Distributed Unsupervised Learning-Based Task Offloading for Mobile Edge Computing Systems

Jianming Wei, Qiuming Liu(✉), Shumin Liu, Yiping Zeng, and Xin Xiong

Department of Software Engineering,
Jiangxi University of Science and Technology, Nanchang, China
jianmingweijmw@163.com, liuqiuming@jxust.edu.cn, liushumin_001@163.com,
yipingZzeng@163.com, xiongxinbear@163.com

Abstract. Mobile edge computing has emerged as a new paradigm to enhance computing capabilities by offloading complicated tasks to nearby cloud server. To conserve energy as well as maintain quality of service, algorithms with low time complexity for task offloading is required. In this paper, a multi-user with multiple tasks scenario is considered, taking full account of factors including data size, bandwidth, channel state information, we propose a distributed unsupervised learning-based offloading algorithm for task offloading, where distributed parallel networks are employed to guarantee the robustness of algorithm. Additionally, we exploit a memory pool to store input data and corresponding decisions as key-value pairs. Based on the experience mechanism, the proposed algorithm can omit the step of data calibration compared with supervised learning method. To further reduce the communication cost, we analyze four bandwidth allocation schemes. Results reveal that the channel state-based strategy cost 3% less than data size-based, mixed and user size-based. Besides, the proposed algorithm can save 14, 18% cost than local and edge schemes respectively. Numerous results show that the proposed algorithm can achieve near-optimal decisions timely as well as having high reliability.

Keywords: Mobile edge computing · Distributed unsupervised learning · Energy efficiency · Task offloading

1 Introduction

Since the number of wireless devices has increased dramatically, the wireless network is facing a number of challenges such as mass data, high real-time requirement, limited resources and energy consumption. Meanwhile, traditional cloud computing exposes a lot of problems, such as data block and delay [1]. To solve problems mentioned above, mobile edge computing (MEC) is emerged to alleviate the task computing at cloud server by migrating the task near to the servers [2]. The MEC can effectively solve problems caused by massive tasks offloading

H. Gao et al. (Eds.): ChinaCom 2020, LNICST 352, pp. 539–553, 2021.
https://doi.org/10.1007/978-3-030-67720-6_37

and limited local resources, especially in wireless timely systems such as vehicular communications, augmented reality or virtual reality, so as to maintain the quality of server. However, a good algorithm with low complexity as well as robustness for task offloading in MEC systems is required. Meanwhile, the performance of algorithm for MEC network is influenced by many factors, such as bandwidth, data size, energy harvesting, task priority, etc.

There are mang studies on algorithms for MEC systems. In [3], an algorithm based on distributed deep learning was proposed, it can generate near-optimal offloading solution within short time. In [4], Madej et al. analyzed the resources allocation problem of task scheduling in MEC network, and demonstrated that the performance of task offloading strategy based on priority is better than the first come first serve. In [5], the author considered a vehicle network, where the channel state information feedback is provided with delay, and proposed a low-complexity algorithm for spectrum and power allocation. In [6], the authors considered the case of channel with interference, and obtained that the task offloading strategy is NP hard. Based on game theory, a distributed optimal algorithm was proposed to reduce the delay. Later, the scenario of energy harvesting devices is considered in [7], the authors not only studied the situation of task offloading, but also analyzed the task dropping strategy. To guarantee the quality of experience, a dynamic task offloading algorithm based on Lyapunov was proposed. In [8], by exploiting task buffer, a Lyapunov algorithm was proposed to optimize resource allocation and improve system performance. In [9], the problem of non-convex optimization strategy was simplified to a semi-definite relaxation problem and the sum of energy consumption of multiple users is quantified. In [10], a joint optimization method based on alternating direction multiplier method was proposed, which can obtain the optimal solution through a few iterations. Literature [11] modeled a user with multiple tasks communication scenario, and proposed an optimal strategy by combining semi-definite relaxation and heuristic random mapping method, which can make near-optimal decisions with a few random iterations. In [12], the authors proposed a deep Q-learning algorithm for multi-user with multiple tasks communication, which can generate near-optimal offloading decisions by training extensive samples.

From above studies, we found that the MEC network performance was impacted by all networks parameters, where the works mentioned above only consider partial parameters. To better analyze the role of multiple factors comprehensively, we model a multi-user with multiple tasks scenario, considering data size, bandwidth, and channel state information factors. Taking advantages of deep learning in MEC [13], we propose a distributed unsupervised learning-based offloading (DULO) algorithm. To fix the architecture of network, we set 3 users with 5 tasks as the value of input layers, the number of which is 18. Thus, the number of output layer is 15. Then, we exploit a memory pool to store input data and corresponding decisions as key-value pairs. Particularly, we design parallel networks to ensure the robustness of algorithm. To better evaluate the DULO algorithm, we use 80% of the samples as training samples. With thousands of iterations, parameters of networks are fitted based on

experience mechanism. Additionally, we take the rest of samples as testing samples to test the performance of a trained model. Experiments show that bandwidth allocation based on channel state can achieve 3% cost less than other three strategies. Besides, the proposed algorithm can save 14, 18% cost than local and edge schemes respectively, which reveals that DULO can make task offloading decisions with near-optimal results.

