J Inf Process Syst, Vol.17, No.4, pp.721~736, August 2021
https://doi.org/10.3745/JIPS.01.0079

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

A Joint Allocation Algorithm of Computing and Communication Resources Based on Reinforcement Learning in MEC System

Qinghua Liu* and Qingping Li*

Abstract

For the mobile edge computing (MEC) system supporting dense network, a joint allocation algorithm of computing and communication resources based on reinforcement learning is proposed. The energy consumption of task execution is defined as the maximum energy consumption of each user's task execution in the system. Considering the constraints of task unloading, power allocation, transmission rate and calculation resource allocation, the problem of joint task unloading and resource allocation is modeled as a problem of maximum task execution energy consumption minimization. As a mixed integer nonlinear programming problem, it is difficult to be directly solve by traditional optimization methods. This paper uses reinforcement learning algorithm to solve this problem. Then, the Markov decision-making process and the theoretical basis of reinforcement learning are introduced to provide a theoretical basis for the algorithm simulation experiment. Based on the algorithm of reinforcement learning and joint allocation of communication resources, the joint optimization of data task unloading and power control strategy is carried out for each terminal device, and the local computing model and task unloading model are built. The simulation results show that the total task computation cost of the proposed algorithm is 5%–10% less than that of the two comparison algorithms under the same task input. At the same time, the total task computation cost of the proposed algorithm is more than 5% less than that of the two new comparison algorithms.

Keywords

Cellular MEC System, Markov Decision Process, Resource Allocation, Reinforcement Learning, Task Unloading

1. Introduction

For the user equipment (UE), limited resources and power are the major factors affecting its service quality. With its birth, mobile edge computing (MEC) technology places the MEC server with efficient computing capabilities at the edge of the mobile network, closer to the user [1-3].

In [4], the offload decision mainly considers three states: task queue state, local processing unit state and launch unit state. Considering the energy consumption of processing data and the allocation of computing resources [5], the semidefinite relaxation scheme is expanded in [6]. However, the above literatures fail to consider the dynamic arrival of tasks [7]. In [8], energy collection and tasks with different priorities are emphasized. At the same time, a low time complexity calculation offloading

Manuscript received May 14, 2020; first revision July 27, 2020; second revision September 4, 2020; accepted October 11, 2020.
Corresponding Author: Qinghua Liu (582867268@qq.com)
* Dept. of Information Technology, Zhejiang Yuying College of Vocational Technology, Zhejiang, Hangzhou, China (582867268@qq.com, lliu178@126.com)

scheme is proposed, and the algorithm training is accelerated by using value function approximation.

The above documents are for a single MEC service node. In order to achieve ubiquitous edge computing, the authors [9] studies a stochastic optimization problem to minimize the cost of the MEC system. In a closed form, the best CPU frequency and the best transmit power can be realized. Dihn et al. [10] proposes a computing offload architecture that allows mobile devices to offload tasks to multiple edge servers. By jointly optimizing offload decisions and CPU frequency scaling, a positive semidefinite relaxation scheme is obtained. In [11], two task unloading algorithms that consider both energy efficiency and computation time constraints are proposed. For the first case, a system environment that rarely changes, an optimal static algorithm based on dynamic programming technology is proposed to obtain an accurate solution.

Although the above literature can achieve good results under different constraints and scenarios for computing offloading, it rarely considers the problems of dynamic offloading, access and joint optimization of resources under multiple servers. Therefore, this paper designs an efficient joint computing and communication resource allocation model for MEC systems. The model uses reinforcement learning algorithm, which aims to ensure the performance of traditional algorithm while solving the high complexity of traditional algorithm in MEC. The main contributions of this paper are as follows:

Under the constraints of task offloading, power allocation, transmission rate, and computing resource allocation, an efficient joint learning and computing resource allocation algorithm based on reinforcement learning is proposed for MEC systems. The problems of computing task offload and power control in the MEC system with multiple base stations alleviate the data processing pressure of the MEC server.

2. Modeling

2.1 System Model

Consider a MEC system consisting of V small base station deploying MEC servers and UEs to support dense networking, as shown in Fig. 1. Define the UE set as $\Sigma = \{1,2,\dots,N\}$; let $i \in \Sigma$ denote UE i, and $|\Sigma| = N$. UE can choose to execute the task locally, and also unload the task to MEC server for execution through the link. The set of small base station (BS) is $\Gamma = \{1,2,\dots,V\}$. Let $j \in \Gamma$ denote small BS j, and $|\Gamma| = V$. Assuming that the UE in the system needs to perform energy-intensive and computationally intensive tasks, and the task request of the UE i can be represented by the binary group (I_i, D_i). Then, I_i (bits) represents the input data amount of the UE task. D_i represents the amount of computing resources required to complete the UE task, and its unit is the number of cycles.

Assuming that the bandwidth of the small BS j and the maximum number of accessible UEs are W_j and B_j, respectively, and the bandwidth and maximum number of UEs of the small base stations are different. To efficiently use the resources of the small BS, it is further assumed that multiple UEs access the small BS simultaneously in an orthogonal manner. Therefore, the sub-channel bandwidth available to the UE accessing the small base station j is W_j^{sub}, and there is no interference between the UEs.

