Scientific Research Publishing

Journal of Computer and Communications, 2024, 12, 152-168
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Multi-Agent Deep Deterministic Policy Gradien-Based Task Offloading Resource Allocation Joint Offloading

Xuan Zhang, Xiaohui Hu

College of Electronics and Information Engineering, Lanzhou Jiaotong University, Lanzhou, China
Email: 605322439@qq.com, 307509416@qq.com

How to cite this paper: Zhang, X. and Hu, X.H. (2024) Multi-Agent Deep Deterministic Policy Gradien-Based Task Offloading Resource Allocation Joint Offloading. *Journal of Computer and Communications*, **12**, 152-168.
https://doi.org/10.4236/jcc.2024.126010

Received: May 29, 2024
Accepted: June 24, 2024
Published: June 27, 2024

Abstract

With the advancement of technology and the continuous innovation of applications, low-latency applications such as drones, online games and virtual reality are gradually becoming popular demands in modern society. However, these applications pose a great challenge to the traditional centralized mobile cloud computing paradigm, and it is obvious that the traditional cloud computing model is already struggling to meet such demands. To address the shortcomings of cloud computing, mobile edge computing has emerged. Mobile edge computing provides users with computing and storage resources by offloading computing tasks to servers at the edge of the network. However, most existing work only considers single-objective performance optimization in terms of latency or energy consumption, but not balanced optimization in terms of latency and energy consumption. To reduce task latency and device energy consumption, the problem of joint optimization of computation offloading and resource allocation in multi-cell, multi-user, multi-server MEC environments is investigated. In this paper, a dynamic computation offloading algorithm based on Multi-Agent Deep Deterministic Policy Gradient (MADDPG) is proposed to obtain the optimal policy. The experimental results show that the algorithm proposed in this paper reduces the delay by 5 ms compared to PPO, 1.5 ms compared to DDPG and 10.7 ms compared to DQN, and reduces the energy consumption by 300 compared to PPO, 760 compared to DDPG and 380 compared to DQN. This fully proves that the algorithm proposed in this paper has excellent performance.

Keywords

Edge Computing, Task Offloading, Deep Reinforcement Learning, Resource Allocation, MADDPG

1. Introduction

Over the past two decades, the widespread popularity of mobile devices such as smartphones and tablets has greatly contributed to the rapid development of mobile applications. With the growing usage of these devices, mobile data traffic has shown explosive and exponential growth, which has created unprecedented and significant challenges to the carrying capacity and stability of global mobile networks. In addition, the rise of emerging technologies such as virtual reality (VR) and augmented reality (AR) has placed an unprecedented demand on computing resources and network bandwidth. These technologies provide users with immersive experiences, but also necessitate high-performance hardware support and extensive data transmission.

To solve these problems, Cloud Computing (CC) has emerged as an innovative computing paradigm [1]. Cloud computing allows users to access a variety of configurable system resources such as computing resources, storage space, and applications over a shared network. The advantage of this model is the ability to rapidly reconfigure and deploy resources with minimal cost and service provider involvement.

In order to overcome these challenges, the industry is exploring new technological solutions such as edge computing, distributed computing and mobile cloud computing. These technologies aim to bring computing and storage resources closer to the user in order to reduce latency, increase bandwidth utilization and reduce energy consumption, thus providing a better user experience.

In recent years, research on MEC has been relatively active. MEC decomposes and sinks a single cloud function into multiple edge servers [2], and exploits the change of physical location to reduce the latency of task execution processing [3]. Some studies simplify MEC as deploying a portion of additional resources on a small base station [4] [5], but do not consider the immovability of the base station location, which makes the deployment of edge servers very restrictive. Most of the existing MECs are cloud-edge collaborative architectures [6], which have the advantages of high-speed processing on the cloud side as well as low-latency on the edge side [7]. For the problem of task offloading in MEC, see [8]. [8] considers edge servers with unified computational capabilities and proposes a pseudo-online task scheduling algorithm. Zhu *et al.* [9] consider a joint computation and caching framework that aims to reduce user latency and energy consumption. The authors propose a reinforcement learning algorithm based on deep deterministic policy gradients to implement computational offloading and task caching decisions for users.

In a recent study, Bowen Yu *et al.* [10] proposed a mobile edge computing framework for ultra-dense networks. The authors developed an optimization model to minimize the overall energy consumption of the device and the base station, while satisfying the constraints of the server quality requirements of the application. Liu *et al.* [11] proposed a computational offloading model based on the frog-jumping algorithm. The authors considered a multi-edge multi-device environment and optimized the offloading decision by using the weighted sum

of delay and energy consumption as the optimization objective. Chen Zhao *et al.* [12] considered a UAV-assisted MEC network environment. The authors used multi-intelligent deep reinforcement learning to optimize the trajectory of the UAV and allocate resources. A parallel deep neural network solution was also used to optimize the energy consumption and delay of the network. The results demonstrate that the algorithm enables the UAV to track user movements, thereby significantly reducing the latency and energy consumption of the system.

Sun *et al.* [13] employed Lyapunov optimization to decompose the task offloading and resource allocation problem into multiple sub-problems. The authors proposed an online energy-efficient task allocation and computational offloading strategy that considers dynamic wireless conditions and delay constraints. Zhou *et al.* [14] proposed a multi-population collaborative elite algorithm based on a genetic algorithm. The authors constructed a directed acyclic graph model of the end-user's application to facilitate task offload scheduling, with the objective of minimizing the delay and energy consumption of the program. Chen *et al.* [15] proposed a deep reinforcement learning algorithm based on DDPG. The authors pruned, compressed and retrained a deep neural network using filter pruning and tensor decomposition, and subsequently applied the trained neural network model to the DDPG algorithm with the objective of minimizing user energy consumption and total delay.

