$$E_{ui}^{off} = P_u \frac{A_i}{R_{u,m}} + A_i \times q_{mj} \tag{5}$$

Therefore, combined with the offloading strategy x_{ki}, the total energy consumption of tasks generated by MUE u offloaded to server j through BS m is:

$$E_{tot}^{off} = \sum_{u=1}^{U} \sum_{i=1}^{N} x_{ki} E_{ui}^{off} \tag{6}$$

Similarly, the time cost of task offloading to MEC-S server is the sum of transmission time and computing time; Therefore, the total time cost of task i generated by MUE u being offloaded to server j through BS m is:

$$T_{tot}^{off} = \sum_{u=1}^{U} \sum_{i=1}^{N} x_{ki}(\frac{A_i}{R_{u,m}} + \frac{l_i}{F_{mj}}) \tag{7}$$

4. Problem Formulation

To ensure minimal execution time and energy consumption, some tasks are processed locally while others are offloaded to the MEC-S server. In this process, the offload transfer time is ignored due to the high transfer rate of the MEC-S server and the returned computation results. If the tasks are computed locally, the communication time is 0. The time

required to complete all tasks is:

$$T_{total} = T_{tot}^{loc} + T_{tot}^{off} \tag{8}$$

Similarly, the total energy consumed to complete all tasks is:

$$E_{total} = E_{tot}^{loc} + E_{tot}^{off} \tag{9}$$

The optimization objective of the JC-TORA problem is to minimize the total execution time and energy consumption of all the tasks by selecting the appropriate computational node for each task. The JC-TORA problem is defined as follows:

$$min \sum_{u=1}^{U} [\omega_1 E_{total} + \omega_2 T_{total}] \tag{10}$$

$$s.t \qquad x_{ki} = \begin{cases} 1, & \forall k \in J, \forall i \in N \\ 0, & \forall k \in U, M, \forall i \in N \end{cases} \tag{11}$$

$$y_{i,k} \leq 1, i \in N, k \in (U, J, M), \tag{12}$$

$$lengtn_{(i)} \leq D_{l_i}, \tag{13}$$

$$\frac{\sum_{g \in G_i} \gamma_{ig}}{\mid G_i \mid} \geq \delta_i, \forall i \in N, \tag{14}$$

$$\sum_{u=1}^{U} \sum_{i=1}^{N} MEM_i \leq MEM_{max}, \tag{15}$$

$$\sum_{u=1}^{U} \sum_{i=1}^{N} CPU_i \leq CPU_{max}, \tag{16}$$

$$\sum_{u=1}^{U} \sum_{i=1}^{N} B_i \leq B_{max}. \tag{17}$$

Equation (10) is objective function to minimize the total energy consumption and time cost, where ω_1 and ω_2 the weight parameters of energy consumption and time, which is allocated using the entropy weight method of the objective weighting method.

Equation (11) and Eq. (12) represent the offloading and resource allocation constraint for task nodes. Equation (12) constrains that tasks must be allocated to nodes and that nodes can only allocate one MUE to perform tasks at a time.

Equation (13) is used to limit the tasks that must be completed by the deadline. $lengtn_{(i)}$ is the scheduling length of task i.

Equation (14) represents the coverage constraint with a perceptual metric [21]. $g \in G_i$ denotes the perceptual range (computational node) of the MUE u that generates task i, which can only be detected if it is within the perceptual range of task i. That is, using $v_{iug} = 1$ indicates that the perceived range G_i of task i is within the monitored range of the corresponding MUE u; otherwise it is not. In addition, the accuracy of the monitoring rate must be guaranteed in the monitored object. The target perception rate f_{iug} is assigned a value greater than 0 and less than $y_{ik} * v_{iug} * f_i$ to indicate that the node k offloaded by the task i generated by the MUE u is within the range g of its perceived target, where f_i is the minimum-sampling rate. Then, the cooperative effective perception f_{ig}^e is given as follows:

$$f_{ig}^e = \frac{\sum_{u \in U} f_{iug}}{1 + T_{ui} \sum_{u \in U} f_{iug}}, \forall i \in N, u \in U, g \in G_i \tag{18}$$

T_{ui} is the execution time. When the cooperative effective perception rate of task i at target g is greater than or equal to the min-sampling rate, $\gamma_{ig} = 1$, otherwise 0.

