Deep Reinforcement Learning Based Task Offloading in SDN-Enabled Industrial Internet of Things

Jiadai Wang[1], Yurui Cao[1], Jiajia Liu[2(✉)], and Yanning Zhang[2]

[1] School of Cyber Engineering, Xidian University, Xi'an, China
jdwang_xd@163.com
[2] School of Cybersecurity, Northwestern Polytechnical University, Xi'an, China
liujiajia@nwpu.edu.cn

Abstract. Recent advances in communication and sensor network technologies make Industrial Internet of Things (IIoT) a major driving force for future industry. Various devices in wide industry fields generate diverse computation tasks with their distinct service requirements. Note that the distribution of such tasks has essential intrinsic patterns and varies according to factors like region, season and time. Different from previous efforts to develop algorithms in specific scenarios for reducing task execution latency without considering the task generation patterns of IIoT, we propose a DRL-based Task Offloading algorithm (DRLTO) to learn such generation patterns and maximize the task completion rate. A SDN-enabled multi-layer heterogeneous computing framework is also introduced to efficiently assign tasks according to the obtained knowledges towards their features. Extensive experiments validate that our algorithm can not only significantly improve the average task completion rate, but also achieve near-optimal results in lots of IIoT scenarios.

Keywords: IIoT · Task offloading · Deep Reinforcement Learning

1 Introduction

With the development of communication technologies, sensor network technologies, Industrial Wireless Networks (IWNs) and artificial intelligence [1,2], Industrial Internet of Things (IIoT) has been considered as an important driving force of future industry, and can bring great opportunities for high-efficiency production, manpower saving and cost reduction. Various IIoT devices in wide fields such as manufacturing, logistics, retailing and energy sector generate diverse computation tasks with their own service requirements. The distribution of these tasks can vary significantly according to region, season and time. For example, as affected by rainfall, light and weather, monitoring and measuring equipments in agriculture and energy industry can produce seasonal requests. In manufacturing plant, the production process with corresponding task flows may have temporal

S. Han et al. (Eds.): AICON 2019, LNICST 287, pp. 425–437, 2019.
https://doi.org/10.1007/978-3-030-22971-9_37

procedures. The task distribution between residential areas and factories will also change regularly with the crowd mobility. Therefore, IIoT task generation has its own patterns that deserve enough attention.

In IIoT environment, it is difficult to process a large part of computing tasks locally due to their stringent delay requirements or high computation costs. Relying solely on cloud computing or edge computing appeared challenging to meet this situation. Cloud has rich computing resources and storage capacity, however, forwarding huge amounts of data to the remote center for processing may lead to serious network congestion and performance degradation [3]. Edge computing can significantly reduce delay by pushing abundant resources near IIoT devices, nevertheless, its computing power is weaker than cloud computing [4,5]. Therefore, the multi-layer heterogeneous computing framework is much more suitable by combining the advantages of different computing resources to meet IIoT task requirements.

There have been some pioneer research works toward the interplay of edge computing and cloud computing for IIoT. Fu *et al.* [6] designed a flexible and economical scheme to store the data in a secure and searchable manner by integrating the fog computing and cloud computing. Kaur *et al.* [2] proposed a multi-objective evolutionary algorithm using Tchebycheff decomposition for data flow scheduling in edge-cloud IIoT framework. Shi *et al.* [7] presented a real-coded genetic algorithm for task reallocation and retransmission, which aimed to reduce the service latency in cloud-fog integrated IIoT architecture. In addition, a generic architecture is developed in [8] for smart processing and aggregation in large-scale manufacturing control systems. Although these works offer precious insights into resource allocation and task scheduling, it is noticed that they failed to provide a dedicated approach to characterize the essential feature of task generation patterns. In view of this, we propose a Deep Reinforcement Learning (DRL) based algorithm to learn their patterns and perform effective task offloading in IIoT accordingly. The main contributions of this paper are summarized as follows.

- To the best of our knowledge, we are the first to focus on the task generation patterns in IIoT. In particular, we propose a DRL-based Task Offloading algorithm (DRLTO) to learn the task generation pattern, make the appropriate decisions based on the interaction with the environment, then carry on the effective task assignment [9].
- A SDN-enabled multi-layer heterogeneous computing framework is also presented to efficiently allocate tasks according to their characteristics. SDN is adopted to facilitate the logically centralized control of distributed edge network infrastructures and IIoT devices [10].
- Different from the previous works, which considered only on reducing the average execution delay of tasks and neglected the task completion rate, our DRLTO is able to maximize the average task completion rate under the SDN-enabled multi-layer computing framework. As corroborated by extensive experiments, the average task completion rate is distinctly improved by our DRLTO, which is even close to the optimal enumeration algorithm.