Zhao *et al.* [16] have investigated a task offloading strategy for UAV-assisted edge computing environments. The authors minimize the sum of execution delay and energy consumption as an optimization objective. They then design UAV trajectory, task assignment and communication resource management to solve the task offloading problem. Finally, they propose a twin-delay depth deterministic policy gradient algorithm based on twin-delay depth to obtain the global optimal policy. Tong *et al.* [17] propose a potential game and Lagrange multiplier hair-based scheme for offloading end-user tasks and allocating computational resources to MEC servers deployed on near-orbital Earth satellites in a star-Earth network environment.

Yeganeh *et al.* [18] proposed a Q-Learning scheduling strategy for task offloading. The authors implemented offloading and resource scheduling for MEC with minimizing execution time and energy consumption as optimization objectives. Zaman *et al.* [19] proposed a lightweight mobility prediction and offloading framework to overcome the user's mobility problem during task delivery. The authors proposed a server selection algorithm based on a multi-objective genetic algorithm to jointly optimize the latency and energy consumption and resource utilization efficiency of MEC servers. Wu *et al.* [20] proposed a deep reinforcement learning-based online task scheduling algorithm for online task offloading. The authors considered setting the edge nodes as public and private nodes.

2. System Modelling and Problem Description

This paper first introduces the system architecture from the perspective of

communication, computation and energy consumption, including communication model, computation model and energy model, and then gives the problem description.

2.1. System Modelling

Figure 1 illustrates the multi-cell, multi-user, multi-server MEC system environment constructed in this study. The system consists of FNV controllers, end users, base stations and MEC servers deployed at the base stations. Each end-user in this MEC system periodically generates tasks. The FNV controller is the control centre of the cell MEC system and can monitor the resources of the MEC in that cell. MECs have more computing power than end users. The MEC server of a cell can make further requests to the FNV servers of other cells to be executed by the MEC servers of other cells and cannot be offloaded to the MEC.

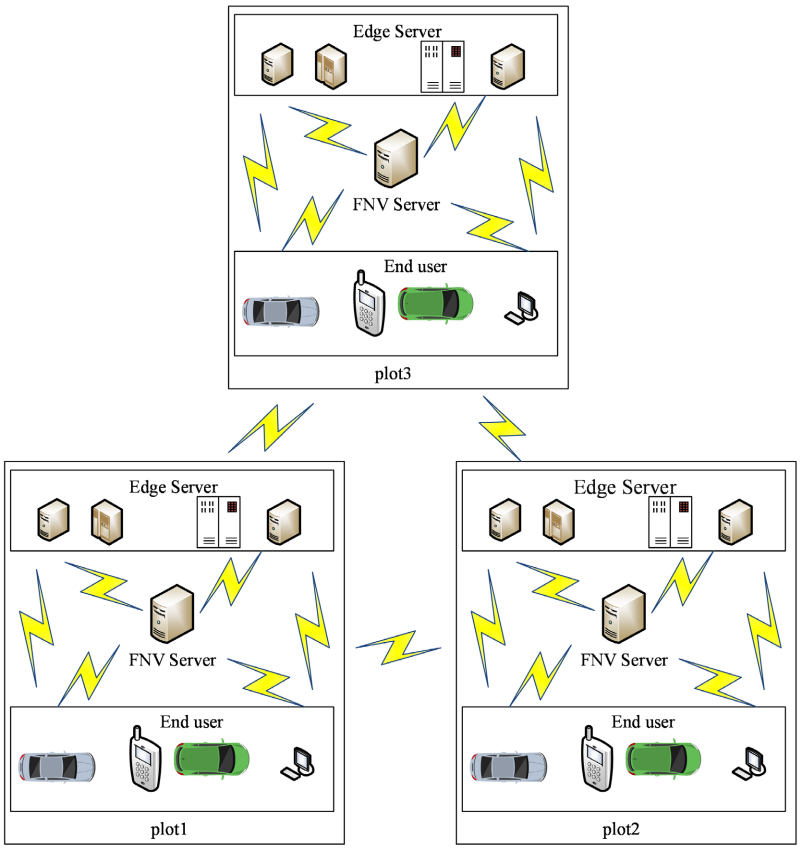

Figure 1. MEC system model with multi-cell, multi-user and multi-MEC servers.

1) Communication models

In the system model constructed in this research, the decision cycle is divided into a number of time slots t, where $t \in T$, $T = [1, 2, \cdots, T_{max}]$. All MEC servers in a cell can provide services to the users in the cell, but each end-user can only send tasks to one MEC server per time slot. The coordinates of the end users are defined as $L_{u_{i,j}}(t) = \left[x_{u_{i,j}}, y_{u_{i,j}}\right]$ and the coordinates of the MEC servers are de-

fined as $L_{MEC_{i,k}} = \left[x_{MEC_{i,k}}, y_{MEC_{i,k}} \right]$. The system has s cells with b end users and m MEC servers in each cell. In this paper it is assumed that the location of the end users within the cell is random, but the MEC servers in the cell are fixed.

This study considers the use of Orthogonal Frequency Division Multiple Access (OFDMA) to implement the communication between users and MEC servers, and the MEC server links between different cells are connected by optical fibres. Next, assume that the kth MEC server is selected for the task. According to Shannon's formula, terminal $u_{i,j}$ transmits the task at a rate of

$$Rate_{i,j}(t) = W_{i,j} \log_2 \left(1 + \frac{p_{i,j}^u h_{i,j}}{\sum_{c=1, c \neq j}^{b} p_{i,c}^u h_{i,c} + \sigma^2} \right) \tag{1}$$

where W is the bandwidth of the subchannel, $p_{i,j}^u$ and $h_{i,j}^z$ are the received power and channel gain of the device in the channel, respectively. σ is Gaussian noise.

