Physical Communication 43 (2020) 101184

Contents lists available at ScienceDirect

Physical Communication

journal homepage: www.elsevier.com/locate/phycom

Full length article

# Deep reinforcement learning based mobile edge computing for intelligent Internet of Things☆

Rui Zhao [a], Xinjie Wang [b], Junjuan Xia [a,*], Liseng Fan [a,*]

[a] School of Computer Science, Guangzhou University, Guangzhou 510006, China
[b] Qingdao University of Technology, Qingdao 266520, China

ARTICLE INFO

*Article history:*
Received 18 March 2020
Received in revised form 29 June 2020
Accepted 12 August 2020
Available online 20 August 2020

*Keywords:*
Deep reinforcement learning
Intelligent IoT
Mobile edge computing

ABSTRACT

In this paper, we investigate mobile edge computing (MEC) networks for intelligent internet of things (IoT), where multiple users have some computational tasks assisted by multiple computational access points (CAPs). By offloading some tasks to the CAPs, the system performance can be improved through reducing the latency and energy consumption, which are the two important metrics of interest in the MEC networks. We devise the system by proposing the offloading strategy intelligently through the deep reinforcement learning algorithm. In this algorithm, Deep Q-Network is used to automatically learn the offloading decision in order to optimize the system performance, and a neural network (NN) is trained to predict the offloading action, where the training data is generated from the environmental system. Moreover, we employ the bandwidth allocation in order to optimize the wireless spectrum for the links between the users and CAPs, where several bandwidth allocation schemes are proposed. In further, we use the CAP selection in order to choose one best CAP to assist the computational tasks from the users. Simulation results are finally presented to show the effectiveness of the proposed reinforcement learning offloading strategy. In particular, the system cost of latency and energy consumption can be reduced significantly by the proposed deep reinforcement learning based algorithm.



## 1. Introduction

In recent years, there has been a great progress in the development and application of wireless communication systems [1,2], and many new techniques have been proposed to speed up the data rate of wireless communication. Among these techniques, relaying technique is one of the most promising techniques to enhance the communication quality, and it can work in many protocols such as decode-and-forward (DF) and amplify-and-forward (AF). In addition, cognitive technique is also very attractive [3,4], since it can help utilize the spectrum resources very effectively [5,6]. Moreover, multiple antenna technique can help enhance the transmission data rate [7,8], and its newest form of massive multi-input multi-output (MIMO), which can help improve the transmission data rate by ten or hundred times [9].

With the development and allocation of wireless communication systems, especially about the fifth-generation (5G) networks, there has been a great progress in the development of internet of things (IoT), which can also support the development of smart cities. In IoT systems, the nodes can not only communicate with each other, but also have the ability to store the data and compute. Among the IoT systems, the technique of wireless caching is quite important, since it can help improve the user's experience quality substantially [10,11]. Fortunately, the storage cost has been decreasing very rapidly due to the development of storage technique. Besides the wireless caching technique, the technique of mobile edge computing (MEC) plays a very important role in the IoT-based systems [12–14], where the nodes can compute the tasks assisted by the near-by nodes instead of remote cloud. In this way, the latency and energy consumption can be reduced substantially.

Driven by the development of big data and deep learning, there has been a trend in the development of intelligent systems, such as the intelligent IoT. In [15,16], the deep convolutional neural networks (CNNs) were incorporated into the conventional

☆ This work was supported in part by the NSFC (No. 61871139), in part by the International Science and Technology Cooperation Projects of Guangdong Province (No. 2020A0505100060), by the Science and Technology Program of Guangzhou (No. 201807010103), in part by the research program of Guangzhou University (No. YK2020008), by the Project of Shandong Province Higher Educational Science and Technology Program (No. J18KA315) and the Shandong Provincial Natural Science Foundation of China (ZR2018MF002).