Equation (15), Eq. (16) and Eq. (17) are used to limit resource requirements to no more than the maximum value.

5. JC-TORA Algorithm

5.1 Theoretical Background

At present, the DRL method is often used to solve the Task Offloading and Resource Allocation (TORA). DQN is a specific algorithm in the DRL method, which combines convolutional neural network (CNN) and Q-Learning. The input of CNN is the original environment data (as state), and the output is the Value Function (Q-value) corresponding to each action. The agent guides the actions to obtain optimal rewards by updating the policy and observing the state of the environment, as shown in Fig. 2.

QL stores the state, action and reward in the memory unit and repeatedly learns and trains the update value network (Q-value) until convergence, obtaining a better action. The Q-value is the expected reward for following a policy to perform a particular action in a given state. A policy is a reflection list that demonstrates how the state space S maps to the action space A, i.e., $\pi(s) : S \rightarrow A$. Therefore, the optimal policy is a policy $\pi^{\theta}(s, a)$ that maximizes the action-value function from each state in MDP.

$$\pi^{\theta}(s, a) = \underset{a'}{argmax} Q^{\pi}(s, a). \tag{19}$$

The agent selects an action based on the Q-value to make the current optimal decision. Then the mean-square error is used as the Loss function to calculate the difference in loss between these values and the gradient descent algorithm is used to minimize this loss.

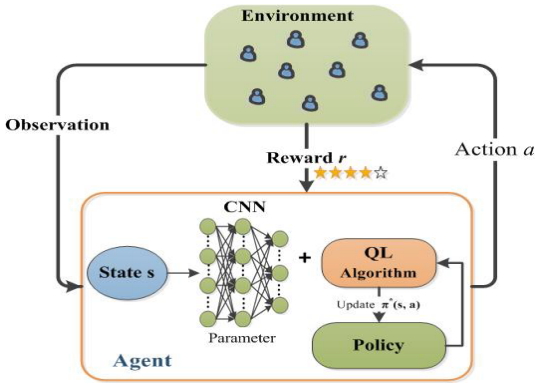


Fig. 2 The DRL framework.

$$Loss = (r + \gamma max Q_t(s^{'}, a^{'}, \theta^{'}) - Q_t(s, a, \theta))^2, \quad (20)$$

After the target network is frozen, experience playback is introduced to make the training process more stable. The method stores a batch of sample records (s, a, r, s') in the experience pool, randomly selects samples for training DQN algorithm to accurately fit the Q-values of different states and actions. Ultimately, the process of getting the Max reward r through continuous environmental learning training is the optimal policy.

5.2 JC-TORA Algorithm Key Elements

The optimization problem of TORA in the MEC system is a typical mixed 0-1 nonlinear programming problem that is NP-hard problem and is difficult to solve. We reformulate it as an MDP, and it can be solved using the DRL method. MDP is represented using a four tuple (s, a, r, s'), where s is the current environment state, a is the action, r is the reward value, and s' is the new environment state.

State space s: Since the neural network can only have one input value (at a time), the optimization objective is weighted here as a state in order to simulate the realistic scenario. The state space is a weighted sum of task execution time and energy consumption on each computational node, defined as Eq. (21) after the weight entropy method and max-min normalization. The main considerations are the resource, idle and perceived status of the computational nodes as well as the attributes and communication relationships of the tasks.

$$s = (s_1, s_2, \ldots, s_p), s_i = \lambda_1 T_{ui}^{tot} + \lambda_2 E_{ui}^{tot}. \quad (21)$$

Action space a**:** In this MEC environment, the selection of computational nodes is taken as the action selection, where computational node refers to the MUE that generates the task and servers on the BS where the MUE is located. Action space can be expressed as follows:

$$a = (UE, server_1, server_2, \ldots, server_j), \quad (22)$$

The selected computational node's action value is 1 and others are 0. Assuming task i is processed locally, $a = (1, 0, \ldots, 0)$.