Assuming that the computing power of the small BS j MEC server and the maximum number of serviceable UEs are F_j and S_j, respectively, ensuring the full use of the computing power of the MEC server and better performance of UE task execution. It is assumed that multiple UEs can be offloaded to

the MEC server through a cellular link for its task, and each UE can be allocated a certain amount of computing resources.

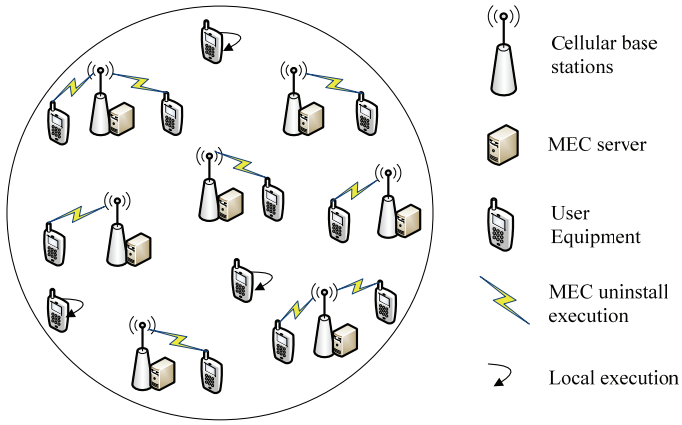

Fig. 1. System model diagram.

2.2 Problem Description

In a cellular MEC system that supports dense networking, the UE can choose two modes of local execution and MEC offload execution to complete its tasks. Due to the difference in channel bandwidth and computing power between base stations, UE selecting different base stations and performing tasks in different modes will produce different energy consumption. When UEs intend to perform energy-intensive and computationally intensive tasks, each UE may have different QoS requirements. This paper focuses on how to select task offload strategy, power allocation, and computing resource allocation strategy for the UE on the premise of base station bandwidth resources, available computing resources, and the task's characteristics.

2.3 Optimization Modeling

Each UE is designed to meet the maximum task execution energy of all UEs, optimal task unloading and resource allocation strategy. Q_i is used to denote the task offload decision of the i-th UE. Therefore, the optimal task offload decision Q_i^* when executing task T_i should satisfy the lowest total calculation and transmission overhead. The optimal task offload decision can be expressed as $Q^* = (Q_1^*, \ldots, Q_i^*)$.

Objective function modeling

In order to take into account the fairness between UEs, the task execution energy consumption is defined as the maximum energy consumption required by each UE task execution in the system, and the expression is

$$E = \max_{i \in N} E_i \tag{1}$$

In the above formula, E_i represents UE i task execution energy consumption. Among them, the UE i task execution energy consumption modeling is:

$$E_i = \left(1 - \sum_{j \in V} \mu_i\right) E_i^L + \sum_{j \in V} \mu_i E_{i,j}^M \tag{2}$$

where μ_i represents the task offload variable, which is a weighting factor used to adjust the execution delay and energy consumption ratio; E_i^L represents the local execution energy consumption of the UE i task; $E_{i,j}^M$ represents the UE i task offloaded to the small BS j and executed on its MEC server Need energy consumption?. Among them, if UE ii offloads the task to small BS j and executes it on its MEC server, $\mu_i = 1$; otherwise, $\mu_i = 0$, $i \in N$, and $j \in V$.

2.4 Constraint Modeling

The modeling optimization problem needs to meet the following constraints.

Task offload constraints

Limited by access and service capabilities, the number of offloaded UEs served by each small BS should not exceed its maximum number of serviceable UEs:

$$C1: \sum_{i \in N} x_{i,j} \le \min\left\{B_j, S_j\right\} \tag{3}$$

where S_j represents the maximum number of UEs that can be served by the MEC server of the small BS j. For simplicity, each UE is assumed to offload its task to a BS at most:

$$C2: \sum_{j \in V} x_{i,j} \le 1 \tag{4}$$

Power allocation constraints

The transmission power of the UE is non-negative and should not exceed its maximum transmission power:

$$C3: 0 \le p_{i,j} \le p_i^{max} \tag{5}$$

where $p_{i,j}$ indicates the power that the UE i sends the task input data to the small BS j.

Transmission rate constraints

In order to ensure the transmission performance of the task in the offload mode, it is assumed that the transmission rate of the UE should be greater than the minimum task transmission rate requirement:

$$C4: R_{i,j} \ge R_i^{min} \tag{6}$$

Computing resource allocation constraints

The computing resources of the MEC server allocated to the UE should be non-negative and should not exceed the total MEC server computing resources of the small BS j:

$$C5: 0 \le F_{i,j} \le F_j \tag{7}$$

The sum of the resources allocated to each UE should not exceed the total resources of the MEC server:

$$C6: \sum_{i \in N} F_{i,j} \le F_j \tag{8}$$

2.5 Optimization Model

Based on the modeled objective function and constraint conditions in Section 3.4, the optimization goal is to minimize the maximum task execution energy consumption. When the constraints such as task offload, power allocation, transmission rate, and computing resource allocation are satisfied, the modeling is realized based on the maximum task execution energy the optimization model of joint task offloading and resource allocation with minimum consumption is:

$$\begin{aligned} &\min_{x_{i,j}, P_{i,j}, F_{i,j}} E \\ &s.t.\ \ C1\text{-}C6, \\ &\quad C7: x_{i,j} \in \{0,1\} \end{aligned} \tag{9}$$

However, finding the optimal solution of formula (9) is an NP problem because it is an example of mixed integer nonlinear programming. And it is beyond the scope of this paper; meanwhile, the use of traditional algorithms to solve this problem has become more and more difficult, and the performance of the algorithm is getting closer to the bottleneck. Therefore, this paper uses reinforcement learning related algorithms to solve this problem.

