Research on task offloading based on deep reinforcement learning in mobile edge environment

Xia Gao[1,*], and *Fangqin* Xu[1]

[1]College of Information Technology, Shanghai Jian Qiao University, Shanghai, 201306, China

Keywords: Mobile edge computing, Task offloading, Deep reinforcement learning, iFogSim.

Abstract. With the rapid development of Internet technology and mobile terminals, users' demand for high-speed networks is increasing. Mobile edge computing proposes a distributed caching approach to deal with the impact of massive data traffic on communication networks, in order to reduce network latency and improve user service quality. In this paper, a deep reinforcement learning algorithm is proposed to solve the task unloading problem of multi-service nodes. The simulation platform iFogSim and data set Google Cluster Trace are used to carry out experiments. The final results show that the task offloading strategy based on DDQN algorithm has a good effect on energy consumption and cost, it has verified the application prospect of deep reinforcement learning algorithm in mobile edge computing.

1 Introduction

With the rapid development of smart devices, cloud computing has been unable to meet the growing demand for data processing. On the one hand, because all the required business data need to be transmitted through the core network, it will cause a great load on the core network during the peak period of the network. On the other hand, according to the relative distance between smart devices and cloud data centers, there will be a large network delay, which seriously affects the quality of service of delay-sensitive applications. In response to the above problem, MEC(mobile edge computing) provides services by using an open platform that integrates network, computing, storage, and application core capabilities on the side close to the physical entity or data source. MEC executes its applications on the edge side, resulting in faster network service response, meeting the industry's basic needs for real-time processing, smart applications, security and privacy protection. Therefore, this paper will design a task offloading strategy based on deep reinforcement learning, which can effectively reduce the energy consumption of each service node and the cost of user payment[1].

* Corresponding author: 14080@gench.edu.cn

2 Problem modeling

2.1 MEC structure

The composition of the MEC usually consists of three parts: the cloud data center layer, the edge server layer, and the terminal device layer. As shown in Figure 1 , the terminal device layer includes various types of sensors, computers, and mobile phones having certain processing performance; The edge server layer divides all edge servers according to relative distance, and each area contains an edge server with moderate performance and heterogeneity; The cloud data center layer contains a large number of high-performance physical servers that form a cluster to serve users[2]. When the task from the mobile terminal needs to be offloaded, the whole mobile application is first divided into sub-tasks that have data interaction with each other but can be executed independently by some sorting algorithm. Some of these sub-tasks must be executed locally. Others are tasks that can be offloaded, which are usually data processing tasks with a large amount of computation. In this paper, all computing devices in the entire MEC are represented by (D, B, G), where D represents a cloud data center with massive computing resources, which is mainly composed of one physical machine cluster; $B = \{b_i | i \in [1, m]\}$ denotes a set having m edge servers; $G = \{g_j | j \in [1, n]\}$ denotes a set having n mobile terminal devices[3].

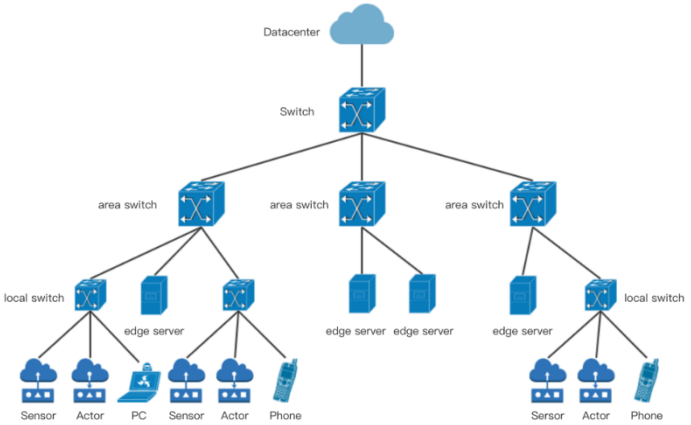

Fig. 1. MEC structure graph.

2.2 Performance metrics

In this paper, the energy consumption model and the cost model are designed to evaluate the results of task offloading strategy based on deep reinforcement learning. The algorithm model is as follows:

Power model: For the total energy consumption generated by all computing devices including smartphones and remote servers in a certain period of time, this paper first defines the power model formula of the i-th computing device as follows:

$$P_i(u) = \begin{cases} K * P_i^{max} + (1 - K) * P_i^{max} * u & if\ u > 0 \\ 0 & otherwise \end{cases} \quad (1)$$

where K represents the percentage of energy consumed by the computing device in the idle state, P_i^{max} represents the energy consumed by the i-th computing device at full load, and u is the CPU utilization. In addition, the computing device load is constantly changing with time. Now assume that $u(t)$ is a function of calculating the CPU utilization of the device

per unit time, then the computing device with a total number of (1+m+n) starting from time t_0. The total energy consumption per unit time t is:

$$Energy_{sum}(t_0) = \sum_{i=1}^{1+m+n} \int_{t_0}^{t_0+t} P_i\big(\mathrm{u(t)}\big)\, dt \tag{2}$$