Reward function r**:** Select an action a within the coverage of task i in the current environment state s. If the task is allocated to the MEC-S server for processing, it is offloaded; Otherwise, it is processed locally. The system environment feedback rewards to the agent based on the quality of the selected action. In general, the higher the value of the reward, the better the action selection. According to the optimization objective of the problem model, the reward function r is designed as the inverse of the weighted sum of time and energy consumption as in Eq. (23), i.e., the larger the reward value r, the smaller the time and energy consumption of the task offloading and resource allocation process.

$$r = \frac{1}{\lambda_1 T_{ui}^{tot} + \lambda_2 E_{ui}^{tot}} \quad (23)$$

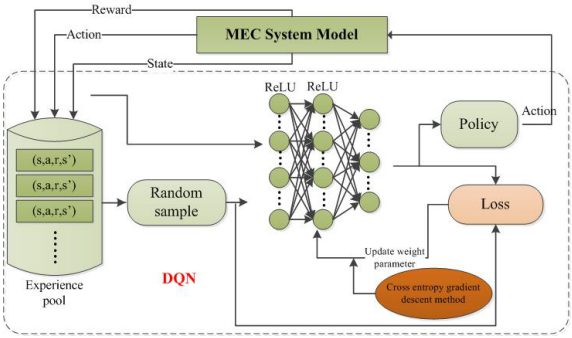


Fig. 3 The DQN algorithm for the MEC system.

Discount factor γ**:** The discount factor $\gamma \in [0, 1]$, when $\gamma \to 1$, the agent prefers the long-term reward, otherwise focuses on the immediate benefit.

5.3 JC-TORA Algorithm Design

In the MEC system, the JC-TORA algorithm is used to find the optimal offload and resource allocation policy for MUE's task within its coverage as shown in Fig. 3. The task acts as an agent in the DRL framework and receives observations and rewards from the MEC environment during each time slot. The agent learns offloading and allocation policies based on the task execution time and energy consumption, and realizes dynamic routing by guiding the task to select the next action (computational node) through feedback rewards. After several training sessions, the action state with maximum reward is selected.

In the process of TORA, the entropy weight method in objective weighting method is used to optimize the objective and reduce bias. The whole process of TORA in MEC system using the DQN is shown in Algorithm 1, which includes replay memory size, discount factor γ and learning rate α. For each episode, the agent selects the action a by the optimal policy. When the agent selects action a, the reward r is computed by Eq. (23). With the step into the next state s_{t+1}, the matrix of transition (s, a, r, s') is stored and the mini-batch size of transition (s, a, r, s') is sampled randomly in the memory pool. Gradient descent and cross-entropy are used to optimize the Loss function and update the weight parameter. Finally, the target Q-network is reset at the end of each episode. The process is an episode from the initial state to a termination state, which is a completion sequence consisting of state sets, action sets and rewards. The JC-TORA algorithm is used to fit the environment states and makes a reasonable decision for each state based on RL to choose a reasonable action. Tasks generated by MUE are processed locally or MEC-S server by policy, and a reward value is obtained to guide the agent learning to explore in the direction of the max reward, which improves the effectiveness of the objective.

Algorithm 1 JC-TORA Algorithm

Input: MUE (Poisson distribution), γ, α
Output: Q-network Q(s,a)

Initialize memory pool, capacity size, MEC environment;
Initialize Q-network and target Q-network with weight λ in DRL model;
Initialize parameter setting;
repeat
    Determine the BS location of the MUE based on the tasks generated by the MUE;
    Obtain the current experimental environment;
    Initialize the start state task¡¯s time and energy consumption is 0;
    In the current environment, DQN algorithm is used to select computing node for task i (select action a) by the optimal policy;
    if the selected computing node is MUE or BS **then**
        Task i is executed locally;
    else
        Task i is offloaded to the MEC-S server corresponding to the BS for execution;
        Calculate the time when task i is offloaded to MEC-S server;
    end if
    Calculate reward r based on Eq. (23);
    Calculate the execution time and energy consumption s' of task i;
    (s, a, r, s') is stored in the memory pool;
    Select sample (s, a, r, s') from the memory pool, calculate $r(s,a) + \gamma max_{a'} Q_t(s', a', \theta')$ to evaluate Q-network;
    Gradient descent method and cross-entropy method are used to minimize the Loss function;
    Update Weights and Q-network;
until $\forall s, a$, $Q_t(s,a)$ convergence;
return The optimal task offloading policy $\pi^{\theta}(s,a)$;
end