The communication link in the system is assumed to be a line-of-sight link channel control, the channel gain can be expressed as:

$$h_{i,j}^z = \frac{P_{i,j}^{up}}{\left\| L_{MEC_{i,k}} - L_{u_{i,j}}(t) \right\|} \tag{2}$$

1) Computational modelling

A partial offloading model is considered in this study. The user offloads part of the computational tasks to the MEC server and further to other cells according to the offloading policy. Meanwhile, due to the use of optical fibre transmission between MEC servers, which has a high transmission rate that can reach 201.6 Gbps [15]. The system model in this study ignores the task backhaul delay and the inter-cell data transmission delay. It is assumed that the task selects the kth MEC server for offloading. The key components of the system experiment during offloading can be calculated as follows:

1) The time taken by the end user to transmit the task to the MEC server;

2) The time delay for the end user to process the task;

3) The time delay for the MEC in that cell to process the task;

4) The time delay for MECs in other cells to process the task.

The time at which the end user transmits the task to the MEC server can be expressed as

$$T_{i,j}^{tra}(t) = \frac{\left(1 - R_{i,j}^{rm}(t)\right) c_{i,j}(t)}{Rate_{i,j}(t)} \tag{3}$$

where $c_{i,j}(t)$ denotes the size of the task generated by the end user. $R_{i,j}^{rm}(t)$ denotes the end-user offload rate.

The end user processing delay is can be denoted as:

$$T_{i,j}^{L}(t) = \frac{\left(1 - R_{rm}(t)\right) c_{i,j}(t) z}{f_{i,j}^{L}} \tag{4}$$

where z is the required calculation frequency per task unit. $f_{i,j}^{L}$ is the computation frequency of the user.

The delay of this cell MEC processing task can be expressed as

$$T_{i,j,v}(t)=\frac{\left(1-R_{i,j}^{rs}(t)\right)R_{i,j}^{rm}(t)c_{i,j}(t)z}{\theta_{i,j}(t)\,f_{i,k}} \tag{5}$$

where $\theta_{i,j}(t)$ denotes the proportion of computational resources of the MEC server to which the user is assigned, and $f_{i,k}$ the computational frequency of the MEC to which it is assigned. $R_{i,j}^{rs}(t)$ denotes the proportion of tasks transmitted to other cells.

The delay of the MEC server in the other cell to process the task can be expressed as:

$$T_{i,j,ov}(t)=\frac{R_{rs}(t)R_{rm}(t)c_{i,j}(t)}{\theta_{i,l}'(t)\,f_{i,k}} \tag{6}$$

where $\theta_{i,l}'(t)$ denotes the proportion of computing resources allocated to the task by MECs in other cells.

In this paper, it is assumed that the MEC server can only execute the task after the transmission is completed. At the same time, the transmission task starts when the end device starts executing the local task. At this time, the delay of each task is the maximum of the end-user computational delay, the computational delay of the MEC server and the transmission delay, which can be expressed as:

$$T_{i,j}(t)=\max\left(T_{i,j}^{L}(t),T_{i,j}^{tra}(t)+T_{i,j,v}(t),T_{i,j,ov}(t)\right) \tag{7}$$

2) Power model

The battery capacity of the MEC Server is indicated as $E_{i,k}^{battery}$. The MEC server consumes energy when serving end users and can be recharged with a charging efficiency of $E_{i,k}^{charge}$. Suppose the task selects the kth MEC server for offloading. The main components of the power consumption are as follows

1) Energy consumed by the end user
2) Energy consumed by the MEC server in the cell
3) Power consumed by other MECs in the cell
4) Power consumed by task transfer to the MEC server
5) Energy consumed when the MEC receives the task.

The energy consumption of the end user can be expressed as

$$E_{i,j}^{L}(t)=P_{i,j}^{L}T_{i,j}^{L}(t) \tag{8}$$

where $P_{i,j}^{L}$ denotes the power of the end user.

The energy consumption of the MEC server in this cell can be expressed as:

$$E_{i,j,v}(t)=P_{i,k,v}T_{i,j,v}(t) \tag{9}$$

where $P_{i,k,v}$ denotes the power of the selected MEC.

The energy consumption of the task transfer to the MEC server can be expressed as:

$$E_{i,j}^{tra,up}(t) = P_{i,j}^{up} T_{i,j}^{tra}(t) \tag{10}$$

where $P_{i,j}^{up}$ denotes the transmission power of the end user.

The energy consumption at the time of MEC receiving task can be expressed as:

$$E_{i,j}^{tra,down}(t) = P_{i,k}^{down} T_{i,j}^{tra}(t) \tag{11}$$

In terms of the MEC charging model, the residual of MEC at moment t can be expressed as:

$$E_{i,k}^{r}(t) = \begin{cases} E_{i,j}^{v}(t), & E_{i,j}^{v}(t) \le E_{i,k}^{battery} \\ E_{i,k}^{battery}, & E_{i,k}^{battery} > E_{i,j}^{v}(t) \end{cases} \tag{12}$$

Among others,

$$E_{i,j}^{v}(t) = E_{i,k}^{r}(t-1) - E_{i,j}^{tra,down}(t-1) - E_{i,j,v}(t) + E_{i,k}^{charge} \tag{13}$$

In summary, the total energy consumption of this system can be expressed as:

$$E(t) = \sum_{i=1}^{s}\sum_{j=1}^{b} E_{i,j}^{tra,down}(t) + E_{i,j}^{tra,up}(t) + E_{i,j,ov}(t) + E_{i,j,v}(t) + E_{i,j}^{L}(t) \tag{14}$$

