Multitasks Scheduling in Delay-Bounded Mobile Edge Computing

Longyu Zhou(✉), Supeng Leng, Ning Yang, and Ke Zhang

University of Electronic Science and Technology of China, Chengdu, Sichuan, China
zhoulyfuture@outlook.com

Abstract. The Mobile Edge Computing (MEC) is a very novel technology in the social network. In order to satisfy the users' delay and enhance data security and reduce energy consumption of the system. In this paper, we design an optimized strategy of edge servers chosen by reinforcement learning (AI algorithm) for resource consumption of multi-edge server. We transform the objective function into a convex optimization problem for the single edge server, and give proofs. In this scenario, it is made up of multi-users with multi-tasks and the same type of edge servers deployed on the edge of the base stations, the edge servers allocate computing resources to users. Meanwhile, the users send different tasks to the edge servers to complete computing tasks. In order to complete users' requirements under their delay-bounded, we design an optimal path algorithm named Betweenness Centrality Algorithm (BCA) to reduce the transmission delay and we use Fuzzy Logical algorithm to classify computing tasks. We propose a resource allocation mechanism under satisfying delay-bounded. Finally, comparing to the random selected strategy and other schemes, we prove the effectiveness of the introduced algorithm, which reduces the energy consumption by 20% approximately for the single edge server, and the simulation proves that the Double Deep Q-learning (Double DQN) algorithm shows better performance than Random Selected Strategy for the multi-edge servers.

Keywords: Mobile edge computing · Energy consumption · Reinforcement learning · Convex approximation · Tasking scheduling

1 Introduction

With the development of 5G technology and the popularity of the intelligent terminals, kinds of application services come into being. Meanwhile, users are stricter in the service quality and delay for requesting. In the case of smart devices with AI, Cartner predicts that the number of smart devices supporting AI will increase from 10% in 2017 to 80% in 2020, it is obvious that smart mobile devices will become the one of the most popular artificial intelligence platform in the world [1]. In December 2017, Google launched android 8.1 which support the Neural Network API(NNAPI), it provides faster hardware computing ability for machine learning [2]. To serve the delay-sensitive applications, the Mobile Edge Computing (MEC) architecture has been proposed. In MEC, edge servers will be addressed locally, and are consequently more efficient in delay-sensitive applications.

S. Han et al. (Eds.): AICON 2019, LNICST 287, pp. 224–237, 2019.
https://doi.org/10.1007/978-3-030-22971-9_19

MEC is a distributed cloud system, which refers to the deployment of computing resources at the edge network [3]. Because mobile devices are short in the resource computing, the terminal device hands over some or all computing tasks to the cloud, the mobile device decides how to offload, how much to offload and what to offload. The resource allocation focuses on how to allocate resources after offloading. In fact, the MEC can be regarded as a cloud service platform which is running on the edge network. By deploying resources such as computing, storage, networking and communications at the edge of the mobile network [4]. To better satisfy the users' delay, we get the lower energy consumption, the link selection and computing cost are suggested to be jointly scheduled [5].

There are some recent papers about the optimization of the computing cost in the MEC system. In [6], the authors study ways to minimum the delay of users in the MEC system by joint optimization of tasks scheduling and resource allocation. In [7], authors propose an online algorithm that decides the local execution and computation offloading policy is developed based on Lyapunov optimization. This method optimizes the power consumption of edge servers. In [8], the authors consider a scenario of Vehicular Edge Computing (VEC), they adopt a Stackelberg game theoretic approach to design an optimal multilevel offloading scheme, which maximizes the utilities of both the vehicles and the computing servers. In [9], the authors study the offloading and auto-scaling in MEC system. They propose an efficient reinforcement learning-based resource management algorithm, which learns on-the-fly the optimal policy of dynamic workload to minimize the long-term system cost. In [10], the authors present a novel framework for offloading computation tasks, from a user device to a server hosted in the mobile edge (ME) with highest CPU availability. In [11] and [12], in order to jointly computation offloading and content caching strategies, the authors transform the original problem into a convex problem and find the optimal solution of approximated problem. In [13], the authors provide an architecture of centralized cloud and distributed MEC over hybrid fiber-wireless network in MEC system, which has the features of supporting diverse network techniques, easy expansibility, high capacity and reliability, low latency and energy consumption. They solve the question by an approximation collaborative computation offloading scheme. They consider the shortcomings of single networking mode, high congestion, high latency and energy consumption, but the proposed algorithms are hard to be implemented.