6. Experimental and Analysis

6.1 Experimental Settings and Comparison Algorithms

This study uses Python 3.6 and Tensorflow1.14 as experiment platform on Windows 10 system to evaluate the performance of the JC-TORA algorithm in various experimental settings. The MUE is deployed in a specific area based on the Poisson distribution, and the experimental parameters in Table 2 were determined through repeated experiments using the Monte Carlo methods. During the parameter setting process, the discount factor γ influences the trend of TORA policies in Fig. 4. To keep the optimal objective, the discount factor is 0.8. Then a comparison experiment is conducted.

• QL algorithms has strong learning effects and is commonly used to solve the TORA problem.

• Local algorithms execute all tasks locally, resulting in insufficient local resources and excessive computational pressure that consumes a lot of energy and time.

• The greedy algorithm enumerates all offloading decision and selects the best result. However, enumerating options takes time, especially for large-scale tasks.

Table 2 Experimental parameters.

Parameter	Parameter
MUE	10, 20, 40, 50, 60, 80, 100
BS	4, 8, 16, 32, 64, 128
CPI(cycle/bit)	500
B_{max}(Mbps)	150
MEM_{max}(GB)	16
CPU_{max}	100
e^{loc}(J/bit)	3.25×10^{-7}
C_u^{loc}(GHZ)	Unif (0.5, 1)
P_u(mW)	100
N_0(dBm/Hz)	-174
q_{mj}(J/bit)	Unif (10^{-8}, 2×10^{-8})
F_{mj}(GHz)	Unif (5, 10)
δ_i	0.78
Activation Function	ReLu
Loss Function	Mean-square error
Mini-Batch size	32
Optimizer	Adam
Experience pool capacity	1000
Learning rate α	10^{-4}
Discount factorγ	0.8

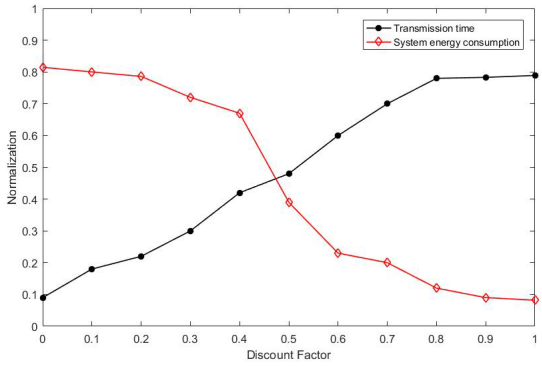


Fig. 4 The influence of discount factor on allocation policy.

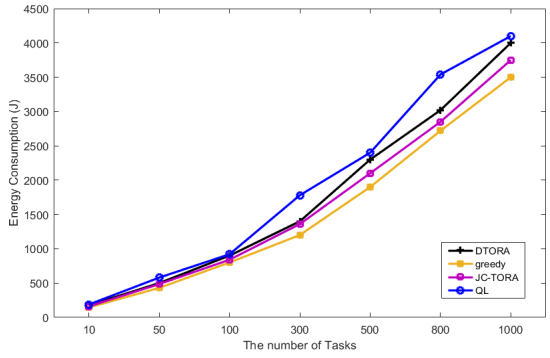
• DTORA algorithm [9] uses DRL method to optimize time and energy consumption based on memory and CPU resource, but the resource constraints are partial.

6.2 Experimental Results and Analysis

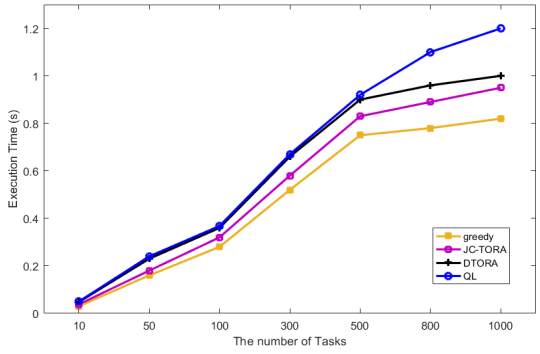
Comparative experiments were designed in terms of energy consumption, time under different experimental environments, including the number of task data size, MUEs and BSs.