3. Joint Assignment Algorithm Based on Reinforcement Learning

3.1 Reinforcement Learning Theory

Reinforcement learning is introduced in this section mainly from the perspective of reinforcement learning problems, value functions, function approximation and strategy optimization.

To set up the problem, a reinforcement learning agent needs to constantly interact with the environment (Fig. 2). At time t, the agent will receive a state s_t from state space S and select an action a_t from action space A. Then a reward value r_t is received, and the state transitions to the next state s_{t+1}. The action selection is determined by the decision $\pi(a_t|s_t)$ made by the agent, and the action selected by the agent for the state is defined as a strategy, that is, the agent's behavior. The transition probability between states is $(s_{t+1}|s_t, a_t)$.

In a reinforcement learning problem, the cycle of state, action, and value reward will continue until the end state is reached and then restarted, that is, the process such as $s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_t, a_t, r_t$ will continue until the end state. Since reinforcement learning seeks the cumulative long-term maximum return sum starting from the current state, the return sum for the state at each moment is

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{10}$$

where γ represents the discount factor, $\gamma \in (0,1]$.

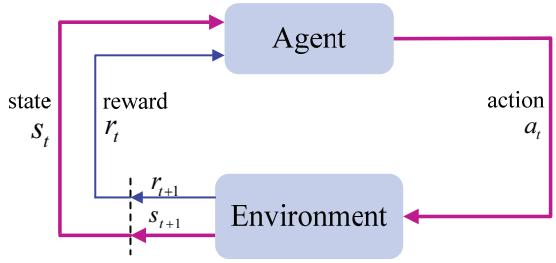

Fig. 2. Interaction between agent and environment in reinforcement learning.

The value function represents the sum of the accumulated, discounted, and future returns from the current moment, that is, the return and R_t in the current state s_t. It is used to evaluate each state or (state, action) pair Good or bad.

The definition of the state value function is as follows:

$$v_\pi(s) = E[R_t \mid s_t = s] \tag{11}$$

It represents the expected return in accordance with strategy π under state s. Dividing the state value into the Bellman equation is equal to:

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s,r} p(s', r \mid s, a)\left[r + \gamma v_\pi(s')\right] \tag{12}$$

Formula (12) splits the current state value function into a cumulative form of the current action reward r and the next state value function $v_\pi(s')$, where s represents the current state, and s' represents the next state generated by action a based on the current state s and policy π. Since the goal of reinforcement learning is to maximize the sum of returns, VS defines the optimal state value as $v_*(s) = max_\pi v_\pi(s)$, which represents the maximum value function that can be achieved for all strategies under state s. Therefore, the Bellman equation that $v_*(s)$ can be split into:

$$v_*(s) = \sum_a \pi(a \mid s) \sum_{s,r} p(s', r \mid s, a)\left[r + \gamma v_*(s')\right] \tag{13}$$

Another kind of value function is (state, action) value function, where the (state, action) value function $q_\pi(s,a) = E[R_t | s_t = s, a_t = a]$ represents the reward of selecting action a from the current state s and executing strategy π. Similar to the state value function, it can also be written in the form of the Bellman equation:

$$q_\pi(s,a) = \sum_{s,r} p(s', r \mid s, a)\left[r + \gamma \max_a \cdot q_\pi(s', a')\right] \tag{14}$$

Similar to the state value function, the best strategy for the (state, action) value function is the largest value that can be obtained among all strategies in the case of state s and action a. The optimal strategy of the (state, action) value function can also be split into the form of Bellman equation, which will not be repeated here.

Temporal difference (TD) learning

When the reinforcement learning problem satisfies Markov's attributes and will only depend on the current state and action rather than the past state and action in the future, it can be modeled as an Markov decision process (MDP) problem. When the system model is available, the best strategy can be obtained through dynamic programming. The quality of the strategy can be evaluated by calculating the state value function or (state, action) value function [12,13].

For the two forms of value function, state value function and (state, action) value function, the best strategy to maximize the cumulative return needs to be determined. As an important learning method in reinforcement learning, time series difference is the basic method of updating strategies and is widely used in finding the best strategy. The TD method often involves the evaluation of state value functions. The methods of SARSA [14] and Q-learning [15] are representative TD methods. They learn the state value function $V(s)$ directly from TD error experience. The update rules for TD learning are:

$$V\left(s\right) \leftarrow V\left(s\right) + \alpha\left[r + \gamma V\left(s^{'}\right) - V\left(s\right)\right] \tag{15}$$

where α represents the learning rate, and $r + \gamma V(s') - V(s)$ is called TD error, which represents the difference between each newly explored state value and the past state value. Therefore, the state value function can be continuously updated by the TD method.