In this paper, we study into how the tasks route to destinations and how the computation resource in the edge server is allocated to users. Considering the limited resource utilization of edge servers and the data-offloaded with bounded delay requirements, so how to utilize the resource of edge servers needs to be paid attention. If an edge server's tasks are too much in a period of time, it will perform a terrible effectiveness. We propose the data route scheme that users can chose a better path to upload data and the chosen edge server offloads data through the same path, this scheme undoubtedly produces collision, so we solve this problem by RTS/CTS scheme. When a user wants to upload data to an edge server through several base stations, the user sends RTS to the first base station, if the base station is idle, it will send CTS to user, if not, it will not send. If a base station sends RTS to another, if it doesn't receive the ACK, data will be stored in the cache. In fact, collision is an event with probability, so we use Free Path Distribution to minimum the collision in a period

of time [14]. However, if all the edge servers are available, the edge servers allocate more computing resources to the tasks with delay bounds, and users can upload more data to edge servers. Meanwhile, other users maybe have the same result that this user want to get, the base station is only as a forwarding role without edge server, through this way, we can save computing consumptions. It is obvious that computing consumption is much than link consumption. We assume that this method obeys an exponential distribution. So far, we solve the link consumption successfully. As for the computing cost, for an edge server, different users with different sizes of data associate the edge server, we can analogy a M/M/1 queuing system. The computing ability of an edge server is directly related to the performance of CPU, we assume that every edge server has the same CPU ability. The MEC system includes multiple users, multiple base stations with edge servers and same base stations without edge servers. Our goal is to minimum the energy cost for the all MEC system [12].

The rest of paper is organized as follows. Section 2 introduces the system model and formulation. In Sect. 3, we study the multiple tasks scheduling schemes for single edge server. In Sect. 4, we study the multiple tasks scheduling for the MEC system. In Sect. 5, we express the Double Deep Q-learning algorithm. In Sect. 6, we show the numerical result through our analysis, and we display the relationship between bounded delay and energy consumption. We conclude in Sect. 7.

2 System Model and Formulation

We consider that a MEC system include multiple users, base stations with computing ability and base stations without computing ability, users communicate with base stations by wireless, in other words, the MEC system is a wireless network. When the computing task is done, the results are sent to users. The MEC system is shown in Fig. 1.

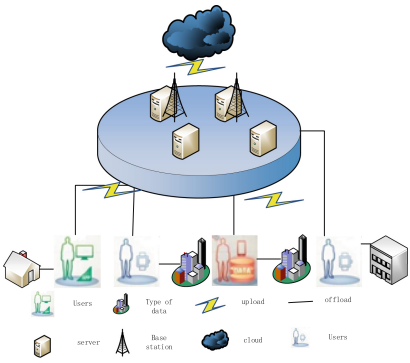

Fig. 1. System model includes that users, base stations with edge servers and base station without edge servers.

2.1 Deployment Model

In this paper, there are M base stations which make up a set $\mathbf{a} = \{\mathbf{1}, \mathbf{2}, \ldots \mathbf{M}\}$, we assume that the number of base stations is larger than the number of edge servers, there are K edge servers in MEC system, which make up a set $\mathbf{b} = \{\mathbf{1}, \mathbf{2}, \ldots \mathbf{K}\}$, and there are N users in a MEC system, which make up a set $\mathbf{c} = \{\mathbf{1}, \mathbf{2}, \ldots \mathbf{N}\}$. We apply the optimal path algorithm to help user find destination. During transmission, if path only has one hop, we use direct transmission strategy. As long as the count of hop is greater than 1, we take Betweenness centrality algorithm to get the better path routing.