The remainder of this paper is organized as follows: Sect. 2 presents the system model of task offloading in SDN-enabled multi-layer heterogeneous computing framework. Then, in Sect. 3, we define the problem of Average Task Completion Rate Maximization (ATCRM) and propose DRLTO as our solution. Following that, Sect. 4 gives extensive performance evaluation. Finally, we conclude this paper in Sect. 5.

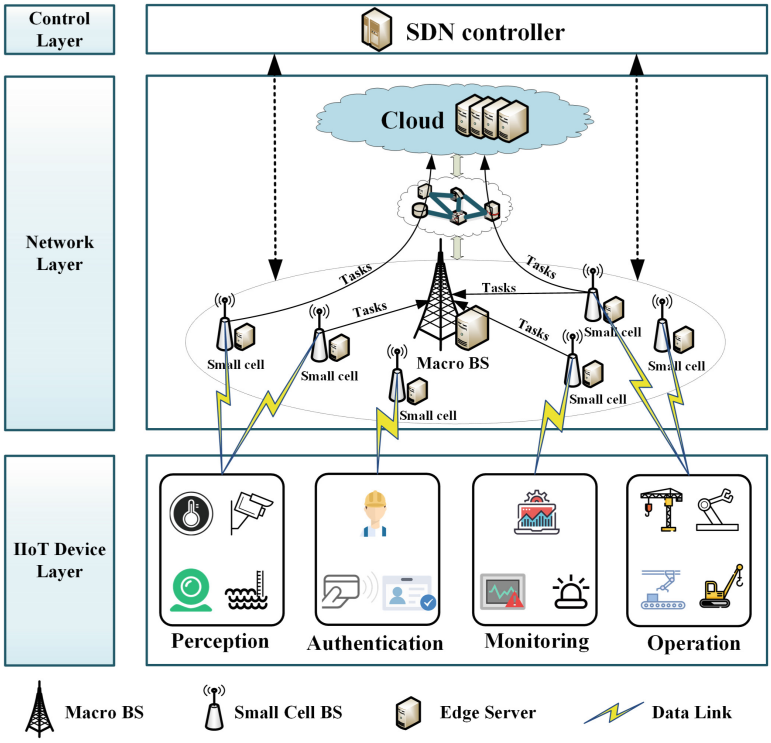

Fig. 1. The SDN-enabled heterogeneous multi-layer computing architecture.

2 System Model

As illustrated in Fig. 1, we consider a SDN-enabled multi-layer heterogeneous computing framework, in which the macro base station with the edge server supports task processing. Small cell base stations with edge servers are also deployed near IIoT devices for achieving high speed network access. We regard base stations with edge servers as edge nodes, and use MBS and SBS to represent the macro base station node and the small cell base station node. For the sake of simplicity, we consider one MBS in our model.

2.1 Task Model

IIoT devices generate various computation tasks for further processing on the edge servers or clouds. We assume that each type of computation task is atomic and cannot be partitioned, and one device only generates one type of tasks. Let $K = \{k_1, k_2, ..., k_{|K|}\}$ denotes the types of IIoT devices and $S = \{s_1, s_2, ..., s_{|S|}\}$ represents the SBSs. IIoT devices communicate with nearby SBSs via wireless data links and all tasks are first assigned on the SBSs. We denote the number of type-k task in L consecutive time slots as $num = \{n_k^1, n_k^2, ..., n_k^L\}$. In particular, these tasks can own diverse features. The task in jth time slot of device k, i.e., a_k^j, can be described in four items, i.e., $a_k^j = (I_k, O_k, C_k, T_k^{\max})$, where I_k is the input size of task a_k^j, O_k is the output size of a_k^j, C_k represents the required CPU cycles to complete the task a_k^j, and $T_k^{\max}$ is the maximum tolerable delay of a_k^j. In our considered architecture, SDN controller is adopted to collect the edge network state and control the task flow. What's more, $|S|$ DRL units are deployed in SDN controller, they can be periodically trained for the assigned tasks on SBSs. Thus the offloading strategy can be decided to complete different types of tasks, i.e., executing the task in the nearby SBS, offloading the task to the MBS which connected to the SBS in wired manner [11], or offloading the task to the cloud. Detailed algorithm will be introduced in Sect. 3. In the following, we present the execution model of the three different offloading strategies separately considering the uplink delay, downlink delay and computing delay.