The rest of this paper is organized as follows. In Sect. 2, we introduce the network model and formulate problems. In Sect. 3, we propose a distributed unsupervised learning-based task offloading algorithm. Section 4 presents numerical results and evaluates the proposed algorithm.

2 System Model

2.1 Network Model

The one-to-many network cell for edge server and wireless devices (WDs) is shown in Fig. 1. Multiple users offload tasks to the edge server through the access point (AP). After the edge server completes the execution of a task, the edge server transmits results back to the corresponding WD. Since the AP and

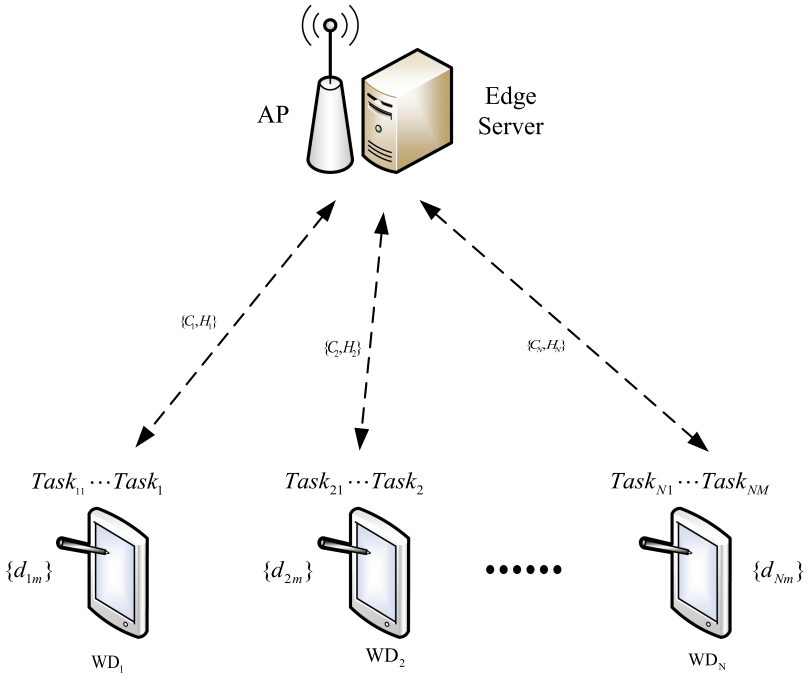

Fig. 1. Network model.

the server communicate with each other by using high speed optical fiber, the communication delay between the AP and the server can be ignored. Under the constraint bandwidth C of the system, C_n and H_n represent the bandwidth and channel gain of user n respectively. We denote $\mathbb{N} = \{1, 2, 3, \cdots N\}$ as a set of communication users, where each WD owns $\mathbb{M} = \{1, 2, 3, \cdots M\}$ tasks, which are mutually independent. We use $Task_{nm}$ to denote m-th task of n-th WD, and denote d_{nm} as data size. Since offloading decision is a binary decision set, we use $x_{nm} = 1$ to represent offloading task to the edge server, otherwise, when $x_{nm} = 0$ represents task being executed locally. Aiming to conserve energy and delay, two sub-models of energy consumption and delay are modeled to simplify the problem.

2.2 Energy Consumption Model

When $Task_{nm}$ is executed locally, we denote E_b as energy consumption for per bit of data processed by local resources. Therefore, the energy consumption E^l_{nm} of $Task_{nm}$ executed locally is given by

$$E^l_{nm} = E_b \times d_{nm}. \tag{1}$$

When task is offloaded to the server, we denote the energy consumption E^t_{nm} for $Task_{nm}$ transmission as

$$E^t_{nm} = P_n \times \frac{d_{nm}}{R_n}. \tag{2}$$

Since the task offloading process is affected by the channel state information, the transmission rate R_n is given by

$$R_n = C_n \times \log(1 + \frac{S_n}{\delta_n}), \tag{3}$$

where $S_n = P_n \times H_n$ and δ_n represents channel noise power of user n. We denote P_n as the transmitting power provided by the n-th WD, and denote S_n as signal power of user n. When the task is completed, results are transmitted to the mobile device. Because the size of results is much smaller than the size of task, the energy consumption for transmitting back is negligible. Therefore, the total energy cost E^r_{nm} for $Task_{nm}$ offloading is given by

$$E^r_{nm} = E^t_{nm} + E_s \times d_{nm}. \tag{4}$$

We use E_s to denote as energy consumption for per bit of data processed by server. When $E_s = 0$, we ignore the energy consumption for sever execution. Similarly, the delay model can also be divided into local execution and cloud execution.