6.2.1 Task Data Size

The first group is shown in Fig. 5, which shows the changes in energy consumption, execution time as the number of tasks increases, respectively. In Fig. 5(a), as the number of tasks increases, the energy consumption of each algorithm increases. This is because there are more and more data tasks that are offloaded to the computational nodes. As the transmission channel and energy consumption increases, the execution time also increases in Fig. 5(b).

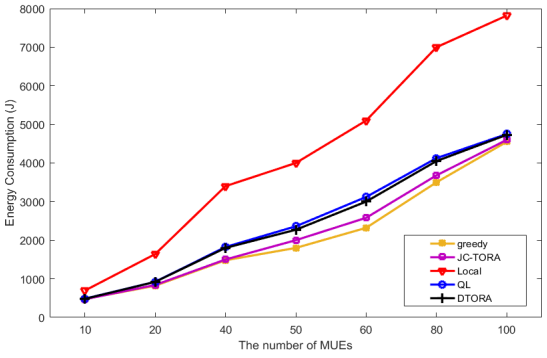


(a) Energy Consumption

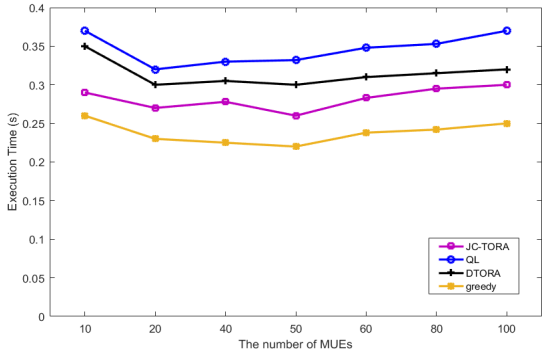


(b) Execution Time

Fig. 5 The performance under different tasks.



(a) Energy Consumption



(b) Execution Time

Fig. 6 The performance under different MUEs.

In Fig. 5, the greedy algorithm shows the optimal results, but at a high cost. The experimental results of the JC-TORA, DTORA and QL are superior, due to their ability to make adaptive learning decisions. However, JC-TORA is based on the adaptive DTORA, considers bandwidth resources and task coverage to complete task offloading. Therefore, JC-TORA outperforms the DTORA in all aspects. Since DQN algorithm combinates QL and RL methods, so DTORA algorithm has better learning effect than QL algorithm. Local processing consumes the most time and energy because of limited local computing resources and capacity.

6.2.2 The Number of MUEs

In Fig. 6, the second group of experiments examines the energy consumption, execution time under different MUEs. In Fig. 6(a) and Fig. 6(b), the energy consumption increases with the number of MUEs, and the execution time of each algorithm remains within a range. This is because increasing MUE is increase of tasks, which directly affects system energy consumption; But the tasks generated by the MUE in the same time period are almost constant and the task queue scheduling is less affected by the MUE, so the effect on the execution time is not significant.

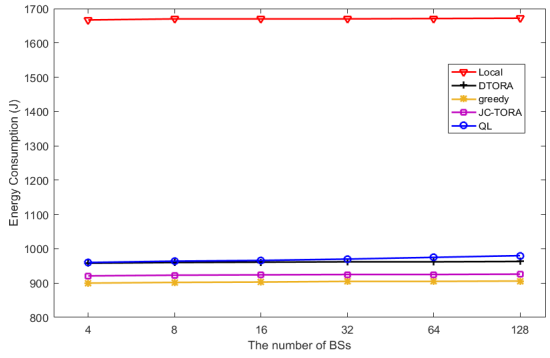
The greedy algorithm is the best solution in a complex environment. In this study, compared with DTORA, the JC-TORA considers bandwidth, task coverage, and other multi-aspect factors in the TORA process. As a result, the JC-TORA algorithm outperforms DTORA and QL algorithm and is close to the greedy algorithm.