2.2 Small Cell Execution Model

The processing and storage capacities of SBSs are limited. Thus SBSs are suitable for handling those computation tasks with low latency requirements and less computation complexity. When task a_k^j is computed in SBS, the total time consumption is composed of three parts, i.e., transmission time of the input data I_k from IIoT device to SBS, the computing delay in the SBS and the transmission delay of the output data O_k from the SBS to the device. In particular, the processing delay T_{ks}^j of task a_k^j in SBS s ($s \in S$) is given by

$$T_{ks}^j = \frac{I_k}{r_{ks}^{ul}} + \frac{C_k}{f_{ks}^j} + \frac{O_k}{r_{ks}^{dl}}. \tag{1}$$

Here, r_{ks}^{ul} and r_{ks}^{dl} are the uplink and downlink rate for task a_k^j via wireless transmission, respectively. f_{ks}^j is the processing capacity of SBS s allocated to the a_k^j. The uplink rate r_{ks}^{ul} can be computed by

$$r_{ks}^{ul} = \frac{B^{ul}}{|K|} \log_2 \left(1 + \frac{p_k^{ul} g_k}{N_0 B^{ul}/|K|}\right), \tag{2}$$

where B^{ul} is the wireless uplink channel bandwidth which is divided among $|K|$ devices equally, e.g., using OFDMA. p_k^{ul} represents the transmit power of device k, g_k denotes the channel power gains of device k, and N_0 is the noise power

spectral density at the receiver. Note that g_k can be calculated by $g_k = (d_k)^{-\alpha}$, where d_k is the distance between IIoT device k and the connected SBS, and α is the path loss factor. After the completion of task a_k^j, the SBS transmits the output data to IIoT device k, and the downlink data rate is given by

$$r_{ks}^{dl} = \frac{B^{dl}}{|K|}\log_2\left(1 + \frac{p_k^{dl} g_k}{N_0 B^{dl}/|K|}\right), \tag{3}$$

where B^{dl} is the wireless downlink bandwidth and p_k^{dl} is the transmit power of SBS allocated to IIoT device k.

2.3 Macro Cell Execution Model

Since the MBS has stronger computation ability than SBS, computation tasks that have lower delay requirements or need higher computation capacity can be assigned to the MBS from SBS. Let m denotes the MBS. The time consumption of processing task on MBS contains five parts, i.e., the transmission time from IIoT device to SBS via wireless access and SBS to MBS in wired manner, the execution time on MBS, and the transmission time of backhaul from MBS to SBS and SBS to IIoT device. Thus the time consumption T_{km}^j of task a_k^j allocated to MBS m can be calculated as follows

$$T_{km}^j = \frac{I_k}{r_{ks}^{ul}} + \frac{I_k}{B_{sm}^{ul}} + \frac{C_k}{f_{km}^j} + \frac{O_k}{B_{sm}^{dl}} + \frac{O_k}{r_{ks}^{dl}}, \tag{4}$$

where B_{sm}^{ul} and B_{sm}^{dl} are the uplink and downlink data rate between MBS and SBS, and f_{km}^j denotes the processing capacity of MBS m allocated to the task a_k^j.

2.4 Cloud Execution Model

In particular, those tasks that have loose delay requirements and need a large amount of calculation can be assigned to the cloud. Due to the powerful computing capacity of the cloud, the execution time on cloud can be neglected. Thus the processing time of the task a_k^j allocated to the cloud contains the transmission time from IIoT device to SBS, the transmission delay t_c^{ul} from SBS to cloud via core network and t_c^{dl} from cloud to SBS, as well as the time from SBS to IIoT device. The T_{kc}^j is given by

$$T_{kc}^j = \frac{I_k}{r_{ks}^{ul}} + t_c^{ul} + t_c^{dl} + \frac{O_k}{r_{ks}^{dl}}. \tag{5}$$

3 Problem Formulation and DRL-based Solution

3.1 Problem Formulation

As we illustrated above, all tasks are first transmitted to SBS and then they will be assigned corresponding offloading strategies according to their features and

the current task generation state. The execution time of task a_k^j is described as follows

$$T_k^j = \delta_k^j T_{ks}^j + \beta_k^j T_{km}^j + \gamma_k^j T_{kc}^j. \tag{6}$$

Here $\delta_k^j, \beta_k^j, \gamma_k^j \in \{0, 1\}$ are the offloading decisions of task a_k^j. $\delta_k^j = 1$ means that task a_k^j will be executed on the SBS, $\beta_k^j = 1$ means that the task a_k^j will be processed on the MBS, and $\gamma_k^j = 1$ means executed on the cloud. Obviously we have that $\delta_k^j + \beta_k^j + \gamma_k^j = 1$.