2.3 Delay Model

We denote T_{nm}^{l} as delay when $Task_{nm}$ is executed locally. We use T_b to denote time cost for per bit of data. Thus, the delay of $Task_{nm}$ for local execution is given by

$$T_{nm}^{l} = T_b \times d_{nm}. \tag{5}$$

When $Task_{nm}$ is executed by server, time cost T_{nm}^{r} including transmission delay and execution delay. We denote transmission delay and execution delay as

$$T_{nm}^{t} = \frac{d_{nm}}{R_n}, \tag{6}$$

and

$$T_{nm}^{c} = \frac{d_{nm}}{F_c}, \tag{7}$$

respectively, where we denote F_c as the size of data that server can process in per second. Thus, the total delay when n-th WD decides to offload $Task_{nm}$ is given by

$$T_{nm}^{r} = T_{nm}^{t} + T_{nm}^{c}. \tag{8}$$

2.4 Problem Formulation

Through analyzing above two sub-models, the total cost for $Task_{nm}$ is given by

$$Cost_{nm} = (E_{nm}^{l} + T_{nm}^{l})(1 - x_{nm}) + (E_{nm}^{r} + T_{nm}^{r})x_{nm}, \tag{9}$$

where $x_{nm} = \{0, 1\}$, $\forall n, m$. To jointly minimize energy consumption and delay in MEC communication system, we formulate the problem as

$$\begin{aligned} &\min \boldsymbol{J}(\mathbf{d}, \mathbf{h}, \mathbf{c}, \mathbf{x}) = \sum_{n=1}^{N} \Big(\sum_{m=1}^{M} (E_{nm}^{l}(1 - x_{nm}) \\ &+ E_{nm}^{r} x_{nm}) + max(\sum_{m=1}^{M} T_{nm}^{r}, \sum_{m=1}^{M} T_{nm}^{l}) \Big) \\ &s.t. \quad \begin{cases} \sum_{n=1}^{N} C_n \leq C & \forall n \in \mathbb{N}, \\ C_n \geq 0 & \forall n \in \mathbb{N}, \\ x_{nm} = \{0, 1\} & \forall n \in \mathbb{N}, m \in \mathbb{M}. \end{cases} \end{aligned} \tag{10}$$

To solve the problem $\boldsymbol{J}$, we have to consider variables including decisions, data size, bandwidth and channel states. The complexity of $\boldsymbol{J}$ mainly depends on the size of N and M, but an algorithm of low time complexity is required. Next, we introduce an algorithm based on unsupervised deep learning (Table 1).

Table 1. Notations

Notation	Definition
$Task_{nm}$	m-th task of n-th user
x_{nm}	The decision of $Task_{nm}$
d_{nm}	Data size of $Task_{nm}$
E_b	Local energy consumption per bit
E^l_{nm}	Energy cost of $Task_{nm}$ for local execution
E_s	Server energy consumption for per bit
E^t_{nm}	Energy cost of $Task_{nm}$ for transmission
E^r_{nm}	Total energy cost of $Task_{nm}$ for edge processing
C_n	Bandwidth allocated to user n
P_n	Transmission power of n-th user
H_n	Channel gain of n-th user
F_c	Edge processing rate
R_n	Transmission rate of n-th user
S_n	Signal power of user n
δ_n	Channel noise power of user n
T_b	Local time consumption per bit
T^l_{nm}	Time cost of $Task_{nm}$ for local execution
T^t_{nm}	Time cost of $Task_{nm}$ for transmission
T^c_{nm}	Time cost of $Task_{nm}$ for server executing
T^r_{nm}	Total time cost of $Task_{nm}$ for edge processing

3 DULO Algorithm

In this section, we employ bandwidth allocation scheme to simplify the complexity of the problem (10) and proposed the DULO algorithm for task offloading, where distributed unsupervised learning are used to generate near-optimal decisions.

Once bandwidth is allocated, the problem (10) will be simplified, which affected by variables including decisions, data size and channel states. There are many proposed schemes on how to efficiently solve the bandwidth allocation problem, in [14] bandwidth allocation is studied. In this paper, we employ data-size-based and channel-state-based allocation schemes respectively, which is given by

$$C_n = \beta \times \frac{d_{nm}}{\sum_{n=1}^{N}\sum_{m=1}^{M} d_{nm}} + \gamma \times \frac{H_n}{\sum_{n=1}^{N} H_n}. \tag{11}$$

Then, the problem (10) is simplified as

$$\begin{aligned} \min \boldsymbol{J}^*(\mathbf{d},\mathbf{h},\mathbf{x}) = & \sum_{n=1}^{N}\Big(\sum_{m=1}^{M}(E_{nm}^{l}(1-x_{nm}) \\ & + E_{nm}^{r}x_{nm}) + max(\sum_{m=1}^{M}T_{nm}^{r}, \sum_{m=1}^{M}T_{nm}^{l})\Big) \\ s.t. \quad & \begin{cases} \sum\limits_{n=1}^{N} C_n \le C & \forall n \in \mathbb{N}, \\ C_n \ge 0 & \forall n \in \mathbb{N}, \\ x_{nm} = \{0,1\} & \forall n \in \mathbb{N}, m \in \mathbb{M}. \end{cases} \end{aligned} \tag{12}$$