6.2.3 The Number of BSs

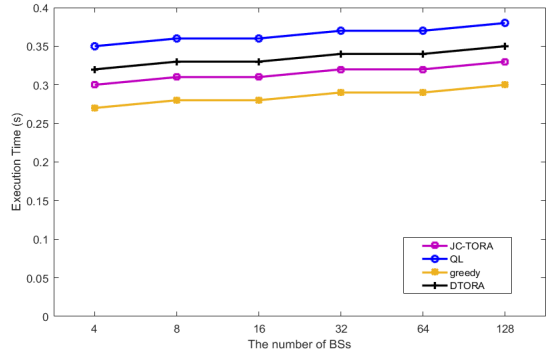
The third group analyzes the effect of different BSs. Because the number of BSs affects the computing power of the MEC system, i.e., the more BSs, the more MEC-S servers, and the more widely the BSs are distributed. The experimental results under different BSs is shown in Fig. 7, which show that the changes are smooth and insignificant as the BSs increases, because the computing resources required for the task are certain under the corresponding number of BSs. Because the number of BSs doesn't affect the task when computational resources are sufficient, the optimization order of algorithm performance for different numbers of BSs is: the greedy, JC-TORA, DTORA, QL and local algorithm.

7. Conclusions and Future Works

In this study, we propose the JC-TORA algorithm to solve the TORA problem in multi-MUE and multi-server MEC environments. The focus of this work is to use DQN to select the appropriate compute node for the user's task, and complete the reliable transmission by adding the appropriate constraints. In the future, the next work intends to conduct



(a) Energy Consumption



(b) Execution Time

Fig. 7 The performance under different BSs.

research in the area of reliable, efficient and secure TORA to improve the security and reliability of task transmission while balancing the computational performance.

Acknowledgments

This work was supported in part by Key Scientific Research Projects of Colleges and Universities in Anhui Province (2022AH051921), Funding project for the cultivation of outstanding talents in Colleges and Universities (gxyqZD2021135), Start Up funds for scientific research of high-level talents of Bengbu University (BBXY2020KYQD02), the Key research and development projects in Anhui Province (202004a05020043), the Key Scientific Research Projects of Anhui Provincial Department of Education (2022AH051376) and Natural Science Foundation Project of Bengbu University, China (2020ZR12).

References

[1] N. Mast, M.A. Khan, M.I. Uddin, S.A. Ali Shah, A. Khan, M.A. Al-Khasawneh, and M. Mahmoud, "Channel contention-based routing protocol for wireless ad hoc networks," Complexity, vol.2021, pp.1–10, 2021.

[2] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," IEEE Internet Things J., vol.5, no.1, pp.450–465, 2018.

[3] J. Xue and X. Guan, "Collaborative computation offloading and resource allocation based on dynamic pricing in mobile edge computing," Computer Communications, vol.198, pp.52–62, 2023.

[4] Z. Li, V. Chang, H. Hu, D. Yu, J. Ge, and B. Huang, "Profit maximization for security-aware task offloading in edge-cloud environment," Journal of Parallel and Distributed Computing, vol.157, pp.43–55, 2021.

[5] V. Cardellini, V.D.N. Persone, V.D. Valerio, F. Facchinei, V. Grassi, F.L. Presti, and V. Piccialli, "A game-theoretic approach to computation offloading in mobile cloud computing," Math. Program, vol.157, no.2, pp.421–449, 2021.

[6] N. Mazumdar, A. Nag, and J.P. Singh, "Trust-based load-offloading protocol to reduce service delays in fog-computing-empowered IOT," Computers and Electrical Engineering, vol.93, 107223, 2021.

[7] M.P.J. Mahenge, C. Li, and C.A. Sanga, "Energy-efficient task offloading strategy in mobile edge computing for resource-intensive mobile applications," Digit. Commun. Networks, vol.8, no.6, pp.1048–1058, 2022.

[8] S. Vimal, M. Khari, N. Dey, R.G. Crespo, and Y.H. Robinson, "Enhanced resource allocation in mobile edge computing using reinforcement learning based MOACO algorithm for IIOT," Comput. Commun., vol.151, pp.355–364, 2020.

[9] Z. Tong, X. Deng, F. Ye, S. Basodi, X. Xiao, and Y. Pan, "Adaptive computation offloading and resource allocation strategy in a mobile edge computing environment," Inf. Sci., vol.537, pp.116–131, 2020.

[10] D. Li, Y. Jin, and H. Liu, "Resource allocation strategy of edge systems based on task priority and an optimal integer linear programming algorithm," Symmetry, vol.12, no.6, p.972, 2020.