The scheme can be displayed through connected undirected graph $G(V, E)$. For all the base station network, we assume that b_i is the i-th sensor, $P(b_i)$ denoted the probability of graph.

$$P(b_i) = \frac{1}{z} \prod_{i \in c_G} \phi_i(b_i) \tag{1}$$

Through the **Hammersley-Clifford Theorem**, the probability of graph can be expressed as the product of the potential functions, which defined on all the largest group for a graph. c_G denotes the set of the largest group, $z = \sum b \prod_{i \in c_G} \phi_i(b_i)$ denotes normalized constant. ϕ_i denotes the potential function of graph.

Proof: Because the sensors are dynamic in a graph, and the path selection only related to the previous path, so this scenario satisfies Markov random field model, so we use BCA to select a node named observation vertex, other two nodes are independent in a condition. In other words, anyone node in a graph satisfy:

$$P(X_i | X_{G \setminus i}) = P(X_i | X_{N_i}) \tag{2}$$

$X_{G \setminus i}$ denotes all the nodes except for i, X_{N_i} denotes all vertices connected to X_i. The formulation is another definition.

2.2 Transmission Model

For an edge server, we assume that l_i bits are transferred in a period of time, and we assume that $X_{t,i}$ denotes that the i-th base station delivers data to the next base station or user in a period of time. We assume that collision obeys Free Path Distribution. The probability of a collision between $X_{t,i}$ and $X_{t,i} + dX_{t,i}$ is as follows.

$$P_{t,i}(X) = \exp(-\frac{X}{\overline{\lambda}}) \tag{3}$$

$\overline{\lambda}$ denotes the average time between two collisions. We can get a conclusion that the probability obeys the exponential distribution.

As for a MEC system, the smaller probability of collision, the better. From the perspective of the edge servers, we assume the energy consumption of receiving data in a period of time is as follows:

$$r_i = \varepsilon_{elec} \times l_i \tag{4}$$

and the energy consumption of sending data in a period of time is as follows:

$$s_i = r_i + \varepsilon_{amp} \times l_i \times T^{\alpha} \tag{5}$$

In formulation (4) and (5), ε_{elec} denotes electric consumption for every bit, and ε_{amp} denotes amplifier's gain consumption while sending a single bit, and T denotes that the distance from the users to the edge server, α denotes that the index of path loss.

During the data transmission, we assume N tasks will be sent in a slot time, so the SINR (Signal to Interference plus Noise Ratio) of the i-th task is as follows:

$$r_i(t) = \frac{p_i g_i(t)}{\sum_{m=1, m \neq i}^{M} p_m g_m(t) + \delta_t} \tag{6}$$

In formulation (6), p_i denotes the power of i-th task, $g_i(t)$ denotes the i-th task's power gain in the t slot time, p_m denotes the m-th task's power, and $g_m(t)$ denotes the m-th task's power gain in the t slot time, δ_t denotes white Gaussian noise. So the i-th task's transmission rate is as follows:

$$v_i(t) = \omega_i \log_2(1 + r_i(t)), \forall i \in c, j \in b \tag{7}$$

In formulation (7), ω_i denotes the i-th task's bandwidth.

2.3 Edge Computing Model

In computing tasks of edge servers, the most consumed is resource of CPU [15]. The processing ability for each CPU is directly related to the efficiency. In other words, the more resource of CPU, the faster processing efficiency, and the more users satisfy the minimum delay. However, the resource for every CPU is limited, and the size of data is usually different. We assume that an edge server receives π tasks in a period of time, which makes up a set $\pi = \{1, 2. . .N\}$, and every edge server receives different size of task because of the size of task is rand, so we introduce Fuzzy logical algorithm to help solve. There are K edge servers, every edge server has bounded resource, we assume the bounded resource is $c_{\max}$ for every edge server. The energy consumption for the i-th edge server follows the model:

$$c_j = \sum_{i=1}^{N} fuzzy(\pi_{i,j}) \times c_{i,j} \tag{8}$$

In formulation (8), $c_{i,j}$ denotes the unit resource consumption of the i-th task from the i-th user to the j-th base station. So the i-th task's transmission rate is as follows:

$$t_i^{trans} = \frac{\pi_i}{v_i(t)} \tag{9}$$

Except for the transmission delay, the computing delay and the waiting time are also not ignored, we assume the concurrency value of one server is ϑ, the computing delay is as follows:

$$t_i^{copt} = \frac{\pi_i}{\vartheta \times QPS} \tag{10}$$

In the above formulation, QPS denotes the unit computing ability.

We assume that the arrival rate of tasks is $e_i(t)$, the unit computing ability is $c_j(t)$, so we can get the average waiting time by Queue theory:

$$t_i^{wait} = \frac{e_i(t)}{c_j(t)(c_j(t) - e_i(t))} \tag{11}$$

After discussing the delay of users, we can define the delay-bounded:

$$t_{c,k} = 2 \times t_k^{trans} + t_k^{copt} + t_k^{wait}, \forall k \in c \tag{12}$$

2.4 Problem Formulation

We assume that consumption among base stations is the same, $l_{\cos t}$ denotes the link consumption, $t_{k,\max}$ denotes the maximum delay tolerance. The link consumption for every segment is the same, in other words, we ignore the subtle difference about different link distance, our optimized goal can be written in the following form, which minimizes the cost in MEC system.

$$\begin{aligned} &\min \sum_{i \in b} c_i + \sum_{j \in a} l_{\cos t_j} \\ &s.t.\ c_i \le c_{\max}, \forall i \in b \\ &\quad t_{c,k} \le t_{k,\max}, \forall k \in c \end{aligned} \tag{13}$$

3 Tasks Scheduling for the Single-Server

For the i-th base station with edge server, the optimization objective is $\sum_{i \in b} c_i + \sum_{j \in a} l_{\cos t_j}$. In this part, we study into how the tasks of users should be scheduled so that the object is minimized.

3.1 Data Transmission

In this scenario, a user transfer data to edge server through several base stations. In other words, data is sent from one base station to another base station. Under this circumstance, probability of no collision is $1-P_{t,i}$, we assume that the delay-bounded is T_i, the number of total segments is $n = \frac{T_i}{X_{t,i}}$. To guarantee to transfer successfully, the probability of transfer successfully can be written exponential distribution:

$$P_{su} = (1 - P(X_{t,i})) \times \exp(n) \tag{14}$$

Proof: In order to ensure that data is transferred successfully between base stations, it is necessary to ensure that data is transferred successfully in each segment.

$$\begin{aligned} P_{su} &= (1 - P(X_{t,i}))^n \\ &= \exp(\log(1 - P(X_{t,i}))^n) \\ &= (1 - P(X_{t,i})) \times \exp(n) \end{aligned}$$

3.2 Computing Consumption

Through our analysis, as for the computing consumption, the size of data is not suitable by Fuzzy algorithm [16]. We introduce the computing time of queuing theory (M/M/1), we assume that the computing time obeys the exponential distribution. The energy consumption for computing is as follows:

$$c_j = \sum_{i=1}^{N} fuzzy(\pi_{i,j}) \times c_{i,j} \tag{15}$$

We convert data capacity to computing time, in other words, $\omega_{t,i}$ denotes that i-th base station's unit computing consumption in a t slot time.

Proof: according to the queuing theory, processing time obeys exponential distribution, so the time is as follows:

$$F(\omega_{t,i}) = 1 - \exp(-(\pi_{i,j} - \lambda_{i,j})\omega_{t,i})$$

$\Rightarrow \omega_{t,i} = \frac{\ln(1-F(\omega_{i,j}))}{\lambda_{i,j}-\pi_{i,j}}$, because $F(\omega_{t,i}) > 1$, we can get $\omega_{t,i} > 0$.