Since the system has 2^{NM} choices to execute tasks, especially, complexity of many traditional algorithms grows exponentially as users or tasks increase, it is NP hard to get the optimal decision. In order to find the optimal solution in the problem $\boldsymbol{J}^*$, we try to find a function $f(u,v)$, which satisfies the following condition

$$f(d_{ij},h_{ij}) = x_{ij}, (i \in \mathbb{N}, j \in \mathbb{M}). \tag{13}$$

Deep learning network can solve complicated problems well, therefore, we propose an algorithm based on deep learning. Supervised deep leaning can fit a complex function well, especially, it can learn to find high dimensional features such as image classification. However, it is hard to get massive samples with labels. Therefore, unsupervised learning algorithm is used, as show in Fig. 2.

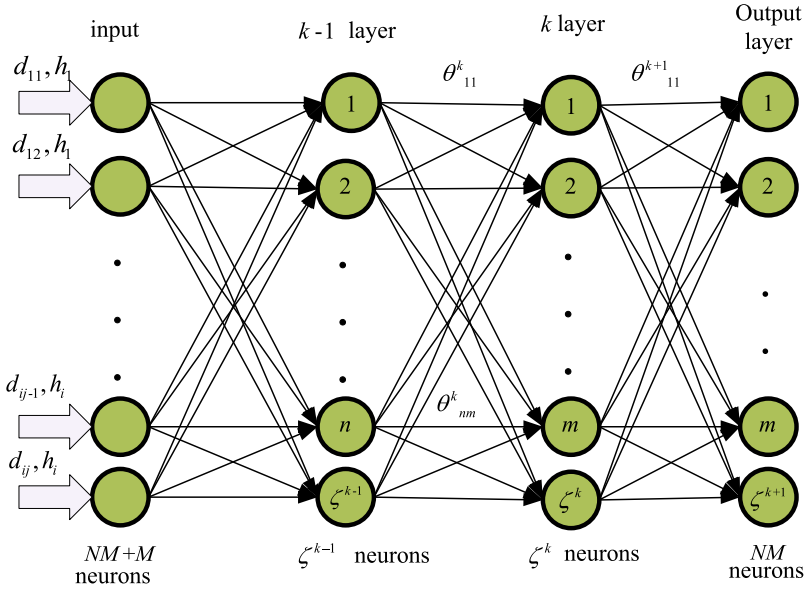

Fig. 2. Unsupervised learning algorithm for offloading network.

We denote data size d_{ij} and channel state information h_i as input values. We use x_{ij} to denote the output value corresponding to the binary offloading decision, if $x_{ij} = 1$, it denotes offloading the task to the server, else, it represents executing the task locally. Concretely, we employ ζ^k neurons as k-th layer and ζ^{k-1} neurons as $k-1$-th layer, which are fully connected. Then the output of m-th neuron of k-th layer is formulated as

$$z_m^k = \sum_{n=1}^{\zeta^{k-1}} (\theta_{nm}^k \times z_n^{k-1} + b_m^k), \tag{14}$$

where we denote θ_{nm}^k as value of weight of k-th layer, which connects between n-th neuron of $k-1$-th layer and m-th neuron of k-th layer. To formulate the nonlinear relationship between input and output, we employ relu function to process the output values of hidden layers, which is given by

$$y_m^k = \max(0, z_m^k). \tag{15}$$

Different from relu function, we use the following function to generate binary values of the output

$$x_{ij} = \begin{cases} 1, & y_m^k > 0; \\ 0, & y_m^k < 0. \end{cases} \tag{16}$$

Since it is a network with binary output, we use the cross-entropy function to optimize parameters by thousands of iterations, as

$$\boldsymbol{L}(\boldsymbol{\theta}) = -\mathbf{x}^{\mathrm{T}} \log(f(\mathbf{d}, \mathbf{h})) - (\mathbf{1} - \mathbf{x})^{\mathrm{T}} \log(1 - f(\mathbf{d}, \mathbf{h})). \tag{17}$$

To better fit the problem $\boldsymbol{J}^*$ and get the optimal solution, parameters about the network need to be fitted by backpropagation. The θ_{nm}^k at time t+1 is given by

$$\theta_{nm(t+1)}^k = \theta_{nm(t)}^k - \eta \frac{\partial L}{\partial \theta_{nm(t)}^k}, \tag{18}$$

where η denotes the learning rate.

However, it is not a good substitute for $\boldsymbol{J}^*$ with only one network, and the cost for training samples is time-consuming. Therefore, we employ I networks with same architecture but different parameters to ensure the robustness of the algorithm. Details about distributed unsupervised deep learning algorithm is shown as in Fig. 3.