* Corresponding authors.
*E-mail addresses:* 2111806073@e.gzhu.edu.cn (R. Zhao), xinjie1023@163.com (X. Wang), xiajunjuan@gzhu.edu.cn (J. Xia), lsfan2019@126.com (L. Fan).

https://doi.org/10.1016/j.phycom.2020.101184
1874-4907/

detectors such as the maximum likelihood detector (MLD), zero-forcing (ZF), and minimum mean square error (MMSE) detectors, and it can be found that the detection performance can be improved significantly. In [17,18], shows a good application of machine learning in flight delay prediction and basic image analysis. In [19], deep learning (DL) has been introduced into automatic modulation classification (AMC) due to its outstanding identification performance. In [20], the Q-learning based intelligent algorithms have been proposed to protect the communication from the smart attacker, which can operate in spoofing, eavesdropping, interfering and silent modes. In [21], the authors extended to study the intelligent secure algorithms for the wireless communication systems in many application scenarios, such as the non-orthogonal multiple-access (NOMA) systems, imperfect channel estimation and multiple levels of primary users in cognitive networks. In [22,23], the deep reinforcement learning were incorporated into the strategy game such as the Weiqi and it can be found that the improved the winning rate of the machine. In particular, in 2016, "Alpha Go" adopted a deep reinforcement learning framework to defeat human go players.

In this paper, we study MEC networks for intelligent IoT, where multiple users have some computational tasks assisted by multiple computational access points (CAPs). By offloading some tasks to the CAPs, the system performance can be improved through reducing the latency and energy consumption, which are the two important metrics of interest in the MEC networks. We devise the system by proposing the offloading strategy intelligently through the deep reinforcement learning algorithm. In this algorithm, deep Q-network is used to automatically learn the offloading decision in order to optimize the system performance, and a neural network (NN) is trained to predict the offloading action, where the training data is generated from the environmental system. Moreover, we employ the bandwidth allocation in order to optimize the wireless spectrum for the links between the users and CAPs, where several bandwidth allocation schemes are proposed. In further, we use the CAP selection in order to choose one best CAP to assist the computational tasks from the users. Simulation results are finally presented to show the effectiveness of the proposed reinforcement learning offloading strategy. In particular, the system cost of latency and energy consumption can be reduced significantly by the proposed deep reinforcement learning based algorithm.

The organization of this paper is given as follows. After the introduction in this section, we will discuss the system model of MEC networks as well as the linearly weighted cost in Section 2. Then, we introduce how to intelligently optimize the system performance by using the DQN as well as the bandwidth allocation and CAP selection in Section 3. Section 4 will present the simulation results and conclusions are finally made in Section 5.

## 2. System model

In the paper, we consider the problem of offloading strategy design for MEC network, in order to determine how many tasks to computed by the CAPs. To further enhance the system performance, the bandwidth allocation is studied to optimize the wireless bandwidth among users and CAPs. Moreover, we consider the problem of CAP selection to choose one best CAP among multiple ones to assist the computation. Specifically, the system model is shown in Fig. 1, where we consider a task offloading network with $N$ users $\{u_n|n = 1, 2, \ldots, N\}$ and $M$ CAP nodes $\{e_m|m = 1, 2, \ldots, M\}$. All users have only one antenna while the MEC nodes have multiple antennas. For each user $u_n$, we assume that the computational task $\{l_n|n = 1, 2, \ldots, N\}$ can be arbitrarily divided into two parts: one part to be computed at local while the other part to be offloaded to the CAP. In Fig. 1, the user $u_n$

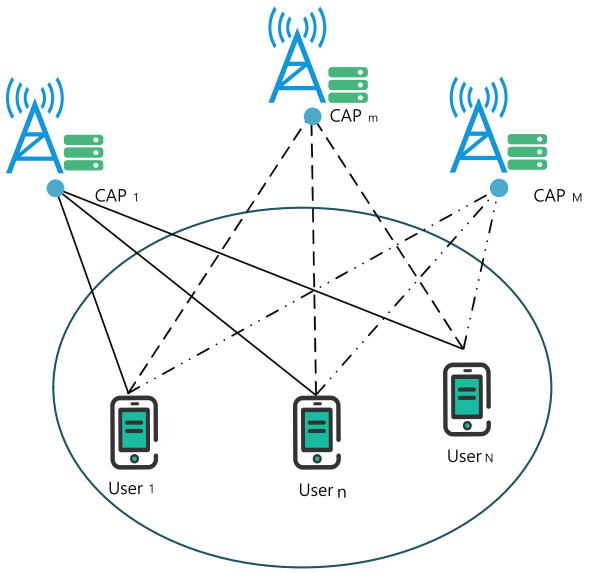



**Fig. 1.** System model of MEC network with multiple users.

firstly selects an optimal MEC node to offload the task $l_n$ through the channel parameters. After that, the user $u_n$ determines an offloading strategy, give by