Cost model: Users need to pay for computing resources provided by remote servers. In this paper, a dynamic price model based on resource surplus is used. When the resource surplus is less, the resource price will be higher. At this time, users tend to choose service nodes with lower unit price as the offloading target, so as to reduce user costs while at the same time improving resource utilization[4]. The formula is based on a dynamic price model that calculates the remaining amount of resources:

$$\mathrm{Cost} = \mathrm{CC} + \mathrm{UT} * \mathrm{RPM} * \mathrm{LU} * \mathrm{TM} \tag{3}$$

CC indicates the cost that the current device has generated, UT indicates the unit interval time for the fee calculation, RPM indicates the price set by the unit computing resource, LU indicates the computing resource ratio that the current device has used, and TM indicates the total computing resource of the current device. At the same time, since the computing resources of the local device belong to the user and do not need to be calculated as the service provided by the operator, the total cost formula of all charging devices (1+m) is:

$$Cost_{sum} = \sum_{i=1}^{1+m} Cost_i \tag{4}$$

3 Algorithm design

3.1 DDQN algorithm

The DDQN algorithm is a value iterative algorithm that combines deep learning and reinforcement learning. It uses the experience pool and the target network to solve the non-static distribution problem and the model instability problem[5]. At the same time, in order to minimize the influence of overestimation, the DDQN algorithm splits the work of selecting the optimal action and estimating the optimal action[6]. Therefore, the calculation formula of the loss function is:

$$\begin{cases} L(\theta) = \mathrm{E}[(\mathrm{TargetQ} - \mathrm{Q}(\mathrm{s}_t, \mathrm{a}_t, \theta))^2] \\ TargetQ = r_{t+1} + \gamma Q(\mathrm{s}_{t+1}, argmax_{a'} \mathrm{Q}(\mathrm{s}_{t+1}, a', \theta); \theta^-) \end{cases} \tag{5}$$

where $\mathrm{Q}(\mathrm{s}_t, \mathrm{a}_t, \theta)$ represents the output of the current network MainNet, used to calculate the Q value of the current state action pair; $Q(\mathrm{s}_{t+1}, argmax_{a'} \mathrm{Q}(\mathrm{s}_{t+1}, a', \theta); \theta^-)$ represents the output of the target network TargetNet, which is used to calculate the target Q value after tsaking the optimal value action.

3.2 MDP model

State space: In order to comprehensively consider the characteristics between MEC neutron tasks and server resources, this paper defines the state space at time step t as $S_t = (\mathrm{M}, U_1, U_2, \ldots, U_i, \ldots, U_{2+m})$, where M Indicates the CPU resources required for this subtask deployment; U_i represents the CPU utilization of the i-th computing device at time step t. At the same time, in order to ensure that the sub-task can only be selected to be executed on the local mobile terminal device or the remote server, only (2+m) calculations need to be

considered for the sub-task[7]. The device contains one cloud data center, one local device and m edge servers.

Action space: In order to offload the subtask to the appropriate computing device, the action space is specified to correspond to the set of available computing devices in the deep reinforcement learning, and $(0/1)_i^j$ is used to indicate whether the i-th sub-task is offloaded to the jth computing device[8]. Therefore, for a cluster containing (2+m) computing devices, the action space is $A = (a_1, a_2, \dots, a_{2+m})$.

Reward function: This paper will consider the energy consumption and cost of all equipment in the MEC to evaluate the advantages and disadvantages of the offloading decision. The calculation formula of the reward function is:

$$\begin{cases} \mathrm{R} = \alpha \sum_{i=1}^{1+m+n} energy_i + \beta \sum_{i=1}^{1+m} \mathrm{cost}_i \\ \alpha + \beta = 1 \end{cases} \tag{6}$$

where $energy_i$ and cost_i represent the energy consumption and cost of the i-th device, respectively; α and β represent the weights they occupy, and their sum is 1. In addition, since the mobile terminal device belongs to the user privately, the module deployed on the local device does not calculate the fee, therefore $\sum_{i=1}^{1+m} \mathrm{cost}_i$ means that only the total cost of remote services provided by the cloud data center and edge servers is calculated[9].