At the beginning of training model, the parameter values $\boldsymbol{\theta}$ are randomly initialized. Then we have to create a empty memory pool and set the number of iterations. As the input values are operated with formulations (14) and (15) repeatedly, and processed by formulation (16). We get a set of binary offloading decisions. For I pairs offloading decisions, we compute the corresponding cost as candidates. From the number of I candidates, we select the minimum as a best candidate temporarily, as

$$\mathbf{x}_{tmp} = \arg\min_{\tau \in I} J^*(\mathbf{d}, \mathbf{h}, \mathbf{x}_\tau). \tag{19}$$

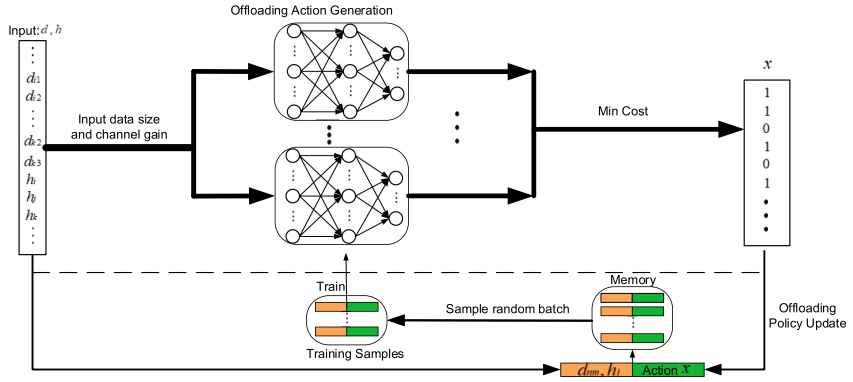

Fig. 3. Architecture of distributed unsupervised learning-based offloading network.

Then, we exploit the memory pool to store $Task_{nm}$ and corresponding decisions $\mathbf{x}_{tmp}$ as key-value pair. Once the temporary offloading decisions $\mathbf{x}_{tmp}$ is obtained, we use it as a temporary label for the task. Along with the iteration, we store better offloading decisions as well as corresponding tasks and discard the old key-value pair in the finite size of memory pool. Based on the experience mechanism, the model can learn to fit optimal parameters after training the network.

As a number of decisions is given, the optimal problem orientated to resources allocation is simplified to a convex problem, the server can better allocate its resources to execute tasks. Algorithm 1 presents the proposed DULO algorithm for task offloading.

Algorithm 1. DULO algorithm for training samples

Require:
The set of training samples for current batch $\boldsymbol{d}$, $\boldsymbol{h}$;
Ensure:
Offloading decision $\boldsymbol{x}$;
Initialization:
  Initialize I networks with random parameters;
  Create an empty memory pool;
  Set the number of iterations K;
while $j = 1, 2, ..., K$ **do**
  Randomly select values of tasks to load the network;
  Select the minimum value as a good candidate;
  Store tasks and decisions as key-value pairs;
  Backpropagate to fit parameters;
  Update the memory pool;
end while

4 Performance Evaluation

4.1 Performance of Training Model

In this paper, we simulate a small cellular network communication process. To fix the architecture of network, we set 3 users with 5 tasks as input layers, and we split 80% of samples as training samples and the remain as testing samples.

We train the model with different number of networks, taking the optimal solution as the benchmark. As shown in Fig. 4, when we use 2 networks to train model, it finally converges at the gain ratio 0.96 after 28 thousands. When more than 2 networks are used, the performance becomes better. We have to notice that when the number of networks is more than 3, it costs 15 thousands iterations to converge. As the number of networks increases, more data with correct labels can generate. Although, when more than 3 networks are used, the speed of convergence can't get faster, the performance get better when more than 3 networks are used. Generally, when more than 3 networks are used, the DULO performances better in convergence and gain ratio.

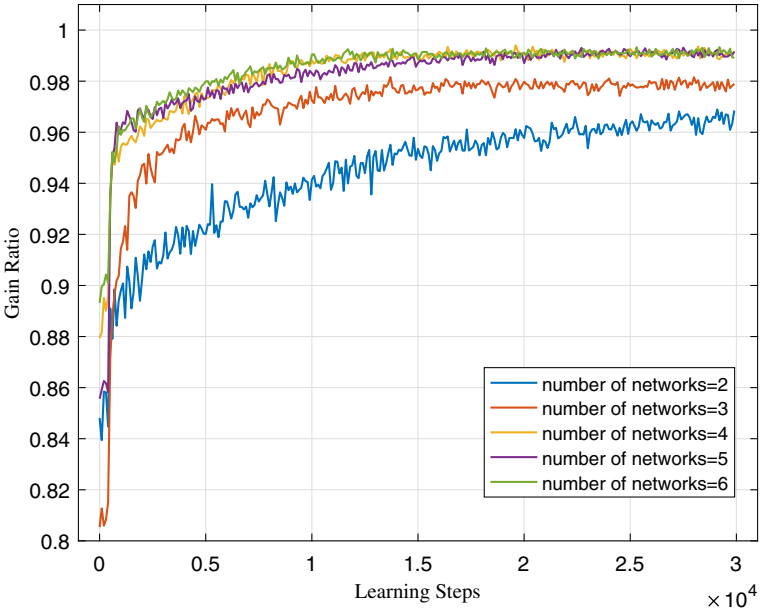

Fig. 4. Performance of training model with different number of networks.