$$\boldsymbol{\alpha}_n = [\alpha_{n,1}, \alpha_{n,2}, \ldots, \alpha_{n,m}, \ldots, \alpha_{n,M}], \tag{1}$$

where $m \in \{1, \ldots, M\}$ and the corresponding component $\alpha_{n,m} \in [0, 1]$ represents the percentage of the task $l_n$ to be offloaded to the MEC node $e_m$. Since the users only select one MEC node to offload tasks, there is at most one element greater than zero in the offloading strategy vector $\boldsymbol{\alpha}_n$. We denote the offloading ratio as

$$A_n = \sum_{m=1}^{M} \alpha_{n,m}. \tag{2}$$

Note that the offloading strategy $\boldsymbol{\alpha}_n$ includes the following three offloading scenarios:

(1) $A_n = 0$, In this scenario the task $l_n$ is computed at local.
(2) $A_n > 0$, In this scenario $A_n$ percents of the task $l_n$ is offloaded to the MEC node $e_m$ while the rest $(1-A_n)$percents of task is computed at local.
(3) $A_n = 1$, In this scenario the task $l_n$ is computed at the CAP node $e_m$.

We assume that the channels follow Rayleigh flat fading. Then, the transmission data rate from the $u_n$ to MEC node $e_m$ can be obtain from the Shannon theory as [24]

$$C_{n,m} = W_n \log_2\left(1 + \frac{P^n_{tran}|h_{n,m}|^2}{\sigma^2}\right), \tag{3}$$

where $W_n$ is bandwidth of the wireless $u_n$-$e_m$ link, $h_{n,m}$ $\mathcal{CN}(0, 2)$ denotes the channel gain of the $u_n$-$e_m$ link, $P^n_{tran}$ represents the transmit power at the user $u_n$, and $\sigma^2$ is the variance of the additive white Gaussian noise (AEGN) at the CAP nodes.

### 2.1. Local-computing model

We denote the computing capability (i.e., number of CPU cycles per second) at the user $u_n$ as $f_n$. Then, the local computation time can be obtain from the [12] as

$$T^n_{local} = \frac{l_n}{f_{n,m}}(1 - \alpha_{n,m})\ \gamma\ \omega, \tag{4}$$

where $\gamma$ is the conversion coefficient from million bits to bits, and $\omega$ is the number of cycles required for the CPU to compute per bit of task. In addition, the local computational energy consumption can be obtain from the [12] as

$$E^n_{Local} = T^n_{local} P^n_{local}, \tag{5}$$

where $P^n_{local}$ is the computational power at the user $u_n$.

### 2.2. Computing-offloading model

The transmission time consumed for the offloading link from $u_n$ to $e_m$ can be described as

$$T^n_{tran} = \frac{l_n}{C_{n,m}} \alpha_{n,m}\ \gamma\ \omega. \tag{6}$$

Similarly, the transmission energy consumption for the user $u_n$ can be described as

$$E^n_{tran} = T^T_{local} P^n_{tran}, \tag{7}$$

where $P^n_{tran}$ denotes the transmit power at the user $u_n$. Since the CAPs generally have steady energies, the computational consumption can be ignored for the CAPs. Moreover, the computation time at the CAP node $e^m$ can be denoted by

$$T^m_e = \frac{l_n}{F_{n,m}} \alpha_{n,m}\ \gamma\ \omega, \tag{8}$$

where $F_{n,m}$ denotes the computational capacity allocated for the user $u_n$ by the MEC node $e_m$. Since the size of the transmitted data returned is small enough, feedback latency and energy consumption can be ignored to simplify the problem. Therefore, we formulate the total system latency as

$$T_{total} = \sum_{n=1}^{N} \left( T^n_{local} + T^n_{tran} + T^m_e \right). \tag{9}$$