Because $F(\omega_{t,i})$ obeys the exponential distribution, it is obvious that $F(\omega_{t,i}) > 1$ is always established.

3.3 Optimization Energy Consumption

In a MEC system, if a user requires data to an edge server, it is likely that the edge server gets data from other users [17]. We assume that not all data is computed by edge server, in a period of time, we can optimize the energy consumption.

$$\min \lambda_{i,j} \times c_i + (\lambda_i - \lambda_{i,j}) \times l_{\cos t} \tag{16}$$

In formulation (16), we can get a conclusion that when data from users is too big, edge server can get some advice from other users, and it is a concave optimization problem.

Proof: as for the formulation (11), we can get the derivative of the $\lambda_{i,j}$. If we want to get the solution of derivation.

$$\frac{\partial F}{\partial \lambda_{i,j}} = 0$$

From computing consumption and link cost, the equation has a solution or not is related to the probability P_{su}, so we can find a suitable probability to change into a concave problem.

4 Tasks Scheduling for Multi-servers

In a MEC system, multi-edge servers need to jointly schedule to complete the computing tasks [18]. By our analysis and the model of the queuing theory, some data needs to be computed by edge servers and others can be got from other users, this is our goal to get the optimization of this problem.

4.1 Computing Consumption Analysis

For the multi-edge servers and multi-tasks, we can't get a valid scheme by transforming concave function. Thanking of the delay-bounded of users and the computing capacity of edge servers, we design an edge server selected scheme. We import a binary variance to complete the work of selecting which edge server to compute and define $\boldsymbol{f_i} = \{0, 1\}$. When an edge server is chosen, the tasks will be offloaded through the path designed by BCA to the edge server. The mathematic model is defined as follows:

$$\min_{f(j)} \sum_{j=1}^{M} f_j \times c_j, \forall j \in a \tag{17}$$

4.2 Bounded Delay for Users

In the actual MEC network, some important data usually be stored in the cache or the remote cloud, when the amount of data is increasing, the early data in the cache will be deleted because the space of cache is limited, we assume that the base stations with edge server or not is a probability event [19]. We can see it that this probability obeys Gaussian distribution, and it is very likely that users can obtain data through other users within a certain period of time, so the delay-bounded of users is redefined as follows:

$$t_{c,k} = 2 \times t_k^{trans} + \kappa_t(t_k^{copt} + t_k^{wait}), \forall k \in c \tag{18}$$

In the above formulation, κ_t denotes Gaussian distribution in t slot time.

4.3 Energy Consumption for MEC System

In this paper, our final optimized goal is minimizing energy consumption for all the MEC system. However, we should satisfy the bounded delay of users, so for all the MEC system, we assume the number of edge servers is K, the number of base stations is M. we define that all the energy consumption includes the computing consumption and link consumption. After receiving inspiration from queuing theory, we get the final option of energy consumption.

$$\begin{aligned} \min \ & \sum_{i \in b} \sum_{j \in a} f_j \times (c_i + l_{\cos t_j}) \\ & s.t. \ f_j \in \{0, 1\} \\ & c_i \le c_{\max}, \forall i \in b \\ & t_{c,k} \le t_{k,\max}, \forall k \in c \end{aligned} \tag{19}$$

For the above formulation, it is a non-linear math model, so we can't get a good result within limited time, so we use AI algorithm to solve this model and get a good strategy. Because we can't get any information about transport channel, so we use the Double Deep Q-learning algorithm that no need training sample.