Additionally, we test the performance of the algorithm with different channel states, including large-scale fading, rayleigh fading, rice fading as well as ideal communication system. We can learn from Fig. 5, the performance of training model maintains good with different channel states, it always converges at 0.98.

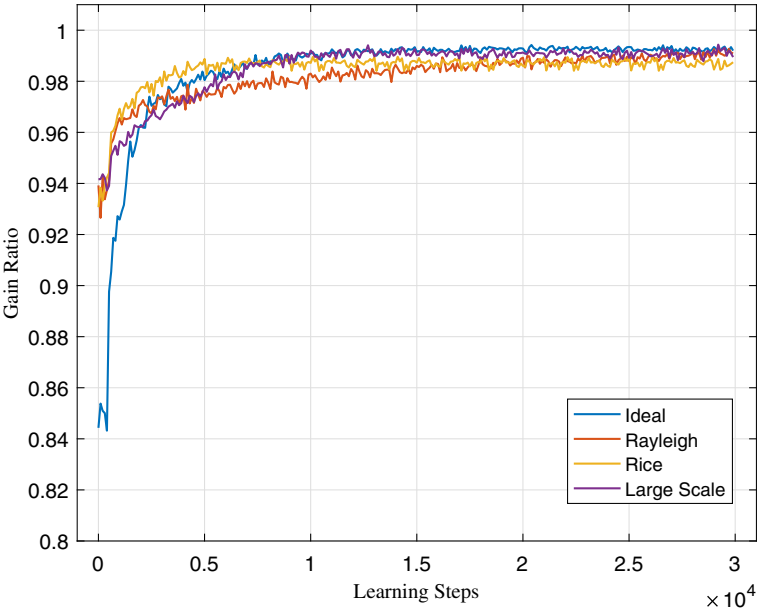

Fig. 5. Performance of training model with different channel states.

But the ideal communication converges faster than other situations. As shown in Fig. 5, when we train the ideal communication, the gain ratio converges after 10 thousands iterations. However, since one more factor considered, the DULO algorithm converges at 0.98 after 20 thousands iterations.

4.2 Performance of Testing Model

In order to evaluate the DULO algorithm better, we use testing samples as input values to analyze the offloading decisions, delay and robustness respectively.

Firstly, we analyze the influence of different weight of β. As shown in Fig. 6, the bandwidth allocation based on data-size strategy cost more than other strategies. When channel-state based strategy is employed, the communication cost 3% less than others. As the value of β increases, the cost goes up. When the β is 0.7, the cost tends to be stable. Generally, the channel state-based strategy costs less, therefore, the channel state factor is valuable.

Then, we compare DULO algorithm with other schemes in Fig. 7. We use enumeration strategy to enumerate all options, and then select the decision corresponding to minimum cost as the benchmark decision. As E_s is the energy consumption for per bit of data processed by server, local cost will not change with its value, edge cost goes up linearly with the value. Results show that the DULO cost less 14, 18% than local and edge schemes respectively, which is the near-optimal solution compared with minimum cost.

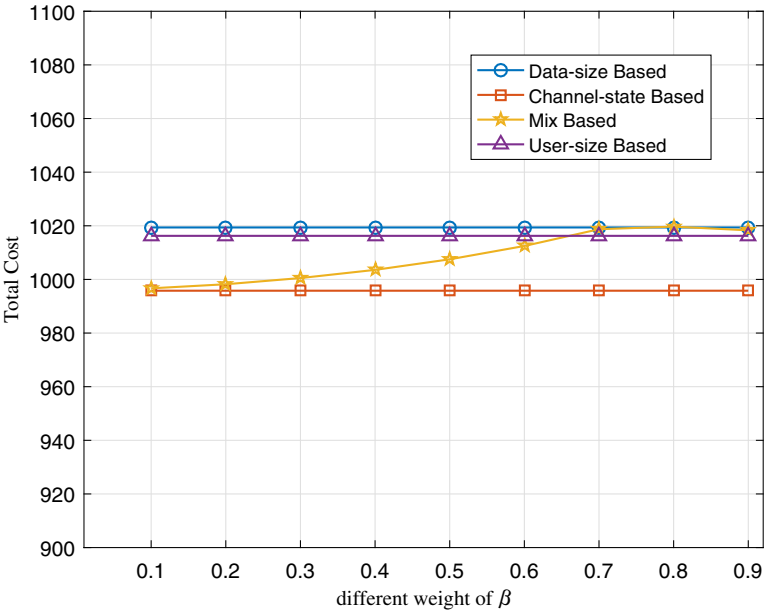

Fig. 6. Total cost under different bandwidth allocation schemes.