In addition the total energy consumption is formulated as

$$E_{total} = \sum_{n=1}^{N} \left( E^n_{local} + E^n_{tran} \right). \tag{10}$$

Note that minimizing both the total latency and the total energy consumption is a multiple objective optimization problem, which is however very complicated to implement in practice. To simplify the problem, and facilitate theoretical analysis, we consider a linear weighted objective function instead, which is given by

$$\Phi_m = \lambda T_{total} + (1 - \lambda) E_{total}, \tag{11}$$

where $\lambda \in [0, 1]$ is a weight factor. In short, the optimization problem of minimizing total latency and the total energy consumption is expressed as

$$\begin{aligned} &\min_{\{\alpha_{n,m}, W_n\}} \Phi_m \\ &\text{s.t.} \quad C_1\text{: } \alpha_{n,m} \in [0, 1] \\ &\qquad C_2\text{: } \sum_{n \in \mathcal{N}} W_n = W_{total}, \end{aligned} \tag{12}$$

## 3. System optimization

The main objective of this paper is to minimize the weighted cost $\Phi_m$. In order to achieve this goal, we firstly optimize the bandwidth allocation, and then we optimize the offloading strategy based on the allocated bandwidth. Finally, we give the CAP selection strategy based on the results of offloading strategy and bandwidth allocation.

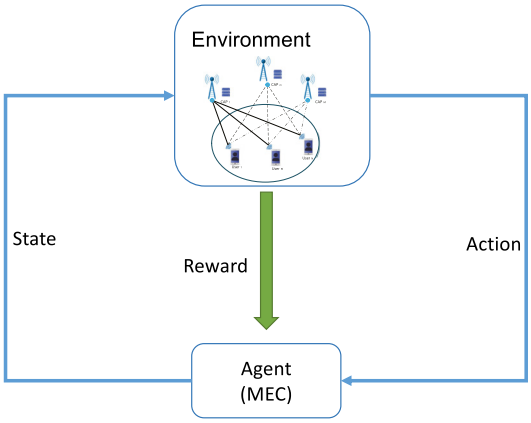



**Fig. 2.** Reinforcement learning model.

### 3.1. Bandwidth allocation optimization

In this part, we employ three bandwidth allocation criteria to assist the users and meanwhile to reduce the system weighted cost $\Phi_m$. The simplest bandwidth allocation strategy is the uniform allocation, which has the lowest computational complexity. In this criterion, $W_n$ is identical for each user $u_n$, given by

$$W_n = \frac{W_{total}}{N}. \tag{13}$$

This bandwidth allocation strategy is independent of specific channel conditions and task length. In order to incorporate the task length in to the bandwidth allocation, we present bandwidth allocation criterion II as,

$$W_n = \frac{l_n}{L} W_{total}, \tag{14}$$

which indicates that the allocated bandwidth $W_n$ depends on the task length of users.

Besides these two criteria, we future consider another dynamic bandwidth allocation strategy, which is related to the task offloading. Let i denote the number of iteration in the offloading process, and we use $\alpha^i_{n,m}$ to represent the updated task allocation strategy. Based on $\alpha^i_{n,m}$, we obtain a dynamic bandwidth allocation strategy as,

$$W_n = \frac{\alpha^i_{n,m}}{\sum_{n=1}^{N} \alpha^i_{n,m}} W_{total} \tag{15}$$

### 3.2. DQN-based resource allocation

After allocating bandwidth for a given CAP node, we will continue to optimize the offload strategy to reduce the system cost $\Phi_m$. Due to the complexity of resource allocation and task scheduling in MEC networks, it is hard to apply the traditional optimization methods solve this problem. Fortunately, the recent reinforcement learning technology has shown good results in solving mobile strategy problems, and it can be recognized as an ideal technology to optimize task offloading strategy in MEC networks.