3.2 Resource Association Model

In the case of using the same multiplication fusion method, that is, a multiplication fusion method from time flow to space flow, the effects of fusion number and position on recognition performance are experimentally analyzed. The experimental results are shown in Table 1. The results reveal the recognition accuracy obtained after training on RML and BAUM dataset. "conv2_1_relu and conv2_1" indicates that the conv2_1 layer of the time stream is connected to the conv2_1_relu layer of the spatial stream for multiplication fusion, and so on.

In the resource allocation system model established in this paper, it is assumed that small base stations are evenly distributed in a circular area with a radius of 1,000 m, and all end user equipment is randomly distributed in this area. In the MEC system, each UE trying to access the MEC server interacts with the environment as an MDP decision maker (i.e., an agent), and the environment is composed of interference caused by the UE sharing the channel and the MEC server. Despite the uncertainty in the environment (the number of other UEs that choose the same small BS as this UE is dynamic; and for the resource allocation algorithm based on power control, due to the different power selection, the interference generated by each UE is also dynamic), Mobile terminal user equipment still seeks to maximize the MEC benefit rate and minimize the total overhead. The actions of the UE can affect the future state of the environment (interference level of the small BS), thereby further affecting the action selection and state space of the UE at the next time step.

More precisely, the task offload decision-making process is modeled as an MDP, which is essentially a discrete-time random control process. An agent-environment interaction process of MDP is called an episode, which is equivalent to a task offload cycle *T*. In each episode, MDP iterates from a random initial state until it finally converges. The end user device as the decision maker selects an action a from the optional actions in the state s_t according to different strategies adopted by different algorithms and executes it. At the same time, the MEC server as part of the environment responds and feeds back the corresponding reward r to UE, and then the UE enters the next state s_{t+1}.

Table 1. Simulation parameter settings

System parameter	Value
Number of small BS V	12
Number of UE N	15
The bandwidth of small BS W_j	10–20 MHz
Maximum number of users that small BS can access B_i	3−5
Noise power σ^2	-80 dBm
Computing ability of the MEC server F_j	10–15 G cycles/s
The amount of data by the task I_i	0.6–3 Mbits
The amount of computing resources D_i	0.4–0.7 Gcycles
Transmission power P	0.1–0.5 W
Computing ability of UE i F_i	1–2 G cycles/s
Weight factor for task offload μ_i	0.6
Learning rate α	0.2
Reward discount factor γ	0.8
Maximum unloading cycle T	60

According to the introduction in Section 3.1, the exact definitions of the state, actions and reward functions of the MDP used in the joint calculation and communication resource allocation algorithm under no power control are given below:

(1) State

At any time step t, if a UE chooses to offload its computing task through the small BS j, we say that the UE is in the state $\varphi_j(\forall j \in V)$; if the UE chooses to perform the computing task locally, it is defined in the state φ_0. Therefore, the UE's state set can be expressed as $S = \{\varphi_0, \varphi_1, \dots, \varphi_V\}$.

(2) Action

For each time step t, the UE must select and execute an action a in the current state s_t according to the strategy used, and at the same time the UE transitions from the current state s_t to the next state $s_{t+1}(\forall s_t, s_{t+1} \in S)$. We use $A = \{\phi_0, \phi_1, \dots, \phi_V\}$ to represent the action space of the UE in the MDP. For a UE, $a = \phi_0$ means that it chooses to perform the computing task locally. Accordingly, $a = \phi_0(\forall j \in S)$ means that it chooses the small BS j to offload its computing task to the MEC server.

(3) Reward function

After the agent-environment interaction in each time step t, the UE as an agent will get a feedback from the environment, that is, reward r, which is used to reflect the good results of the UE after performing an action in a certain state of bad situation. The measure of maximizing the benefit rate of

the first optimization target MEC proposed above is also the overall cost of the UE, while the average cost of the second optimization target UE is directly based on the ratio of the total system cost to the number of UEs. The reward function in the resource allocation algorithm without power control can be specifically defined as:

$$r(s,a)=\begin{cases}\lambda_1/E_i^L, & \text{if } a=\phi_0\\ \lambda_2/E_{i,j}^M, & \text{if } a=\phi_V\end{cases},\forall s\in S,\forall a\in A \tag{16}$$

Among them, λ_1 and λ_2 are standardized variable.

3.2.1 Local computing model

For the UE i, $\boldsymbol{T}_i$ represents its calculation task, which is defined as a two-dimensional array (b_i, c_i), where b_i (in bits) represents the input data amount of the task, and c_i(in CPU revolutions per bit) represents the calculation of each bit task, the required CPU revolutions. The values of b_i and c_i depend on the nature of the specific task and can be obtained by offline measurement. We use $d_i = j$ to indicate that the *i*-th UE chooses to offload the task to the MEC server through the j-th small base station, that is, there is $d_i \in \{0,1,\dots,j,\forall j \in V\}$ and when $d_i = 0$ it indicates that UE i chooses to perform its computing task locally.