5 Double DQN Analysis

In this paper, we use the Double DQN algorithm to solve the non-linear mathematic model. The core of this algorithm is that agent interact with environment, by learning Y_t^Q value to update the last true value to get the optimized result. the Y_t^Q is defined as follows:

$$Y_t^Q = R_{t+1}(s, a) + \gamma Q(s_{t+1}, \arg\min_a Q(s_{t+1}, a; \theta_t); \theta_t) \tag{20}$$

The double Q-learning error can be written as:

$$Y_t^{DQ} = R_{t+1}(s, a) + \gamma Q(s_{t+1}, \arg\min_a Q(s_{t+1}, a; \theta_t); \theta_t^{'}) \tag{21}$$

In above formulation, $R_{t+1}(s, a)$ denotes that when adapting action a and state s, the system's reward, γ denotes penalty factor, θ_t denotes a parameter updating the main network, and $\theta_t^{'}$ denotes a parameter updating the target network. The algorithm flow chart is as follows (Table 1):

Table 1. Algorithm flow chart

input number of iteration, dimension of state, action, step, penalty factor, target and main network, frequency updating. output: Q value

for i,..T, do:
get the state's eigenvector $\phi(s)$;
get Q-value of all the state, using greedy algorithm to choose action A;
execute action A, get new state s' 's eigenvector $\phi(s')$ and reward R;
update the state set;
calculating target Q-value using samples ;
updating all the parameters in Q-network;
updating the target parameters every C steps;
end for;

6 Numerical Results

In this part, the numerical results are simulated to show the optimization, we consider the data about some people's behaviors from RAWDAD, which is a community resource for archiving wireless data at Dartmouth. In Fig. 2, the x-axis denotes the bounded delay of people from RAWDAD, and y-axis denotes satisfied requirement, in other words, the number or rate of the satisfied people in a period of time, from Fig. 1, we can get that the delay between 0.1 s and 3 s. So we set the range of bounded delay from 0.1 s to 3 s. For the base station, the arrival rates of data $\lambda_i \in [0, 30]$, the range of tasks from 10 to 50, and the sizes of the tasks obeys the Gaussian Fuzzy, the mean value is 0, and the variance is 0.1.

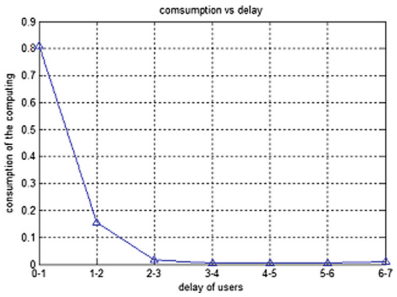

Fig. 2. The satisfied requirement under different delays

In Fig. 2, the relationship between the satisfied requirement that the satisfied or successful rate of requirement and the delay of the users. We can get some conclusions from it, the data is from the RAWDAD, and the most delays of the users are under 3 s,

so we assume that the delay between 0.1 s and 3 s can get better results. If the delay is bigger than 3 s, we assume that the high rate of people cannot stand. And we can learn that the satisfied rate focus on under 2 s. The data of MEC can help us the work for the simulation, and we can make the goal well done under the conference of the data.

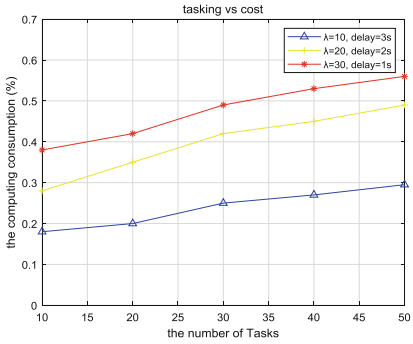

Fig. 3. The computing consumption under the number of tasks, which is compared in different delay and the Poisson stream.

In Fig. 3, the performance of four sides are compared in single-edge server. In the figure, x-axis denotes the number of tasks, which stand for the size of the queue, and y-axis denotes the rate of the computing consumption for the edge servers, which is set that the total rate is 1. When the data is required firstly, we assume that the most results should be got from the edge servers, and when the Poisson stream is large, we assume that some data can be got from edge servers and others can be got from other users. On the whole, the different side with different size is linear growth but multiple growth approximately, which proves that the optimization has a better performance. Under the bounded delay for users, we can get several conclusions from the figure. Firstly, for every trend of curve, the gradient is decreasing slowly, which proves that as the tasks are increasing, the relative computing consumption is reduced. Secondly, we can compare computing consumption under the different sides, we can get result that the algorithm improves 20% approximately. Finally, the algorithm can give us the better results from our analysis. For multi-tasks and multi-edge servers, we design the DDQN algorithm to solve, the result about multi-edge servers is as follows.