#### 3.2.1. RL

As shown in Fig. 2, the framework of reinforcement learning consists of an agent and the corresponding environment which the agent interacting. In this scenario, each MEC node is viewed as an agent and everything except the MEC nodes is regarded as the environment. The agent makes decisions by observing the change

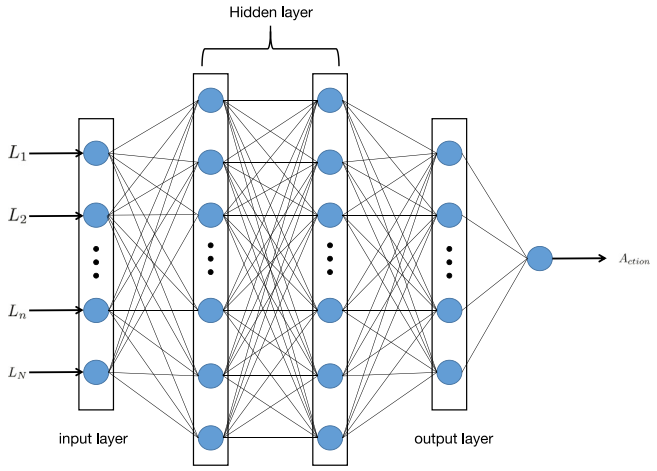



**Fig. 3.** Structure of the deep neural network.

of states. Reinforcement learning is an unsupervised learning technology, for which the agent can find the optimal behavior sequence via on-line learning. The agent repeatedly interacts with the environment by trial and error. Eventually, the agent modifies its own strategy to adapt to different environments to accomplish tasks.

In RL, it is of vital importance to design a proper reward function the specific decision problem, with the purpose to reward appropriate behaviors with respect to the current state. In this paper, we design the reward function as,

$$r_{=}\begin{cases} 1, & \text{if } (\ \Phi_t \text{ - } \Phi_{t-1}\ ) \text{ is larger than zero} \\ -1, & \text{other} \end{cases}, \tag{16}$$

where $\Phi_t$ and $\Phi_{t-1}$ denote the total cost at time slot $t$ and $t-1$, respectively.

The task offloading process can be regarded as a Markov decision process(MDP) for during the time domain. Let $\mathbf{S} = \{l_1\alpha_{1,m}, l_2\alpha_{2,m}, \ldots, l_N\alpha_{N,m}\}$ be the state space and $\mathbf{A} = \{\alpha_{1,m}, \alpha_{2,m}, \ldots, \alpha_{N,m}\}$ be the action space. The set of feasible actions for the state $S_t \in \mathbf{S}$ is $A_S^t$, which is a subset of $\mathbf{A}$. The transformation from $S_t$ to $S_{t+1}$ with an specific action $A_t$ follows the probability $P(S_{t+1}|S_t, A_t)$. The action is decided by a policy $\pi\colon S \to A$. The policy is obtained by training the agent through reinforcement learning.

### 3.2.2. DQN

The traditional reinforcement methods such as Q-learning and Sarsa learning have a common feature, which is to use tables to store state value functions. However, for the general task offloading problem, the value function cannot be saved in the form of table due to huge state dimensions. Therefore, we choose DQN to solve the task offloading problem. Compared with Q-learning, DQN uses a deep neural network(DNN) with parameters $\omega$ as a value function approximator to solve the task offloading problem. As shown in Fig. 3, we take state $s$ as the input of the DNN. The DNN consists of an input layer $H^i$, the $k$ shared hidden layers $\{H_1, H_2, \ldots, H_k\}$, and an input layer $H_j^i$. In order to get the next action, we set a greedy selection strategy on the output of the DNN, with probability $\epsilon$ to select a random action $a$, which can be described as follows,

$$A = \begin{cases} random, & \text{when the probability is } \epsilon \\ \arg\min_a(Q(s_t, a; \omega)), & \text{when the probability is } 1-\epsilon \end{cases} \tag{17}$$

There is a certain correlation between the interaction sequence and the state action in RL. If we train the neural network directly, the effect of the model will not be good as expected. To solve this problem, we adopted the experience replay structure proposed by DeepMind team in NeurIP in 2013.

The replay buffer includes two parts: the collecting samples and the sampling samples. The collected samples are stored in replay buffer according to the time sequence. If the replay buffer is full of samples, them the new samples will overwrite the oldest sample in time sequence. Generally speaking, a batch of samples will be randomly sampled from the cache evenly for learning, and the training effect will be more stable. At the same time, a sample will be trained many times to improve the sample utilization rate.