For UE i, if it chooses to execute the task locally, the delay generated by the calculation task $\boldsymbol{T}_i$ is expressed as:

$$T_i^L = b_i c_i / F_i, \forall i \in N \tag{17}$$

where F_i represents the computing power of the UE i and is measured in revolutions per second of the CPU (revolutions per second). We use V_i to represent the energy consumed by the UE i per second when performing local calculations. The total energy consumed by the local computing task $\boldsymbol{T}_i$ is defined as:

$$E_i^L = V_i T_i^L, \forall i \in N \tag{18}$$

In this paper, we have considered various service quality requirements of UEs, that is to say, some delay-sensitive UEs (such as mobile phones, monitoring equipment, etc.) need to obtain the lowest possible delay, but can accept higher energy consumption. For some UEs that are sensitive to energy consumption (such as sensor nodes, IoT devices, etc.), they need to meet the minimum energy consumption. These devices usually do not have high requirements for delay. Therefore, we adopt the composite index of calculation cost as in [16] to reflect the total cost of a terminal device performing a calculation task under different service quality requirements. Specifically, when the UE i chooses to execute the computing task $\boldsymbol{T}_i$ locally the total cost can be defined as:

$$U_i^L = \mu_i T_i^L + (1-\mu_i) E_i^L, \forall i \in N \tag{19}$$

When a UE is in a low power state, it is more meaningful to reduce energy consumption compared to calculation and transmission delay. Therefore, μ_i can be set to 0. On the contrary, when a UE has sufficient power and needs to run some delay-sensitive applications, μ_i can be set to 1. Therefore, for

devices with different service quality requirements and applications with different service types, the setting of the weighting factor μ_i is also one of the indicators to improve the algorithm performance of the system.

3.2.2 Task offload model

In view of the previous research on MEC and mobile networks, in order to facilitate the analysis, we only consider a quasi-static situation: a group of active UEs in a task offload decision period T (for example, hundreds of milliseconds) and its wireless channel conditions. It stays the same, and may change in different cycles but have no effect on the algorithm performance of the system. We also assume that each small base station has only one physical channel and does not overlap with each other. Each UE can select a specific small base station to offload computing tasks to the MEC server.

The transmission power used by the UE i to upload the task input data to the MEC server through the small BS is $p_{i,j}$. In the resource allocation algorithm without power control, we use a fixed transmission power for each UE according to the relevant power control algorithm. At the same time, we give the active UE's decision combination $q = (Q_1, \ldots, Q_i)$ and the data transmission rate when the *i*-th UE selects the *j*-th small BS to offload its task:

$$R_{i,j} = W_j^{sub} \log_2 \left(1 + \frac{p_{i,j} g_{i,j}}{\sigma^2 + \sum_{n \in N - \{i\} \text{ and } d_n = d_i} p_{n,j} g_{n,j}} \right) \tag{20}$$

where σ^2 represents the noise variance at the *j*-th small base station, $g_{i,j}$ represents the power gain of the channel between the *i*-th UE and the *j*-th small base station, and the $(n \in N - \{1\} \text{ and } d_n = d_i)$ term indicates that other than the UE also chooses the j-th small base station to offload UE n to MEC server. The sub-channel bandwidth W_j^{sub} of small BS j is modeled as:

$$W_j^{sub} = \frac{W_j}{B_j} \tag{21}$$

Because the MEC server provides powerful computing power (because many telecom operators have the ability to make large-scale infrastructure investments). In addition, because the amount of data in the calculation result is small, compared with the amount of input data and task calculation, the feedback delay is also negligible. Therefore, when the small base station j performs the calculation task T_i remotely on the MEC server, the delay can be expressed as

$$T_{i,j}^t = I_i / R_{i,j}, \forall i \in N \text{ and } \forall j \in V \tag{22}$$

The energy consumption mainly comes from the UE transmitting the task input data to the small base station, which can be expressed as:

$$E_{i,j}^t = p_{i,j} T_{i,j}^t, \forall i \in N \text{ and } \forall j \in V \tag{23}$$

When the terminal device i chooses to offload the task to the MEC server through the small base station j, we can give the following definition, and use the weighted sum of execution delay and energy consumption to describe the UEI calculation total cost.

$$E_{i,j}^{M} = \mu_i T_{i,j}^{t} + \left(1-\mu_i\right) E_{i,j}^{t}, \forall i \in N \text{ and } \forall j \in V \tag{24}$$

4. Experiment and Discussion

In this section, MATLAB simulation software is used to verify the performance of the algorithm proposed in this paper, and comparison is made with the algorithms in [7] and [8], which are also based on reinforcement learning, and the algorithms of [9] and [11], which are two new algorithms of computing offload decision and resource allocation.

4.1 Experimental Parameters

The simulation scenario is a MEC system composed of multiple small base stations and multiple UEs that supports dense networking. The size of the simulation area is 1,200 m × 1,200 m. Both the UE and the small base station are randomly distributed within the simulation area. As shown in Table 1, other simulation parameters take values according to Table 1 unless otherwise specified. In addition, in all simulation experiments, for any transmission power involved in no power control, the maximum transmission power is used for all UEs.