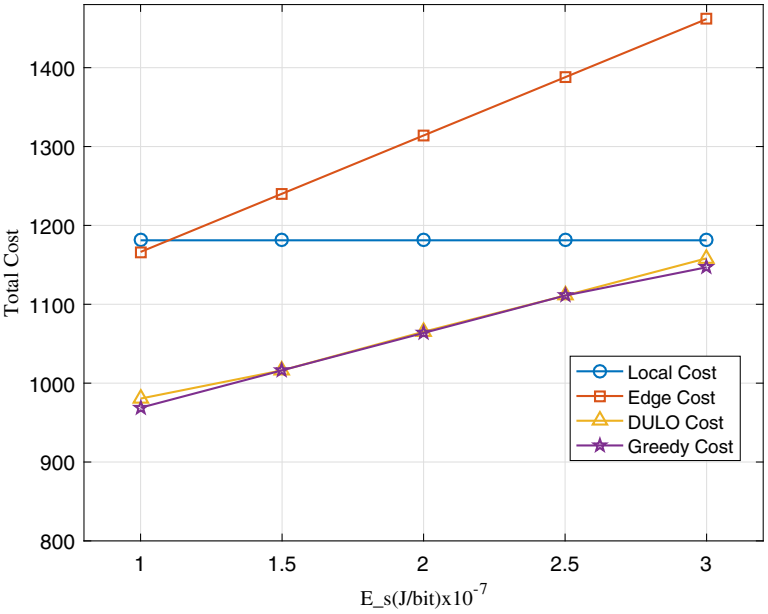

Fig. 7. Total cost with different schemes.

To test whether the algorithm can meet the real-time requirement of the system, time cost with different number of networks is tested. We can learn from Fig. 8, when channel state information is considered, it costs 0.02 s more than the ideal communication, since there are more data to compute. Additionally, time cost goes up with the number of networks. When the number of networks is less than 6, DULO algorithm generates near-optimal solution in 0.2 s. Specially, when the number of networks is 2, we can get a solution within 0.1 s. However, when 2 networks are used in the network, it performances worse than 3 networks. Both in terms of time cost and accuracy, a proper number of networks is required, therefore, more than 2 networks are used in this paper.

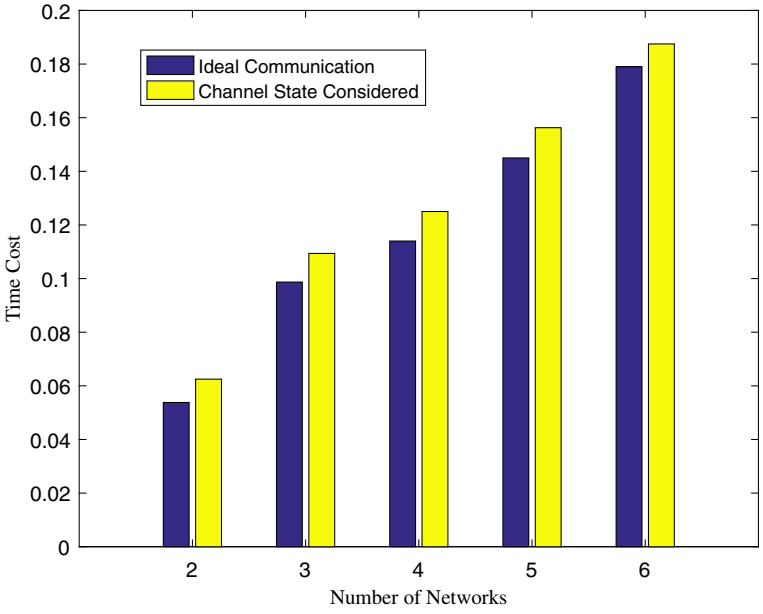

Fig. 8. Time cost for offloading decision with different number of networks.

Besides, we consider different hyper-parameters including number of networks, learning rates and memory sizes to test the robustness of DULO algorithm. As shown in Fig. 9, gain ratio goes up as the number of networks increases, when more than 5 networks are used, the performance of DULO is better than others. As memory pool gets larger, the DULO algorithm performances better, since there are more data with labels to learn for the DULO, which makes the algorithm robust. And when the value of learning rate is 0.03 and memory size is 2048, it has good stability with good performance whatever the number of network is. Taking different hyper-parameters into consideration, we can learn from Fig. 9 that the trained model can generates a near-optimal solution over the gain ratio 0.95.

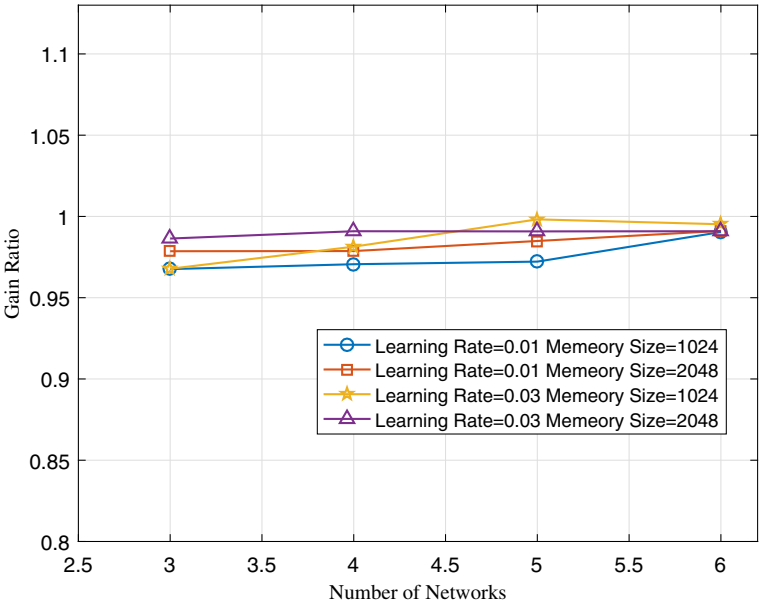

Fig. 9. Performance of testing model with different hyper-parameters.