Average Task Completion Rate Maximization (ATCRM): According to our system model in Sect. 2, IIoT devices generate different types of computation tasks. For each task, it is either computed on the SBS which is close to the IIoT device, or offloaded to the MBS further, or offloaded to the cloud. Since each task has its maximum tolerable delay T_k^{max}, when the execution time T_k^j exceeds T_k^{max}, the task will be dropped. Given the $|K|$ devices and L consecutive time slots, our aim is to maximize the task completion rate within the tasks' tolerable delay. The problem can be formulated as follows

$$\max \sum_{j=1}^{L} \frac{\sum_{k=1}^{|K|} \varphi_k^j}{\sum_{k=1}^{|K|} n_k^j}, \tag{7}$$

where φ_k^j is the successful completion number of a_k^j in time slot j.

3.2 DRL-based Solution

In complex IIoT environment, it is difficult to adapt to the changeable task flow using traditional offloading schemes. Thus, ATCRM problem should be handled more intelligently with learning the task generation pattern. In this section, we propose a DRL based Task Offloading algorithm (DRLTO) to solve the ATCRM problem. As a branch of machine learning, DRL can acquire knowledge in the environment, improve policy to adjust to the changeable IIoT environment and make sequences of decisions to realize the effective task offloading [9]. Its goal is to obtain maximal cumulative rewards. We adopt the effective scheme Deep Q-Network (DQN) to realize our DRLTO, in which the task generation state is adopted as the input of the Deep Neural Network (DNN) and each possible action's Q value as the output. Three key elements of our DRLRO in DQN, i.e., state, action and reward, can be firstly defined as follows.

- **State:** In each time slot, the state vector of our proposed DRLRO can be denoted as $S = \{\{p_k^s\}, \{p_k^m\}, \{p_k^c\}, \forall k \in K\}$. Thereinto, p_k^s, p_k^m, p_k^c represent the number of k-type tasks that are assigned to SBS, MBS and cloud respectively.
- **Action:** The action vector can be given as $A = \{\{\alpha_k^s\}, \{\alpha_k^m\}, \{\alpha_k^c\}, \forall k \in K\}$, while α_k^s, α_k^m, α_k^c denote the action taken by the k-type tasks on SBS, MBS

and cloud, respectively. Reasonable action choices include staying in SBS, offloading q k-type tasks to MBS, and offloading q type-k tasks to cloud. Note that the action is only a part of the assignment process and is not actually performed until final offloading decision has been made.

– **Reward:** In each task assignment step, after taking the reasonable action A, the DRL agent can acquire a certain reward that needs to reflect the objective of our DRLTO, i.e., to maximize the average task completion rate. We define the reward that the DRL agent receives as $R = (\frac{\sum_{k=1}^{|K|} \varphi_k}{\sum_{k=1}^{|K|} n_k})^2$ if the task completion rate increases after doing a reasonable action. Otherwise, the agent will receive reward $R = -1$.

Algorithm 1. DRL based Task Offloading (DRLTO) Algorithm

Require: Discount rate γ, exploration rate ε, replay memory capacity C

Initialize replay memory D to capacity C
Initialize evaluation DNN with parameters θ
Initialize target DNN with parameters $\theta^- = \theta$
for each episode e **do**
  Initialize state S_1
  for each task assignment step t **do**
    Generate random number $\mu \in [0, 1]$
    if $\mu < \varepsilon$ **then**
      Randomly select an action A_t
    else
      Select $A_t = \arg\max_{A_t} Q(S_t, A_t; \theta)$, where Q is estimated by evaluation DNN
    end if
    Execute action A_t in emulator
    Observe reward R_t and new state S_{t+1}
    Store transition (S_t, A_t, R_t, S_{t+1}) in D
    Sample random mini-batch of transitions (S_j, A_j, R_j, S_{j+1}) from D
    if episode terminates at step$_{j+1}$ **then**
      Set $Y_j = R_j$
    else
      Set $Y_j = R_j + \gamma \max_{A_{j+1}} Q'(S_{j+1}, A_{j+1}; \theta^-)$, where Q' is estimated by target DNN
    end if
    Execute gradient descent using MSE function $(Y_j - Q(S_j, A_j; \theta))^2$
    Each Z steps reset $\theta^- = \theta$
  end for
end for