3) MEC storage model

In this paper, we consider a type of MEC server with limited memory resources. However, the offload rate of tasks transmitted from other users to other cells is 0. Therefore, the residual power of the MEC at time slot t can be expressed as follows:

$$M_{i,k}^{r}(t) = \begin{cases} M_{i,k}^{memory}, \\ M_{i,k}^{r}(t), & M_{i,k}^{r}(t) \le M_{i,k}^{memory} \end{cases} \tag{15}$$

Among others,

$$M_{i,k}^{r}(t) = M_{i,k}^{r}(t-1) - \left(1 - R_{i,j}^{rs}(t)\right) R_{i,j}^{rm}(t) c_{i,j}(t) + \sum_{j=1}^{b} \theta_{i,j} f_{i,k} st \tag{16}$$

2.2. Problem Description

In this paper, we study the optimisation objective of minimising the weighted sum of the system energy consumption and the average delay of the users. The joint optimisation problem can be expressed as:

$$\min_{\theta_{i,j}(t),\theta'_{i,j}(t),R_{i,j}^{m}(t),R_{i,j}^{rs}(t),R_{i,j}^{ism}(t),R_{i,j}^{iss}(t),t\in T} \sum \rho_1 E(t) + \rho_2 T(t) \tag{17}$$

$$\text{s.t. } [0,0] \le L_{u_{i,j}}(t) \le \left[L_{max,x}, L_{max,y}\right] \tag{18}$$

$$0 \le E_{i,k}^{r}(t) \le E_{i,k}^{battery} \tag{19}$$

$$1 \le R_{ism}(t) \le m,\ R_{ism}(t) \in R, \quad 0 \le R_{rm}(t) \le 1 \tag{20}$$

$$1 \le R_{iss}(t) \le s,\ R_{iss}(t) \in R, \quad 0 \le R_{rs}(t) \le 1 \tag{21}$$

$$0 \le M_{i,k}^{r}(t) \le M_{i,k}^{memory} \tag{22}$$

$$0 \le \theta_{i,j}(t) \le 1, \sum_{j=1}^{b} \theta_{i,j}(t) \le 1 \tag{23}$$

$$0 \le \theta'_{i,j}(t) \le 1, \sum_{j=1}^{b} \theta'_{i,j}(t) \le 1 \tag{24}$$

$$T_{i,j}(t) \le T_{i,j,max}(t) \tag{25}$$

where Equation (18) indicates that the location of the user in the cell cannot exceed the range of that cell, and Equation (19) indicates the energy limit of the MEC. Equation (20) indicates that the offloading rate is between 0 and 1 and the number of offloaded MECs cannot exceed the number of MEC servers in the cell. Equation (21) indicates that the offload rate is between 0 and 1 and the number of offloaded cells cannot exceed the total number of cells. Equation (22) indicates a range of memory resources for the MEC server. Equation (23) denotes a range in which the sum of the allocation ratios of the arithmetic resources and the ratios cannot exceed 1. Equation (24) represents a range where the sum of the allocation ratios of the arithmetic resources to the tasks passed from other cells cannot exceed 1 as well as the ratio. Equation (25) indicates that the delay of the task cannot exceed the maximum delay that the user can receive. After many experiments, $\rho_1 = 0.53$, $\rho_2 = 0.007$ can better guide the model to find the optimal policy.

3. MADDPG-Based Dynamic Computational Offloading Algorithm

This study proposes a dynamic computational offloading algorithm based on Multi-Agent Deep Deterministic Policy Gradient (MADDPG) to obtain the optimal policy. The algorithm will be described in detail in the following section.

3.1. Markov Decision-Making Process

The process of determining the optimal offloading strategy is modelled as an MDP problem in this study. The DRL algorithm is then employed to identify the optimal optimization strategy. The state space, action space and reward of the Markov decision process are presented in the subsequent section.

State space: the amount of tasks $c(t)$ and the location $L_u(t)$ of the user for each cell user in the cell, the remaining power $E^r(t)$ and the storage resource $M^r(t)$ of the MEC as the current state. Thus the state space can be represented as Equation (26):

$$s(t) = \left\{ c(t), L_u(t), E^r(t), M^r(t), M_{i,k}(t) \right\} \tag{26}$$

Action space: this study considers offloading to a specific MEC server $R_{i,j}^{ism}(t)$, offloading rate $R_{i,j}^{rm}(t)$, whether to further offload to MEC servers $R_{i,j}^{ism}(t)$ and offloading rate $R_{i,j}^{rm}(t)$ in other cells as the actions of the intelligences. Therefore the action space can be expressed as Equation (27):

$$a_{i,j}(t) = \left\{ R_{i,j}^{ism}(t), R_{i,j}^{rm}(t), R_{i,j}^{iss}(t), R_{i,j}^{rs}(t) \right\} \tag{27}$$

Reward: The optimization objective of this study is to minimize the weighted sum of the total delay $T(t)$ and the total energy consumption $E(t)$. The reward can be expressed as follows. Therefore the reward can be expressed as follows Equation (28).

$$r(t) = -\left(\rho_1 E(t) + \rho_2 T(t)\right) \tag{28}$$

where, ρ_1 and ρ_2 denote the weights of energy consumption and delay respectively.

3.2. Resource Distribution

In this study, MEC server resources will be allocated after computational offloading. Effective allocation of resources can optimize the performance of the MEC system. This study considers the allocation of arithmetic resources.