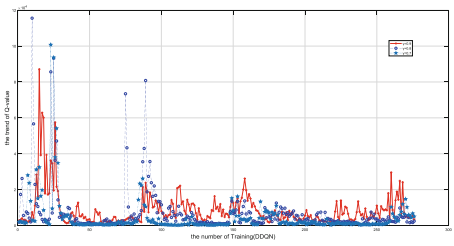

Fig. 4. Under different penalty factor, the trend of the Q-value.

In Fig. 4, the penalty factor is from 0.7 to 0.9, the x-axis denotes the step of training and the y-axis denotes the trend of the Q-value, the Q-value can be convergent in a range, but different penalty factor has different speed of convergence. From Fig. 4, we can get when the penalty factor is 0.7, Q-value has a better convergence.

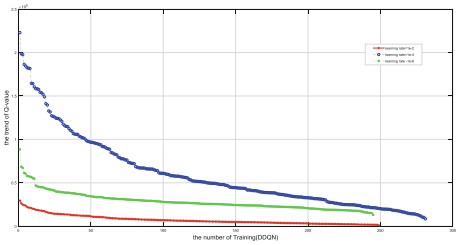

Fig. 5. Under different learning rate, the trend of the Q-value.

In Fig. 5, the learning rates we chosen are 1e–2, 1e–4 and 1e–6, the performance for the trend of Q-value is easily gotten that the good convergence. Comparing to the three learning rates, when equaling with 1e–2, the performance of convergence is best. We confirmed validation by comparing with Random Selection Scheme. the Fig. 6 is showed as follows.

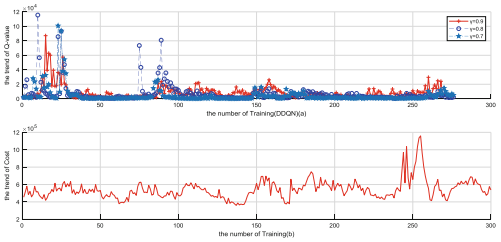

Fig. 6. Under different scheme, different scheme shows different validation and confirms Double DQN's effectiveness in this paper.

In Fig. 6, the DDQN algorithm is compared with Random Selection Algorithm (RSA), the two algorithm respectively corresponds Fig. 6(a) and (b). The x-axis denotes the number of training, and y-axis denotes the different cost between the two algorithms. From above results, Firstly, we add the number of the base stations with the edge servers, the average computing consumption will decrease. Secondly, from the performance, we can get that the speed of the DDQN's convergence is obviously better than RSA. Finally, the energy consumption of DDQN is one magnitude lower than RSA, which based on the energy analysis.

7 Conclusions

In this work, we consider the mobile computing for edge servers and the delay of users in a scenario in which some base stations with edge servers and others without edge servers. By jointly scheduling tasks to the base stations with edge servers, we optimize a MEC system so that the cost of link and computing resource. By solving the problems, we can get several conclusions. Firstly, in the MEC system, we should consider the users who are strict with the bounded delay. Secondly, we should choose the near neighbors to communicate and transfer what a user need. Thirdly, when the time of communication is larger, the edge servers should first find data from other users. Finally, for a certain MEC system, the number of the tasks scheduling is related to the number of the edge servers, but not absolutely. If we consider the number of the edge servers, we believe that the result will be better.

Acknowledgment. This work is supported by the National Key R&D Program of China (No. 2018YFC0807101).

This work is supported by the National Natural Science Foundation of China (No. 61374189), the joint fund of the Ministry of Education of China and China Mobile (MCM 20160304).

This work is supported by fundamental research funds for the central universities, China, under Grant No. 2672018ZYGX2018J001.