The training procedure of DRLTO is illustrated in Algorithm 1. In each episode, that is, the task assignment process, according to the task distribution information collected by the SDN controller, DRLTO can make decisions by the evaluation Q-values $Q(S, A; \theta)$ outputted by the evaluation DNN with neural network parameters θ. After choosing action using ε-greedy strategy, the certain reward R can be calculated as we defined above, and the next state S' can also be obtained accordingly. Each transition item (S_j, A_j, R_j, S_{j+1}) is stored in the experience replay memory and will be extracted randomly to execute the training procedure. The parameters θ^- of target DNN are updated on the basis of fixed-Q target strategy [9], namely, θ^- are updated according to θ at regular interval, while θ can be updated by Mean Square Error (MSE) loss function. Through the iteration of the above process, the task offloading decisions can be obtained.

The training process can be periodically executed for the varying task generation pattern. The SDN controller is responsible to send control messages using control plane function, e.g., Open Network Operating System (ONOS) and OpenDayLight (ODL) [12], to each SBS for operating the task flow.

4 Performance Evaluation

4.1 Experimental Settings

We considered a multi-layer heterogeneous computing scenario with the coexistence of centralized cloud, distributed SBS and MBS. A simulator in Python was developed to realize the DRLRO algorithm for smart task offloading in IIoT environment. Based on several in-lab testing applications described in [13], we set the average task input data size between 100 Kb and 500 Kb, and the output size between 5 Kb and 500 Kb. The task requirement of CPU cycles varied from 50 Megacycles to 500 Megacycles. The maximum tolerable delay of task was set between 200 ms and 1000 ms. The DNN structure in DRL was configured as three fully connected layers, and each hidden layer had 50 units. Table 1 presents detailed parameter settings in our experiment.

Three offloading strategies are presented to be our benchmark. (1) Random task offloading: Tasks are randomly offloaded to SBS, MBS or cloud. (2) Classified task offloading: Tasks are classified and offloaded to SBS, MBS and cloud according to their features. Here we use the required CPU cycles and the maximum tolerable delay of the task as classification criteria. Note that this classification is fixed and does not change with the task distribution. (3) Enumeration: Enumerate each possibility and find the optimal solution with high time complexity. All algorithms ran on a workstation with double Intel Xeon E5-2630 V4 2.2 GHz CPUs, 128 GB Random Access Memory (RAM), Nvidia Titan 12G GPU, and Ubuntu 14.04 64-bit operating system.

Table 1. Parameter settings in the simulation

Parameters	Value
Uplink and downlink bandwidth of the SBS	20 MHz
Uplink and downlink bandwidth of the MBS	40 MHz
Transmission rate between SBS and MBS	1 Gbps
Transmit power of IIoT device	100 mW
Distance between IIoT device and SBS	10–50 m
Distance between IIoT device and MBS	80–500 m
Path loss factor	4
Input data size of task	100–500 Kb
Output data size of task	5–100 Kb
Number of the required CPU cycles to complete the task	50–500 Megacycles
Maximum tolerable delay of task	200–1000 ms
Noise power spectral density	−147 dbm/Hz
Number of hidden layers	2
Number of neurons of each hidden layer	50
Replay memory size	2000
Mini-batch size	32
Learning rate	0.001
Target network update rate	500
Reward discount parameter	0.8
ϵ-greedy parameter	0.1

4.2 Numerical Results

From Fig. 2, one can easily observe that with the increase of total processing capacity, the combination of SBS and MBS is much more efficient for ATCRM problem than simple use of the SBS or MBS. This is because the two types of base stations give more options to various tasks. Each type of tasks can select different strategies for processing according to its own characteristics, and thus can also further decentralize the computing load.