The allocation of arithmetic resources refers to the allocation of arithmetic resources uploaded to the MEC server under the management of the NFV controller, with the main purpose of more fully meeting the needs of end users. The main purpose of the allocation of arithmetic resources is to reduce the processing delay of the user, and this study assumes that a user will only use one channel within a timestamp. Derivation of $\theta_{i,j}$ in $T_{i,j,v}$ shows that:

$$\frac{\partial T_{i,j,v}}{\partial \theta_{i,j}} = -\frac{\left(1 - R_{i,j}^{rs}(t)\right) R_{i,j}^{rm}(t) c_{i,j}(t)}{\theta_{i,j} f_{i,k}} \tag{29}$$

It can be seen in the derivation of the above equation:

$$\frac{\partial^2 T_{i,j,v}}{\partial \theta_{i,j}^2} = 2\frac{\left(1 - R_{i,j}^{rs}(t)\right) R_{i,j}^{rm}(t) c_{i,j}(t)}{\theta_{i,j}^3 f_{i,k}} \geq 0 \tag{30}$$

Since $R_{i,j}^{rs}(t)$, $R_{i,j}^{rm}(t)$, $f_{i,k}$ and $\theta_{i,j}$ are all greater than 0, it follows that the user can only pair the resources of one channel in a given timestamp. Consequently, the Hesse matrix of channel resources at that moment is positive definite. The optimal arithmetic resource allocation can be solved by Lagrange Multiplier Method as follows.

$$\theta_{i,j}^* = \frac{\sqrt{\left(1 - R_{i,j}^{rs}(t)\right) R_{i,j}^{rm}(t) c_{i,j}(t)}}{\sum_{i=1}^{m} \sqrt{\left(1 - R_{i,j}^{rs}(t)\right) R_{i,j}^{rm}(t) c_{i,j}(t)} + \sqrt{s_{ov}}} \tag{31}$$

The optimal arithmetic allocation for tasks transmitted from other cells is thus determined:

$$\theta_{ov}^* = \frac{\sqrt{s_{ov}}}{\sum_{i=1}^{m} \sqrt{\left(1 - R_{i,j}^{rs}(t)\right) R_{i,j}^{rm}(t) c_{i,j}(t)} + \sqrt{s_{ov}}} \tag{32}$$

In this context, the symbol s_{ov} represents the size of the task assigned to the kth MEC from other cells.

3.3. MADDPG-Based Dynamic Computational Offload Algorithm

One of the main reasons why traditional reinforcement learning methods are

difficult to apply to multi-intelligent body environments is that the strategy of each intelligent body is constantly changing during the training process, leading to an unstable environment for each individual intelligent body. MADDPG [21] is a reinforcement learning algorithm designed for multi-intelligent body systems. It extends DDPG for environments where multiple intelligences are present simultaneously. The learners of each intelligence share the experience playback buffer and use the strategies of other intelligences to assist in decision making, resulting in an overall collaborative behaviour. MADDPG uses a strategy of centralised training and decentralised execution in the training process.

In MADDPG, the actions of the intelligences affect the state transfer of the environment so that the state transfer function can be described as

$$s_{t+1} = f\left(s_t, a_1, a_2, \cdots, a_N\right) \tag{33}$$

The reward function of MADDPG can be formulated as follows:

$$r_t^i = R_i\left(s_t, a_1, a_2, \cdots, a_N\right) \tag{34}$$

where A denotes the reward given to, the ith intelligent in the tth timestamp, and B denotes the reward function of the intelligent.

The value function of MADDPG can be expressed as:

$$Q_i\left(s_t, a_1, a_2, \cdots, a_N\right) \tag{35}$$

3.4. MADDPG-Based Dynamic Computational Offloading Algorithm

Algorithm 1 is a dynamic computational offloading algorithm based on MADDPG.

Algorithm 1. MADDPG-based dynamic computational offloading algorithm.

MADDPG-based joint offloading algorithm (MADDPGDCO algorithm)

1: Input: initial position of MEC $L_{MEC_{i,k}}$,initial power $E_{i,j}^{v}\left(t\right)$,initial amount of data for computational task $c_{i,j}\left(t\right)$

2: Initialisation: actor network parameters θ_{target}^{π} and critical network parameters θ^{Q} for all intelligences

3: Initialisation: target network parameters θ_{target}^{π} and θ_{target}^{Q}

4: Set learning rate α^{π} and α^{Q}

5: Set experience playback buffer D

6: Set batch size batch_size

7: Set γ as discount factor

8: Set the exploration noise parameter

9: For each episode:

10: For each time step t:

11: For each intelligence i:

12: Use actor network to select action $a_{i,j}\left(t\right)=\left\{R_{i,j}^{ism}\left(t\right), R_{i,j}^{rm}\left(t\right), R_{i,j}^{iss}\left(t\right), R_{i,j}^{rs}\left(t\right)\right\}$ based on current observation $s\left(t\right)=\left\{c\left(t\right), L_u\left(t\right), E^r\left(t\right), M^r\left(t\right), M_{i,k}\left(t\right)\right\}$, according to policy π_{θ}

13: If $R_{i,j}^{ism}\left(t\right)==0$ then

Continued

14: Calculate the delay of the task according to Eq. $T_{i,j}^{L}(t)=\frac{\left(1-R_{rm}(t)\right)c_{i,j}(t)z}{f_{i,j}^{L}}$ and $E_{i,j}^{L}(t)=P_{i,j}^{L}T_{i,j}^{L}(t)$ and the offloading rate A is 0. Calculate the reward value according to Equation (28).

15: else if $R_{i,j}^{ism}(t)!=0$

16: if $R_{i,j}^{iss}(t)==0$ then

17: Calculate the delay and energy consumption of the task according to $T_{i,j}(t)=\max\left(T_{i,j}^{L}(t),T_{i,j}^{tra}(t)+T_{i,j,v}(t),T_{i,j,ov}(t)\right)$ and $E_{i,j}^{v}(t)=E_{i,k}^{r}(t-1)-E_{i,j}^{tra,down}(t-1)-E_{i,j,v}(t)+E_{i,k}^{charge}$.