[11] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility aware joint task scheduling and resource allocation for cooperative mobile edge computing," IEEE Trans. Wireless Commun., vol.20, no.1, pp.360–374, 2021.

[12] B. Gong and X. Jiang, "Dependent task-offloading strategy based on deep reinforcement learning in mobile edge computing," Wireless Communications and Mobile Computing, vol.2023, p.12, 2023.

[13] T. Yang and J. Yang, "Deep reinforcement learning method of offloading decision and resource allocation in MEC," Computer Engineering, vol.47, no.8, pp.37–44, 2021.

[14] F. Zhou, Y. Wu, R.Q. Hu, and Y. Qian, "Computation rate maximization in uav-enabled wireless powered mobile edge computing systems," IEEE J. Sel. Areas Commun., vol.36, no.9, pp.1927–1941, 2018.

[15] W. Feng, N. Zhang, S. Li, S. Lin, R. Ning, S. Yang, and Y. Gao, "Latency minimization of reverse offloading in vehicular edge computing," IEEE Trans. Veh. Technol., vol.71, no.5, pp.5343–5357, 2022.

[16] X. Chen, Y. Cai, L. Li, and M. Zhao, "Energy-efficient resource allocation for latency-sensitive mobile edge computing," IEEE Trans. Veh. Technol., vol.69, no.2, pp.2246–2262, 2020.

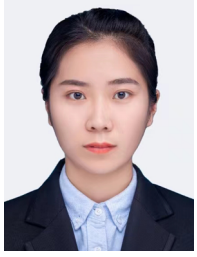
[17] Y. Mao, T. Zhou, and P. Liu, "Multi-user task offloading based on delayed acceptance," Computer Science, vol.48, no.1, pp.49–57, 2021.

[18] Z. Tong, F. Ye, J. Mei, B. Liu, and L. Li, "A novel task offloading algorithm based on an integrated trust mechanism in mobile edge computing," Journal of Parallel and Distributed Computing, vol.169, pp.185–198, 2022.

[19] I.A. Elgendy, W. Zhang, and H. He, "Joint computation offloading and task caching for multi-user and multi-task MEC systems: Reinforcement learning-based algorithms," Wireless Netw., vol.27, no.3, pp.2023–2038, 2021.

[20] O.K. Shahryari, H. Pedram, and V. Khajehvand, "Energy and task completion time trade-off for task offloading in fog-enabled IOT networks," Pervasive and Mobile Computing, vol.74, no.101395, 2021.

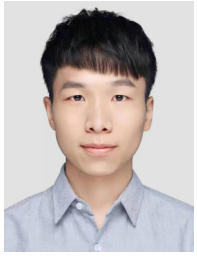
[21] X. Zhu and Y. He, "Energy efficiency dynamic task scheduling in wireless sensor networks," Computer Engineering and Design, vol.41, no.2, pp.313–318, 2020.



Daxiu Zhang received her M.S. degree from Anhui University of Technology, Huainan, China in 2018. She has been working in the School of Computer Engineering, Bengbu University and Information Technology, Quanzhou Vocational College of Economics and Business. Her research interests include IoT, Machine Learning and Big Data.

Xianwei Li received his M.S. degree from Hunan University, Changsha, China in 2010 and ScD degree from Waseda University, Tokyo, Japan in 2019. He is now an associate professor with the School of Computer and Information Engineering, Bengbu University. His research interests include IoT, Machine Learning and Big Data.

Bo Wei received the Ph.D. degree from Waseda University, Tokyo, Japan, in 2019. From 2019 to 2023, she was an Assistant Professor with the Graduate School of Fundamental Science and Engineering, Waseda University. She is currently a specially appointed Assistant Professor with The University of Tokyo and the PRESTO Researcher with JST. Her research interests include wireless communication, machine learning, adaptive video transmission, computer networking, quantum computing, and the Internet of Things.



Yukun Shi received his M.S. degree from Anhui University of Technology, Huainan, China in 2018, and is currently a Ph.D. student at the School of Cyberspace Security, Hangzhou University of Electronic Science and Technology, Hangzhou, China. His research interests include Cyberspace Security, Machine Learning and Big Data.