For the combination of SBS and MBS, we have done more experiments to compare the proposed DRLTO with three benchmark algorithms, i.e., random task offloading, classified offloading and the enumeration algorithm. In Fig. 3, the experimental results show that with the increasing of total processing capacity, the four algorithms all growing, while our proposed DRLTO algorithm is much better than the random and classified offloading algorithms and is very close to optimal enumeration algorithm that has high complexity. This is because that with the increasing processing capacity, the average task completion rate will also raise as the processing delay of tasks become smaller. Besides, our DRLTO algorithm can adaptively learn the task generation pattern of the IIoT devices,

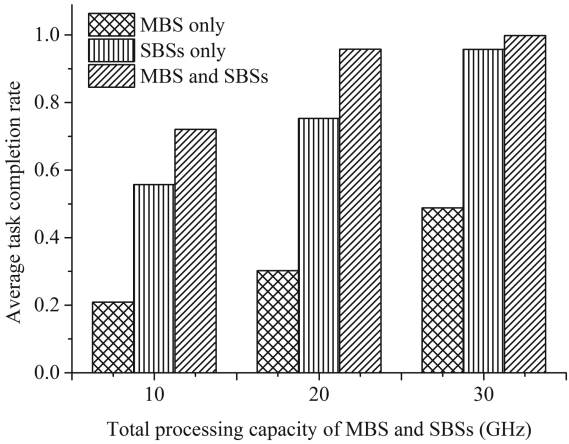

Fig. 2. Performance comparison on the simple use of the SBS or MBS and the combination of SBS and MBS under different total processing capacities.

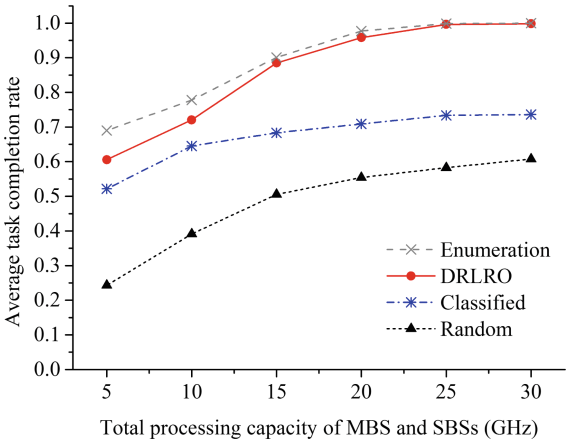

Fig. 3. Performance comparison of the benchmark algorithms and the proposed DRLTO under different total processing capacities.

make the appropriate decision based on the interaction with the environment, thus is more intelligent and better than the random and classified offloading algorithms.

Similar conclusion can also be found in Figs. 4 and 5. In Fig. 4, we set the capacities of MBS and SBS as 10 GHz and 1 GHz respectively. When the processing capacity of MBS is fixed, with the increasing number of SBSs, the DRLTO, random and enumeration algorithms all show growing trends, while the classified task offloading algorithm first tends to invariance, then grows again. This is due to that the classification of tasks is constant in classified algorithm. When the

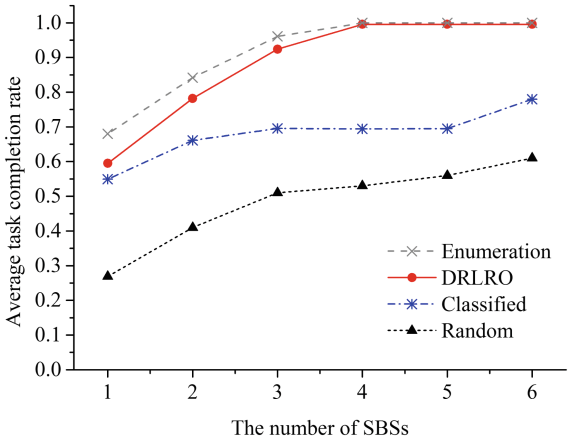

Fig. 4. Comparison of the benchmark algorithms and the proposed DRLTO under different number of SBSs.

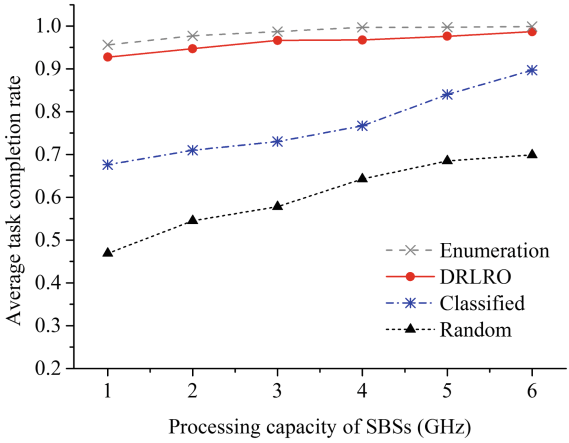

Fig. 5. Performance comparison of the benchmark algorithms and the proposed DRLTO under different processing capacities of SBSs.