18: else

19: Assign to run in the corresponding MEC server and calculate the energy consumption and delay of the task according to equation $E(t)=\sum_{i=1}^{s}\sum_{j=1}^{b}E_{i,j}^{tra,down}(t)+E_{i,j}^{tra,up}(t)+E_{i,j,ov}(t)+E_{i,j,v}(t)+E_{i,j}^{L}(t)$ and equation $T(t)=\sum_{i=1}^{s}\sum_{j=1}^{b}T_{i,j}(t)$.

20: Send the action to the environment, obtain the reward $r_{i,j}$ by equation (3) and find the next state $s(t+1)$

21: Store the observation, action, reward, and next state of each intelligence in the experience replay buffer D

22: If the size of the experience playback buffer D is larger than the batch size batch_size:

23: For each intelligence i:

24 Draw a random batch sample $\left\{s(t),a_{i,j}(t),r_{i,j}(t),s(t+1)\right\}$ from D

25: Calculate the target value

26: Update the critical network parameters θ^{Q} to minimize the loss $L=\left(y_{i,j}-Q\left(s(t),a_{i,j}(t),\theta^{Q}\right)\right)^{2}$

27: Update actor network parameters using gradient descent algorithm θ^{π}

28: Update target network parameters:

29: $\theta_{target}^{\pi}=\tau\theta^{\pi}+(1-\tau)\theta_{target}^{\pi}$

30: $\theta_{target}^{Q}=\tau\theta^{\pi}+(1-\tau)\theta_{target}^{Q}$

31: Output: optimal policy

In this algorithm, the environment first interacts with each intelligence, and then the environment allocates tasks based on the actions returned by the intelligences. After all the intelligences have generated actions, the computational resources for different tasks are allocated based on the size of the tasks that are offloaded to the MEC server. Finally, the reward for each task is obtained.

4. Simulation Results and Analyses

This paper presents a series of simulation results that validate the advantages of the MADDPGDCO algorithm proposed in this study in comparison to the baseline algorithm used for simulation. The algorithm and simulations proposed in this study are written and implemented in the Python language. The environ-

ment used for simulation is PyCharm Professional 2022.3.3. In the paper, simulation experiments are carried out using the Python language through the Anaconda platform. Three square edge computing service networks with a length and width of 500 are simulated. The parameters required for the experimental simulation environment are shown in Table 1, while the parameters in the network are set out in Table 2.

4.1. Simulation Settings

In this study, the following schemes will be used as baseline algorithms for comparison tests:

1) Proximal Policy Optimization (PPO) algorithm
2) Deep deterministic policy gradient algorithm
3) Deep Q-network

Table 1. Major environmental parameters.

Parameter name	Parameter setting
Number of cells s	3
Number of users per cell	20
Maximum bandwidth for a single channel	20 MHz
Maximum power of the MEC server	1000 Wh
Total amount of memory of the edge computing server	8 GB
Upload power of end users	16 W
Maximum amount of data generated by users for a single task	2 GB
Noise index	9 dB
Computing frequency of end users	2 GHz
Computing frequency of edge computing server	4 GHz
End-user processing power	20 W
Edge computing server processing power	40 W
Edge computing server receive power	10 W

Table 2. MADDPGDOC network training parameters.

Parameter name	Parameter setting
Experience pool size	1e6
Batch size v	256
Activation function	Tanh
Optimizer	Adam
Actor network learning rate	1e−4
Critic network learning rate	1e−4
Discount factor	0.99
Update step size	10
Hidden layer size z	256
Number of training rounds e	500

The parameters required for the experimental simulation environment are shown in Table 1 and the parameters in the network are set in Table 2.

4.2. Simulation Results

This paper presents a series of simulation results to demonstrate the advantages of the MADDPGDCO algorithm proposed in the study. Firstly, the study explores the proportion of task offloading allocated to other cell MECs in the optimization strategy of the algorithm. Secondly, the simulation compares the MADDPGDCO algorithm proposed in this paper with the baseline algorithm used in the simulation in terms of delay, energy consumption and reward.

Firstly, the simulation studies the ratio of different task-based cell offloading and different cell offloading under this algorithm. As shown in Figure 1, the algorithm proposed in this paper fully achieves the joint offloading of MEC servers in different cells. From Figure 2, it can be seen that the MADDPGDCO algorithm proposed in this paper can effectively enable the MECs in different cells to process the tasks.

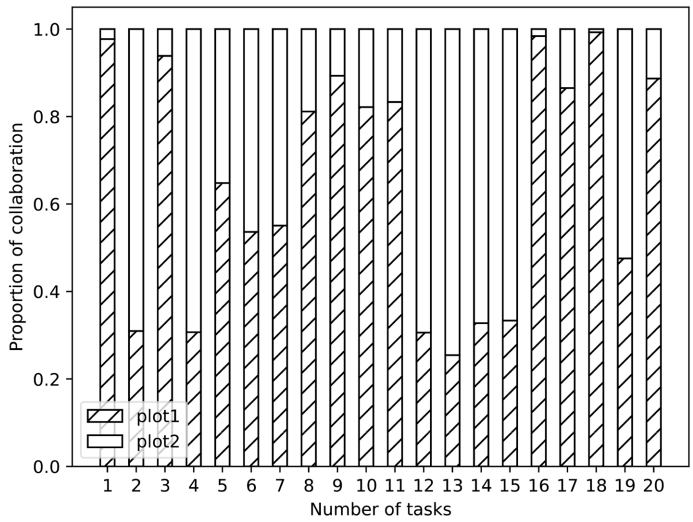

Figure 2. Proportion of tasks synergistically offloaded.