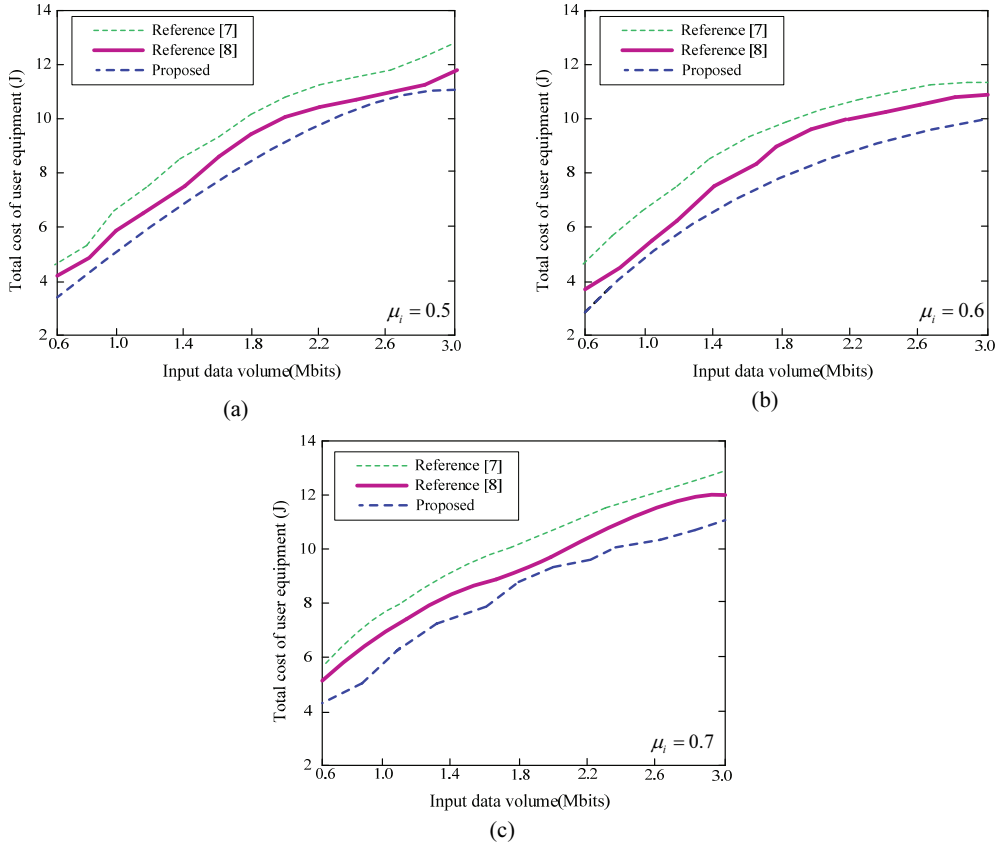

Fig. 3. The relationship between the total cost of task calculation and the amount of input data: (a) μ_i=0.5, (b) μ_i=0.6, and (c) μ_i=0.7.

4.2 Comparison and Analysis of RL Algorithm

Fig. 3 is a graph of the relationship between the task execution cost and the amount of task input data obtained by the proposed algorithm, the algorithm in [7], and the algorithm in [8]. Here, the task unloading weight factor μ_i is set to values 0.5, 0.6, 0.7, respectively, and comparative analysis is made in three cases. As can be seen from Fig. 3(a), 3(b), and 3(c), in three cases, as the amount of task input data increases, the task execution energy consumption of the three algorithms increases, because the task input data The increase in the amount will cause the task execution delay to increase, which in turn will cause the task execution energy consumption to increase. Under the same task input data volume, the total computational cost of the proposed algorithm task is 5%–10% less than that of the two comparison algorithms. This is because the algorithm proposed in this paper considers the problems of dynamic offloading, joint access and resource optimization under multiple servers, and whether the terminal equipment benefits from MEC is measured according to its comprehensive cost. With the addition of power control, the average over.head of all terminal devices in the entire system is reduced to a certain extent.

Fig. 4 is a comparison of task execution cost when the proposed algorithm has unloading weight factors μ_i set to values 0.5, 0.6, and 0.7, respectively. It can be seen that when the μ_i value is 0.6, the execution cost of the proposed algorithm task is the smallest. Therefore, in subsequent experiments, the proposed task unloading weight factor μ_i all takes a value of 0.6.

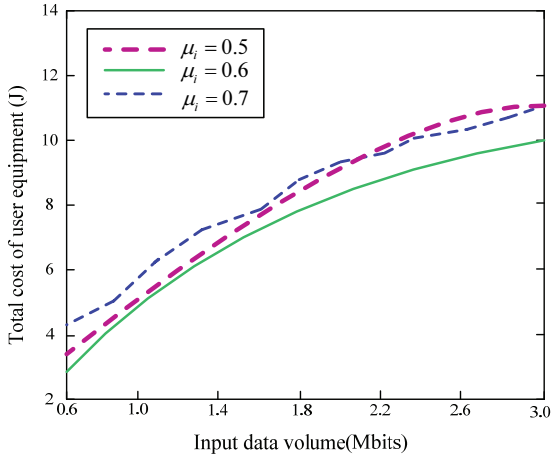

Fig. 4. Comparison of the proposed algorithm in three values of unloading weight factor.