4 Simulation experiment

4.1 Simulation environment

This paper uses iFogSim to simulate the MEC-based task unloading problem, and compares the energy consumption and cost of each algorithm in large-scale heterogeneous clusters to reflect the advantages and disadvantages of the offloading decision[10]. The algorithms implemented include a policy Mobile based on local device prioritization, a policy Edge based on edge server prioritization, a policy DQN based on deep reinforcement learning, and a policy DDQN based on improved deep reinforcement learning. The equipment cluster simulated by the simulation experiment mainly includes a cloud data center, 60 edge servers and a number of mobile terminal devices, in which all edge servers are equally divided into 10 different regions, and each mobile terminal device can only send one application offload request at the same time. In this paper, the corresponding simulation device configuration and average performance-to-power ratio are set with reference to SPEC (Standard Performance Evaluation Corporation). The larger the value, the less energy the device consumes under the same performance. The detailed information is shown in Table 1.

In order to simulate the offloading process after the mobile application is split into different subtasks, this paper constructs a sub-task dependency of online shopping. As shown in Figure 4, the application is mainly by edge, front end, login, accounts, orders, shipping. Subtasks such as catalogue, cart, and payment, where edge must be executed on the mobile device, and the remaining subtasks can be selected based on the decision whether to offload. Online shopping applications usually have very high requirements for offloading decisions. On the one hand, they need to offload high-computing data processing modules to remote servers to minimize the energy consumption of mobile devices. On the other hand, computing modules need to be as close as possible to data sources to reduce the delay caused by data transfer between modules. In this paper, the parameters of all deep reinforcement learning algorithm models are uniformly set to ensure the fairness of training results. The memory space size of deep reinforcement learning is defined as M=100000, the learning rate of optimization algorithm SGD is α=0.005, and the batch learning size is K=32. The update period of the target network parameter is C=50, and the discount coefficient γ=0.9.

Table 1. Computing device detailed configuration table.

Model	Type	CPU Frequency (MHZ)	Cores	performance-to-power ratio	Unit
RX4770 M4	Datacenter	2100	112	12828	0.05
RX350 S7	Edge server	2200	16	5035	0.01
DL325 Gen10	Edge server	2000	32	8083	0.01
DL360 Gen10	Edge server	2500	28	11550	0.01
TX120	Local device	2666	2	454	0
TX150 S5	Local device	2666	2	356	0
TX150 S6	Local device	2400	4	667	0

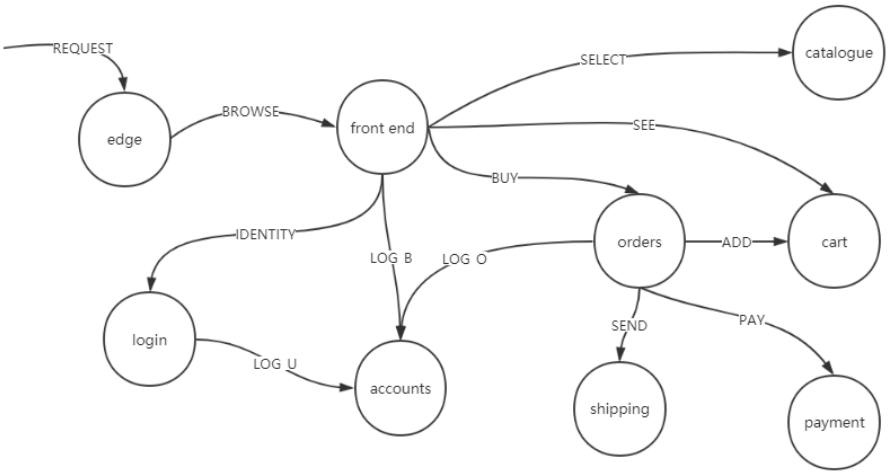

Fig. 2. Network shopping subtask dependency graph.

4.2 Analysis of experimental results

In order to reflect the changes in resource utilization of mobile applications in different time periods, this paper uses Google Cluster Trace dataset to simulate the change of each module utilization over time. In addition, in order to ensure that the strategies generated by each deep reinforcement learning algorithm are highly efficient, this paper first selects some data from the Google dataset to train each neural network, then select other data to test the trained network model to compare the versatility and efficiency of each strategy. Figure 3 shows the loss function scores of the small batch samples of the DQN algorithm and the DDQN algorithm during training. The smaller the loss function, the better the network model results. It can be seen from the figure that the DDQN algorithm has faster convergence speed and higher stability than the DQN algorithm under the same parameters. Figure 4 and Figure 5 is resource loss graphs generated by each algorithm in task offloading. As the number of applications increases, the offloading strategies generated by each algorithm increase in energy consumption and cost. Among them, the offloading strategy based on Mobile algorithm achieves good results in cost, while the performance in energy consumption is general. The offloading strategy based on Edge algorithm performs generally in terms of energy consumption and cost; because DQN algorithm converges slowly and is unstable, the offloading strategy generated by DQN algorithm performs worst in terms of energy

consumption and cost when the training process only iterates 300 times; the offloading strategy based on DQN algorithm performs better than DQN algorithm in terms of energy consumption and cost. In terms of energy consumption, it consumes the least of all the algorithms, and is second only to Mobile in terms of cost.