Secondly, the simulation examines the performance of delay, energy consumption and reward under different schemes. Figure 3 illustrates the delay of different schemes after convergence. It can be observed that the average delay is approximately 9.5 ms for MADDPGDCO, 14.5 ms for PPO, 11 ms for MADDPGDCO and 20.2 ms for DQN.

Figure 4 illustrates the energy consumption of different schemes following convergence. It can be observed that the average energy consumption of MADDPGDCO is approximately 600 units, PPO is approximately 900 units, DDPG is approximately 760 units and DQN is approximately 980 units.

Figure 5 illustrates the comparative reward outcomes between MADDPGDCO and the baseline algorithm. The reward plots for training with different scenarios demonstrate that MADDPGDCO outperforms the baseline algorithm.

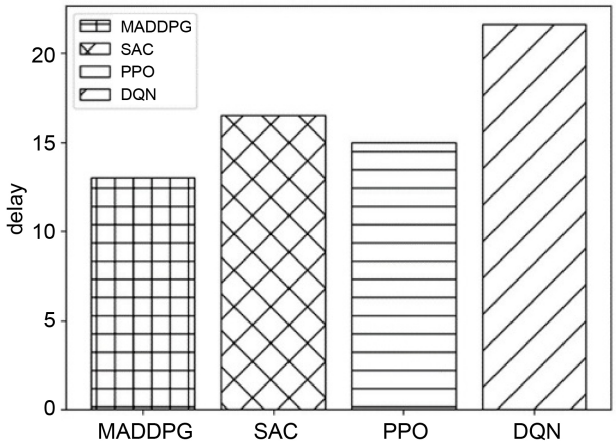

Figure 3. Comparison of latency of MADDPGDCO with baseline algorithm.

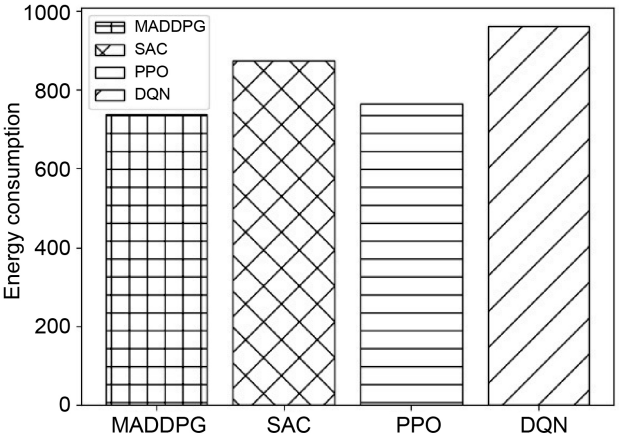

Figure 4. Comparison of energy consumption between MADDPGDCO and baseline algorithm.

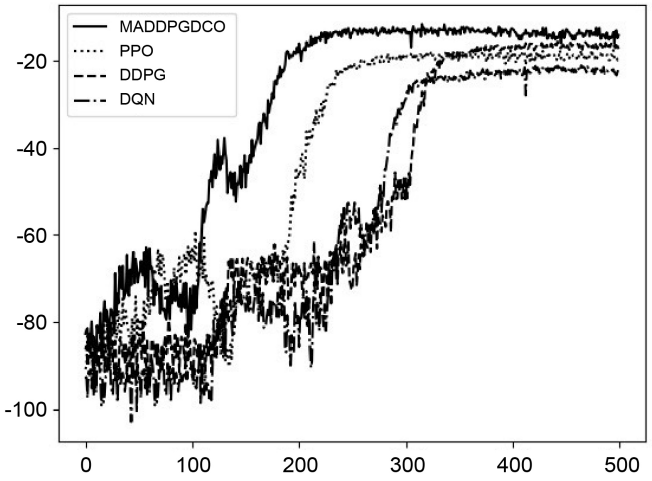

Figure 5. Comparison of MADDPGDCO and baseline algorithm rewards.

5. Conclusions

In this paper, we address the problem of joint optimization of task offloading and resource allocation for multi-cell, multi-user and multi-server. First, this paper provides a detailed analysis of channel resource allocation and determines the optimal channel allocation scheme. In terms of computational offloading, this paper works on minimizing the weighted sum of energy consumption and delay as the optimization objective. To achieve this objective, the following parameters are co-optimized: the specific MEC servers to offload to, whether to offload further to MEC servers in other cells, the offload rate, and the ratio of arithmetic resource allocation. The co-optimization of these parameters aims to develop an efficient offloading strategy. In order to evaluate the performance of the proposed algorithm, the method in this paper is compared with other delay calculation methods. The experimental results show that the algorithm proposed in this paper reduces the delay by 5 ms compared to PPO, 1.5 ms compared to DDPG and 10.7 ms compared to DQN, and reduces the energy consumption by 300 compared to PPO, 760 compared to DDPG and 380 compared to DQN, which fully proves that the algorithm proposed in this paper has excellent performance.

Although the algorithm proposed in this paper has a great advantage over the baseline algorithm, the algorithm proposed in this paper cannot adapt to the situation where the number of end users changes or the end users moves to other cells while uploading the task, and there are some limitations.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Zeng, S., Li, Z., Yu, H., Zhang, Z., Luo, L., Li, B., *et al.* (2023) Hfedms: Heterogeneous Federated Learning with Memorable Data Semantics in Industrial Metaverse. *IEEE Transactions on Cloud Computing*, **11**, 3055-3069. https://doi.org/10.1109/tcc.2023.3254587

[2] Queralta, J.P., Qingqing, L., Zou, Z. and Westerlund, T. (2020). Enhancing Autonomy with Blockchain and Multi-Access Edge Computing in Distributed Robotic Systems. 2020 *Fifth International Conference on Fog and Mobile Edge Computing* (*FMEC*), Paris, 20-23 April 2020, 180-187. https://doi.org/10.1109/fmec49853.2020.9144809

[3] Ma, X.T., Zhao, J.H., Gong, Y., *et al.* (2017) Key Technologies of MEC towards 5G-Enabled Vehicular Networks. In: Wang, L., Qiu, T. and Zhao, W., Eds., *Quality, Reliability, Security and Robustness in Heterogeneous Systems*, Springer, 153-159.