4.3 Comparison and Analysis with New Computing Offload Decision and Resource Allocation Algorithms

Two new algorithms of [9] and [11] are selected for comparative experiments. In this experiment, the proposed algorithm task offload weight factor value μ_i is 0.6.

Fig. 5 shows the relationship between the task execution cost and the amount of task input data obtained by the proposed algorithm, the algorithm in [9], and the algorithm in [11]. Under the same task input data volume, the total computational cost of the proposed algorithm task is more than 5%, less than the two comparison algorithms. In addition, as the minimum rate requirement for task transmission increases, the task execution energy consumption increases. The higher the minimum transmission rate requirement,

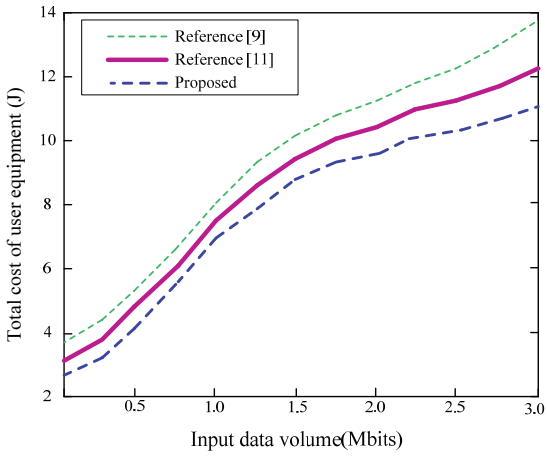

Fig. 5. The relationship between the total cost of task calculation and the amount of input data.

the higher the transmission power required, so more task execution overhead is consumed.

Based on reinforcement learning, the joint calculation and communication resource allocation algorithm optimizes the data task offload and power control strategy for each terminal device. When the terminal device chooses to offload its task to the MEC server, the choice of power becomes more flexible. At the same time, the impact on other terminal equipment is also relatively reduced, providing better performance optimization for the coordinated operation of the entire system.

As can be seen from Fig. 6, as the amount of computing resources required to complete the task increases, the total cost of task calculation increases. This is because more intensive computing tasks lead to increased execution delay, which in turn consumes more energy. From the comparison of the three algorithms, we can see that the algorithm of [9] has the highest algorithm cost and a linear growth trend. The total cost of the algorithm in [11] and the algorithm proposed in this paper decreases in turn, and the cost of the latter is lower than the task execution cost of the comparison algorithms. This is because the algorithm of [9] only supports random offloading of tasks and is not optimized according to the amount of computing resources required to complete the task.

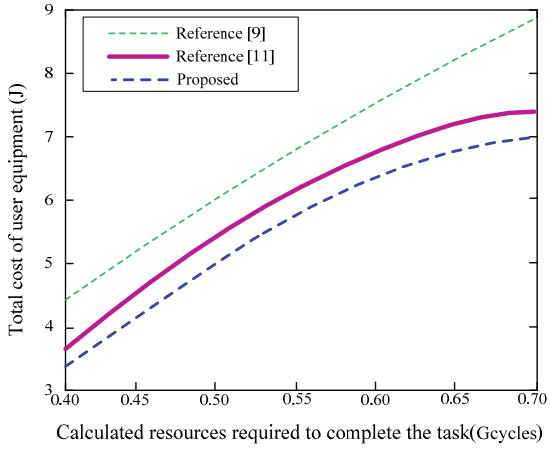

Fig. 6. The relationship between the total cost of task calculation and the amount of computing resources required to complete the task.

Fig. 7 compares the total computational cost of each algorithm task in two scenarios. It can be seen that the task execution energy consumption of the algorithm in [9] in the best and worst cases is the highest among the three algorithms. This is because the authors [11] uses system task execution energy consumption and optimizes task offloading and resource allocation strategies as expected, but fails to take into account user fairness, resulting in a large gap between the total task computations overhead in both cases. The optimal energy consumption of the algorithm in [11] is lower than that of the algorithm in [9], which can reflect the advantage of task offloading to save equipment energy consumption to a certain extent, but due to its failure to calculate the channel gain and the load situation of the MEC server determines the joint strategy. The worst-case task execution cost is still close to the algorithm of [9].

In addition, in [11], the total task computational cost is lower than the energy consumption obtained by the algorithm proposed in this paper in the worst case, but the gap is small. The algorithm proposed in this paper is based on the joint algorithm of computing and communication resource allocation for reinforcement learning, and each UE is jointly optimized for data task offloading and power control strategy. When the terminal device chooses to offload its task to the MEC server, the choice of power becomes more flexible. At the same time, the impact on other UEs is relatively reduced, providing better performance optimization for the coordinated operation of the entire system.