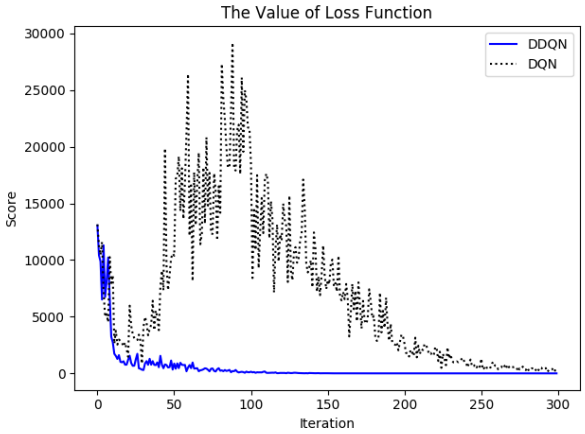

Fig. 3. Algorithmic loss function graph.

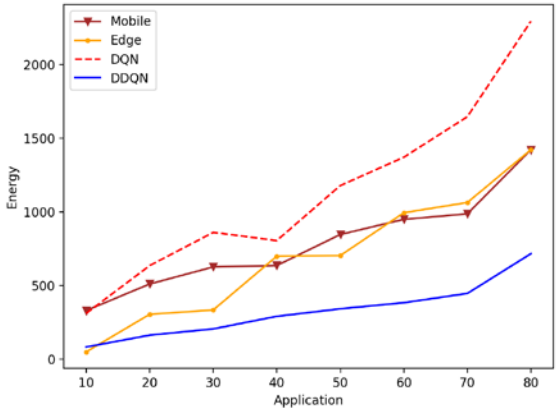

Fig. 4. Energy consumption graph of each algorithm.

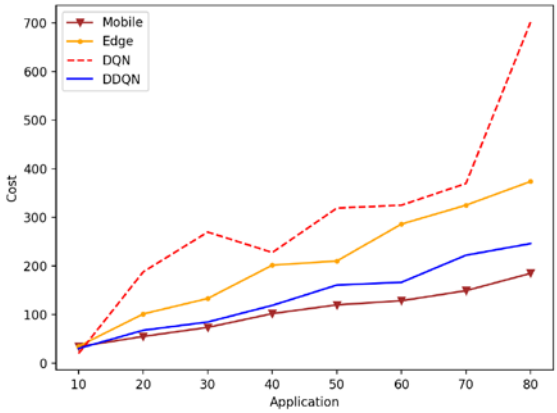

Fig. 5. Cost graph of each algorithm.

5 Summary

In this paper, deep reinforcement learning is proposed to solve the task unloading problem in mobile edge computing. Experiments on iFogSim simulation platform verify the advantages and disadvantages of each algorithm in terms of energy consumption and cost. According to the experimental results, the unloading strategy based on DDQN algorithm has the best comprehensive performance compared with Mobile algorithm, Edge algorithm and DQN algorithm, which proves the application prospects of deep reinforcement learning algorithm in mobile edge computing.

Reference

1. Roman R, Lopez J, Mambo M. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges[J]. Future Generation Computer Systems, 2018, 78: 680-698.
2. Li H, Ota K, Dong M. Learning IoT in edge: Deep learning for the Internet of Things with edge computing[J]. IEEE Network, 2018, 32(1): 96-101.
3. Chen M, Hao Y. Task offloading for mobile edge computing in software defined ultra-dense network[J]. IEEE Journal on Selected Areas in Communications, 2018, 36(3): 587-597.
4. Tran T X, Pompili D. Joint task offloading and resource allocation for multi-server mobile-edge computing networks[J]. IEEE Transactions on Vehicular Technology, 2018, 68(1): 856-868.
5. Haifeng Lu, Chunhua Gu, Fei Luo, Weichao Ding, Xinping Liu, Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning,Future Generation Computer Systems,2019
6. Lyu X, Tian H, Ni W, et al. Energy-efficient admission of delay-sensitive tasks for mobile edge computing[J]. IEEE Transactions on Communications, 2018, 66(6): 2603-2616.
7. Bi S, Zhang Y J. Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading[J]. IEEE Transactions on Wireless Communications, 2018, 17(6): 4177-4190.
8. Yang C, Liu Y, Chen X, et al. Efficient mobility-aware task offloading for vehicular edge computing networks[J]. IEEE Access, 2019, 7: 26652-26664.
9. Gupta H, Vahid Dastjerdi A, Ghosh S K, et al. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments[J]. Software: Practice and Experience, 2017, 47(9): 1275-1296.