[4] Andrews, J.G., Claussen, H., Dohler, M., Rangan, S. and Reed, M.C. (2012) Femtocells: Past, Present, and Future. *IEEE Journal on Selected Areas in Communications*, **30**, 497-508. https://doi.org/10.1109/jsac.2012.120401

[5] Dhillon, H.S., Ganti, R.K., Baccelli, F. and Andrews, J.G. (2012) Modeling and Analysis of K-Tier Downlink Heterogeneous Cellular Networks. *IEEE Journal on*

Selected Areas in Communications, **30**, 550-560. https://doi.org/10.1109/jsac.2012.120405

[6] Wu, T., Xu, L. and Liu, C. (2020) Research on 5G MEC Edge Intelligence Architecture. *Information and Communication Technology*, **14**, 46-49. (In Chinese)

[7] Ge, H.B., Feng, A.Q. and Wang, Y. (2020) Offloading Strategy for Workflow Tasks in 5G Edge Computing Environment. *Sensors and Microsystems*, **39**, 130-133, 137. (In Chinese)

[8] Xu, Z., Liang, W., Xu, W., Jia, M. and Guo, S. (2016) Efficient Algorithms for Capacitated Cloudlet Placements. *IEEE Transactions on Parallel and Distributed Systems*, **27**, 2866-2880. https://doi.org/10.1109/tpds.2015.2510638

[9] Zhu, X., Jia, Z., Pang, X. and Zhao, S. (2024) Joint Optimization of Task Caching and Computation Offloading for Multiuser Multitasking in Mobile Edge Computing. *Electronics*, **13**, Article No. 389. https://doi.org/10.3390/electronics13020389

[10] Yu, B.W., Pu, L.J., Xie, Y.T., *et al*. (2018) Research on Mobile Edge Computing Task Offloading and Base Station Association Collaborative Decision-Making Problem. *Computer Research and Development*, **55**, 537-550. (In Chinese)

[11] Liu, G.Q., Zhang, X.X., Ma, H.Y. and Yan, H. (2023) A Computational Task Offloading Algorithm for Multi-Edge Node Scenarios. *Information and Control*, **52**, 679-688. (In Chinese)

[12] Chen, Z. and Gong, B.C. (2024) UAV-Assisted Two-Layer Deep Reinforcement Learning Task Offloading Algorithm. *Computer Applications Research*, **41**, 426-431. (In Chinese)

[13] Sun, Y., Wei, T., Li, H., Zhang, Y. and Wu, W. (2020) Energy-Efficient Multimedia Task Assignment and Computing Offloading for Mobile Edge Computing Networks. *IEEE Access*, **8**, 36702-36713. https://doi.org/10.1109/access.2020.2973359

[14] Zhou, H., Jiang, K., Liu, X., Li, X. and Leung, V.C.M. (2022) Deep Reinforcement Learning for Energy-Efficient Computation Offloading in Mobile-Edge Computing. *IEEE Internet of Things Journal*, **9**, 1517-1530. https://doi.org/10.1109/jiot.2021.3091142

[15] Chen, X., Ge, H., Liu, L., Li, S., Han, J. and Gong, H. (2021). Computing Offloading Decision Based on DDPG Algorithm in Mobile Edge Computing. 2021 *IEEE* 6*th International Conference on Cloud Computing and Big Data Analytics* (*ICCCBDA*), Chengdu, 24-26 April 2021, 391-399. https://doi.org/10.1109/icccbda51879.2021.9442599

[16] Zhao, N., Ye, Z., Pei, Y., Liang, Y. and Niyato, D. (2022) Multi-Agent Deep Reinforcement Learning for Task Offloading in UAV-Assisted Mobile Edge Computing. *IEEE Transactions on Wireless Communications*, **21**, 6949-6960. https://doi.org/10.1109/twc.2022.3153316

[17] Tong, M., Wang, X., Li, S. and Peng, L. (2022) Joint Offloading Decision and Resource Allocation in Mobile Edge Computing-Enabled Satellite-Terrestrial Network. *Symmetry*, **14**, Article No. 564. https://doi.org/10.3390/sym14030564

[18] Yeganeh, S., Babazadeh Sangar, A. and Azizi, S. (2023) A Novel Q-Learning-Based Hybrid Algorithm for the Optimal Offloading and Scheduling in Mobile Edge Computing Environments. *Journal of Network and Computer Applications*, **214**, Article ID: 103617. https://doi.org/10.1016/j.jnca.2023.103617

[19] Zaman, S.K.u., Jehangiri, A.I., Maqsood, T., Haq, N.u., Umar, A.I., Shuja, J., *et al*. (2022) Limpo: Lightweight Mobility Prediction and Offloading Framework Using Machine Learning for Mobile Edge Computing. *Cluster Computing*, **26**, 99-117. https://doi.org/10.1007/s10586-021-03518-7

[20] Wu, H., Geng, J., Bai, X. and Jin, S. (2024) Deep Reinforcement Learning-Based Online Task Offloading in Mobile Edge Computing Networks. *Information Sciences*, **654**, Article ID: 119849. https://doi.org/10.1016/j.ins.2023.119849

[21] Lowe, R., Wu, Y.I., Tamar, A., *et al*. (2017) Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Proceedings of the* 31*st International Conference on Neural Information Processing Systems*, Long Beach, 4-9 December 2017, 6382-6393.