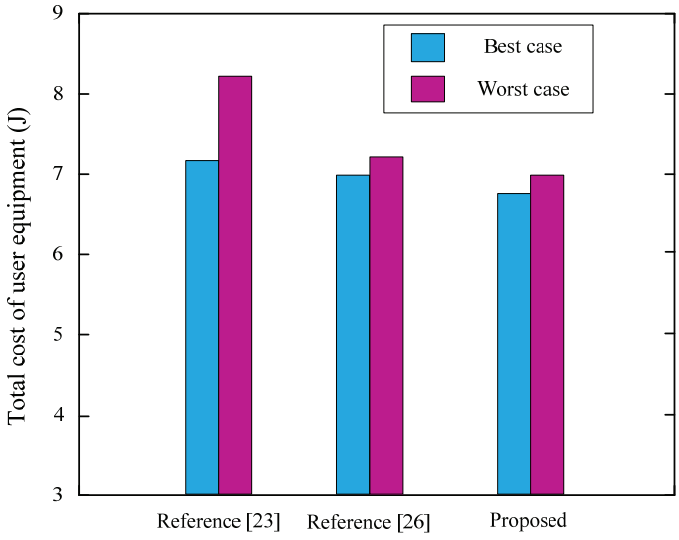

Fig. 7. Comparison graph of total task calculation overhead obtained by various algorithms in different situations.

5. Conclusion

Theoretical knowledge of reinforcement learning used in the study is introduced in detail, including several commonly used algorithms and frameworks in reinforcement learning. In order to highlight the comparison of the effects of resource allocation algorithms and the results of simulation experiments, a joint algorithm for computing and communication resources allocation based on reinforcement learning is proposed, and the algorithm is analyzed in detail, and the simulation experiments are carried out, and

finally the simulation results are compared, analyzed and summarized. The problems of computing task offload and power control in the MEC system with multiple base stations alleviate the data processing pressure of the MEC server.

However, there are still some problems and deficiencies in the joint resource allocation algorithm proposed in this paper. In the task offloading model, we propose an assumption that the wireless channel conditions of a group of terminal devices remain the same during the same task offloading period. This is difficult to achieve in application scenarios. Therefore, for this problem, in the next stage of research, we will add the time-varying situation of the wireless channel to the resource allocation system model, so that the performance of the algorithm is closer to the real scene, to further expand reinforcement learning in MEC application in the system.

References

[1] P. Mach and Z. Becvar, "Mobile edge computing: a survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628-1656, 2017.

[2] C. Li, J. Tang, and Y. Luo, "Dynamic multi-user computation offloading for wireless powered mobile edge computing," *Journal of Network and Computer Applications*, vol. 131, pp. 1-15, 2019.

[3] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: the communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322-2358, 2017.

[4] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proceedings of 2016 IEEE International Symposium on Information Theory (ISIT)*, Barcelona, Spain, 2016, pp. 1451-1455.

[5] M. H. Chen, B. Liang, and M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," in *Proceedings of 2016 IEEE International Conference on Communications (ICC)*, Kuala Lumpur, Malaysia, 2016, pp. 1-6.

[6] M. H. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, Atlanta, GA, 2017, pp. 1-9.

[7] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *Proceedings of 2018 IEEE Wireless Communications and Networking Conference (WCNC)*, Barcelona, Spain, 2018, pp. 1-6.

[8] Z. Wei, B. Zhao, J. Su, and X. Lu, "Dynamic edge computation offloading for internet of things with energy harvesting: a learning method," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4436-4447, 2018.

[9] W. Chen, Y. He, and J. Qiao, "Cost minimization for cooperative mobile edge computing systems," in *Proceedings of 2019 28th Wireless and Optical Communications Conference (WOCC)*, Beijing, China, 2019, pp. 1-5.

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

[11] Z. Zhang, J. Wu, L. Chen, G. Jiang, and S. K. Lam, "Collaborative task offloading with computation result reusing for mobile edge computing," *The Computer Journal*, vol. 62, no. 10, pp. 1450-1462, 2019.

[12] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, "Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA," *IEEE Access*, vol. 8, pp. 54074-54084, 2020.

[13] T. D. Parker, C. F. Slattery, J. Zhang, J. M. Nicholas, R. W. Paterson, A. J. Foulkes, et al., "Cortical microstructure in young onset Alzheimer's disease using neurite orientation dispersion and density imaging," *Human Brain Mapping*, vol. 39, no. 7, pp. 3005-3017, 2018.

[14] D. Ramachandran and R. Gupta, "Smoothed sarsa: reinforcement learning for robot delivery tasks," in *Proceedings of 2009 IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009, pp. 2125-2132.

[15] A. Larmo and R. Susitaival, "RAN overload control for machine type communications in LTE," in *Proceedings of 2012 IEEE Globecom Workshops*, Anaheim, CA, 2012, pp. 1626-1631.

[16] 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.

Qinghua Liu **https://orcid.org/0000-0002-0689-1292**

She has got Master's degree of Computer Application Technology. She Graduated from Shanghai University in 2011. She is working in Zhejiang Yuying College of Vocational Technology. She is a lecturer now. Her research interests include computer information management technology and computer network technology.

Qingping Li **https://orcid.org/0000-0003-3110-5588**

He has got Bachelor's degree of Computer Science and Technology. He Graduated from Jiangxi Agricultural University in 1991. He is working in Zhejiang Yuying College of Vocational Technology. He is a professor now. His research interests include computer network technology and programming language.