5 Conclusion

In this paper, to improve quality of service by minimizing energy consumption and delay, we propose a DULO algorithm with known data size and channel states information for MEC networks. By exploiting the advantages of deep learning and parallel architecture, DULO algorithm makes up for the disadvantages of traditional algorithms such as decision delay. Compared with supervised learning, it avoids the complex work of data calibration and has better robustness. Results show that the bandwidth allocation based on channel-state strategy can save 3% cost than other schemes, and the proposed DULO algorithm can achieve a near-optimal decision in a short time less than 0.2 s, which cost 14, 18 less than percent than local and edge strategies respectively.

Acknowledgment. This work was supported in part by National Natural Science Foundation of China (No. 61761021), Natural Science Foundation of Jiangxi Province (Grant No. 20181bab202018, 20202BAB212003), Special Fund for Postgraduate Innovation of Jiangxi Province(No. YC2020-S481), Projects of Humanities and Social Sciences of universities in Jiangxi (JC18224), Jiangxi art planning project (YG2018042) and the Doctoral Research Fund of Jiangxi University of Science and Technology.

References

1. Shamsi, J.A., Khojaye, M.A., Qasmi, M.A.: Data-intensive cloud computing: requirements, expectations, challenges, and solutions. J. Grid Comput. **11**(2), 281–310 (2013)
2. Syamkumar, M., Barford, P., Durairajan, R.: Deployment characteristics of "the edge" in mobile edge computing. In: ACM Special Interest Group on Data Communication, Budapest, pp. 43–49 (2018). https://doi.org/10.1145/3229556.3229557
3. Huang, L., Feng, X., Feng, A., Huang, Y., Qian, L.P.: Distributed deep learning-based offloading for mobile edge computing networks. Mobile Netw. Appl. **23**(6), 1–8 (2018). https://doi.org/10.1007/s11036-018-1177-x
4. Madej, A., Wang, N., Athanasopoulos, N.: Priority-based Fair Scheduling in Edge Computing. arXiv Distributed, Parallel, and Cluster Computing (2018)
5. Liang, L., Kim, J., Jha, S.C.: Spectrum and power allocation for vehicular communications with delayed CSI feedback. IEEE Wireless Commun. Lett. **6**(4), 458–461 (2017)
6. Chen, X., Jiao, L., Li, W.: Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE ACM Trans. Netw. **24**(5), 2795–2808 (2016)
7. Zhao, H., Du, W., Liu, W.: QoE aware and cell capacity enhanced computation offloading for multi-server mobile edge computing systems with energy harvesting devices. In: IEEE SmartWorld, Ubiquitous Intelligence and Computing, Guangdong, pp. 671–678 (2018). https://doi.org/10.29007/mq2s
8. Mao, Y., Zhang, J., Song, S.H.: Power-delay tradeoff in multi-user mobile-edge computing systems. In: IEEE Global Communications Conference, Washington, pp. 1–6 (2016). https://doi.org/10.1109/GLOCOM.2016.7842160
9. Chen, M., Liang, B., Dong, M.: Joint offloading decision and resource allocation for multi-user multi-task mobile cloud. In: International Conference on Communications, Kuala Lumpur, pp. 1–6. Springer, Heidelberg (2016). https://doi.org/10.1109/ICC.2016.7510999
10. Author, B.S.: Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. IEEE Trans. Wireless Commun. **17**(6), 4177–4190 (2018)
11. Chen, M., Liang, B., Dong, M.: A semidefinite relaxation approach to mobile cloud offloading with computing access point. In: International Workshop on Signal Processing Advances in Wireless Communications, Stockholm, pp. 186–190. IEEE (2015). https://doi.org/10.1109/SPAWC.2015.7227025
12. Huang, L., Feng, X., Qian, L., Wu, Y.: Deep reinforcement learning-based task offloading and resource allocation for mobile edge computing. In: Meng, L., Zhang, Y. (eds.) MLICOM 2018. LNICST, vol. 251, pp. 33–42. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00557-3_4
13. Li, H., Ota, K., Dong, M.: Learning IoT in edge: deep learning for the internet of things with edge computing. IEEE Network **32**(1), 96–101 (2018)
14. Huang, L., Feng, X., Zhang, C.: Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing. Digital Commun. Netw. **5**(1), 10–17 (2019)