number of SBSs is 4 or 5, the increase in processing capacity is not enough to affect the performance of this task offloading algorithm. When the number of deployed SBSs increased to 6, the total processing capacity of the base stations becomes larger, so the result of classified algorithm becomes better. We then set the capacity of MBS as 15 GHz with two SBSs in Fig. 5. The experimental results show that when the number of SBSs is fixed, with the increasing processing capacity of SBS, the four algorithms all growing, while our proposed DRLTO algorithm performs better than the random and classified task offloading algorithms and is close to optimal enumeration algorithm.

5 Conclusion

In this article, we have taken advantage of the IIoT task generation patterns to efficiently enhance the task offloading performance. Under the SDN-enabled multi-layer heterogeneous computing scenario, the aforementioned experimental results have shown that our proposed DRLTO is much more effective than random and classified algorithms, and its performance is very close to the optimal enumeration algorithm in maximizing average task completion rate. This is because our DRLTO can actively learn the task generation patterns through the training process and intelligently choose appropriate action for different types of tasks in complex IIoT environment. Therefore, it is obvious that DRL technology has great potential in smart task offloading and deserves further study. Future work is in progress to supplement more environment details into the state and action vector representation.

References

1. Li, X., Li, D., Wan, J., Liu, C., Imran, M.: Adaptive transmission optimization in SDN-based industrial internet of things with edge computing. IEEE Internet Things J. **5**(3), 1351–1360 (2018)
2. Kaur, K., Garg, S., Aujla, G.S., Kumar, N., Rodrigues, J.J., Guizani, M.: Edge computing in the industrial internet of things environment: software-defined-networks-based edge-cloud interplay. IEEE Commun. Mag. **56**(2), 44–51 (2018)
3. Nuratch, S.: The IIoT devices to cloud gateway design and implementation based on microcontroller for real-time monitoring and control in automation systems. In: 2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA), pp. 919–923 (2017)
4. Chekired, D.A., Khoukhi, L., Mouftah, H.T.: Industrial IoT data scheduling based on hierarchical fog computing: a key for enabling smart factory. IEEE Trans. Ind. Inform. **14**(10), 4590–4602 (2018)
5. Sun, W., Liu, J., Yue, Y., Zhang, H.: Double auction-based resource allocation for mobile edge computing in industrial internet of things. IEEE Trans. Ind. Inform. **14**(10), 4692–4701 (2018)
6. Fu, J., Liu, Y., Chao, H.-C., Bhargava, B., Zhang, Z.: Secure data storage and searching for industrial IoT by integrating fog computing and cloud computing. IEEE Trans. Ind. Inform. **14**(10), 4519–4528 (2018)
7. Shi, C., et al.: Ultra-low latency cloud-fog computing for industrial internet of things. In: 2018 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–6 (2018). https://doi.org/10.1109/WCNC.2018.8377192
8. Raileanu, S., Borangiu, T., Morariu, O., Iacob, I.: Edge computing in industrial IoT framework for cloud-based manufacturing control. In: 2018 22nd International Conference on System Theory, Control and Computing (ICSTCC), pp. 261–266. IEEE (2018)
9. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)
10. Qazi, Z.A., Lee, J., Jin, T., Bellala, G., Arndt, M., Noubir, G.: Application-awareness in SDN. In: ACM SIGCOMM Computer Communication Review, vol. 43, pp. 487–488. ACM (2013)

11. Wang, C., Liang, C., Yu, F.R., Chen, Q., Tang, L.: Computation offloading and resource allocation in wireless cellular networks with mobile edge computing. IEEE Trans. Wirel. Commun. **16**(8), 4924–4938 (2017)
12. Bhowmik, S., Tariq, M.A., Koldehofe, B., Durr, F., Kohler, T., Rothermel, K.: High performance publish/subscribe middleware in software-defined networks. IEEE/ACM Trans. Netw. **25**(3), 1501–1516 (2017)
13. Guo, H., Liu, J.: Collaborative computation offloading for multi-access edge computing over fiber-wireless networks. IEEE Trans. Veh. Technol. **67**(5), 4514–4526 (2018)