This work was supported by the Science & Technology Department of Sichuan Province under Grant 2018GZ0092.

References

1. Reiter, A., Zefferer, T.: Hybrid mobile edge computing: unleashing the full potential of edge computing in mobile device use cases. In: IEEE/ACM International Symposium on Cluster. IEEE (2017)
2. Wang, X., Liu, Z.: An energy-aware VMs placement algorithm in cloud computing environment. In: Second International Conference on Intelligent System Design & Engineering Application (2012)
3. Hung, C.H., Hsieh, Y.C., Wang, L.C.: Control plane latency reduction for service chaining in mobile edge computing system. In: International Conference on Network & Service Management. IEEE Computer Society (2017)
4. Abbas, N., Zhang, Y., Taherkordi, A., et al.: Mobile edge computing: a survey. IEEE Internet Things J. **99,** 1 (2017)
5. Bellavista, P., Chessa, S., Foschini, L., et al.: Human-enabled edge computing: exploiting the crowd as a dynamic extension of mobile edge computing. IEEE Commun. Mag. **56**(1), 145–155 (2018)
6. Zhao, T., Zhou, S., Guo, X., et al.: Tasks scheduling and resource allocation in heterogeneous cloud for delay-bounded mobile edge computing. In: IEEE International Conference on Communications. IEEE (2017)
7. Mao, Y., Zhang, J., Song, S.H, et al.: Power-delay tradeoff in multi-user mobile-edge computing systems (2016)
8. Zhang, K., Mao, Y., Leng, S., et al.: Optimal delay constrained offloading for vehicular edge computing networks. In: ICC 2017 - 2017 IEEE International Conference on Communications. IEEE (2017)

9. Xu, J., Ren, S.: Online learning for offloading and autoscaling in renewable-powered mobile edge computing. In: 2016 IEEE Global Communications Conference (GLOBECOM). IEEE (2017)
10. Wang, Y., Min, S., Wang, X., et al.: Mobile-edge computing: partial computation offloading using dynamic voltage scaling. IEEE Trans. Commun. **64**(10), 4268–4282 (2016)
11. Yu, Y., Zhang, J., Letaief, K.B.: 2016 IEEE Global Communications Conference (GLOBECOM) - Joint Subcarrier and CPU Time Allocation for Mobile Edge Computing, Washington, DC, USA, pp. 1–6, 4–8 December 2016
12. Wang, C., Liang, C., Yu, F.R., et al.: Computation offloading and resource allocation in wireless cellular networks with mobile edge computing. IEEE Trans. Wirel. Commun. **16**(8), 4924–4938 (2017)
13. Wang, C., Yu, F.R., Chen, Q., et al.: Joint computation and radio resource management for cellular networks with mobile edge computing. In: IEEE International Conference on Communications. IEEE (2017)
14. Ti, N.T., Le, L.B., Ti, N.T., et al.: Computation offloading leveraging computing resources from edge cloud and mobile peers. In: IEEE International Conference on Communications. IEEE (2017)
15. Corcoran, P.: Mobile-edge computing and internet of things for consumers: part II: energy efficiency, connectivity, and economic development. IEEE Consum. Electron. Mag. **6**(1), 51–52 (2016)
16. Ma, L., Liu, X., Pei, Q., et al.: Privacy-preserving reputation management for edge computing enhanced mobile crowdsensing. IEEE Trans. Serv. Comput. **99**, 1 (1939)
17. Liai, W., Xudong, Z., Chengyi, S., et al.: Studying parameter of MEC used to multi-modal optimization by two-level MEC. In: International Conference on Artificial Intelligence & Computational Intelligence. IEEE Computer Society (2009)
18. Xiang, S., Ansari, N.: EdgeIoT: Mobile Edge Computing for the Internet of Things (2016)
19. Lee, S.Q., Kim, J.U.: Local breakout of mobile access network traffic by mobile edge computing. In: International Conference on Information & Communication Technology Convergence. IEEE (2016)