Note that the traditional reinforcement learning updates the status value based on the return value of the current time and the next estimated value. But the instability of the data causes the neural network training results to fluctuate at each iteration. These fluctuations will be reflected at the next iteration, and hence the training result is difficult to be stable. In order to deal with the deviation in temporal difference and reduce the impact of correlation, we need to decouple the two parts as much as possible. Hence, we introduce the target network. Firstly, the two models use the same parameters before the training. Secondly, during the training process, behavior network is responsible for interacting with the environment and getting interaction samples. Then, in the learning process, the target value is calculated by target network, and the target value is obtained by comparing the estimated values of the target network and then the behavior network, and the behavior network is updated. Finally, when the iterations reach a certain number, the parameters of the behavior network are synchronized to the target network, and the next stage of learning can be carried out. Similar to the supervised learning, we define the loss function of the DQN as the variance between the target value $Q_{targe}$ and the predicted value $Q(s, a:\omega)$ weights $\omega$ to minimize the loss,

$$Loss_\omega = ((r - \gamma \arg\min_a(Q(s', a'; \omega'))) - Q(s, a:\omega))^2 \tag{18}$$

Therefore, the target value is fixed in a certain period of time through the target network and the target network reduces the volatility of the model eventually.

The process of DQN algorithm is described in Algorithm 1. The main steps are as follows.

(1) Using a neural network with a parameter of $\omega$ as the approximator of $Q$ value.
(2) Defining a loss function using the mean square error of the $Q$ value.
(3) Calculating the gradient of loss function for the parameter $\omega$.
(4) Using Stochastic Gradient Descent(SGD) to optimize the parameters.

### 3.3. CAP selection

After optimization of bandwidth allocation and offloading, in order to further reduce the system cost $\Phi_m$ we performed a CAP selection operation. In this work, we propose a method for choosing the best CAP to help users perform calculations.

For this MEC network, the user firstly sends some pilot signals, from which the MEC estimates the associated channel parameters. Then, the channel which has the smallest gain is measured among the N channels,

$$\theta_m = \min_{m\in[1,M]} \{|h_{1,m}|^2, |h_{2,m}|^2, \ldots, |h_{N,m}|^2\}. \tag{19}$$

**Algorithm 1** DQN-Based MEC network Resource Allocation Optimization

**Input:** user task $l$,radio bandwidth resource $W_{total}$, computing capability of users and CAP nodes
**Output:** task offloading result a
Initialize replay memory D to capacity N
Initialize action-value function Q with weight $\omega$
Initialize target action-value function $\widehat{Q}$ with weights $\omega' = \omega$
initialize states $s_1$
**for** t = 1,T **do**
with probability $\omega$ select a random action $a_t$

$$a_t = \begin{cases} random, & \epsilon \\ \arg\min_a(Q(s_t, a; \omega)), & 1-\epsilon \end{cases}$$

Perform $a_t$ in the environment, observe the reward r and the next state $s_{t+1}$
Store transition ($s_t$,$a_t$,$r_t$,$s_{t+1}$) in D
Sample random minibatch of transitions ($s_i$,$a_i$,$r_i$,$s_{i+1}$) from D
$y_i = r_i + \arg\min_a(\widehat{Q}(s'_t, a'; \omega')$
Performing gradient descent
Interval C step to update $\widehat{Q} = Q$
**end for**

Each CAP is associated with of $\theta_m$. From the set $\{\theta_m | 1 \leq m \leq M\}$, we select one best CAP which has the largest $\theta_m$ among M ones,

$$e^* = \arg\max \theta_m. \tag{20}$$

## 4. Simulation results

In this part, we use several bandwidth allocation strategies and CAP selection methods to evaluate the proposed optimization algorithm. All channels in the network experience Rayleigh flat fading. If not specified, the transmit and computing powers at the users are set to 2 W and 3 W, respectively. The deep neural network has two hidden layers. The CPU of each CAP has the same computing power with computational capacity of $6.3 \times 10^8$ cycle per second (cyc/s). Moreover, the six users have different computational capacities, which are set to $1.4 \times 10^8$ cyc/sec, $0.21 \times 10^8$ cyc/s, $0.95 \times 10^8$ cyc/s, $0.13 \times 10^8$ cyc/s, $0.53 \times 10^8$ cyc/s and $0.52 \times 10^8$ cyc/s, respectively. The task sizes of the six users are set to 5.3 Mb, 3.5 Mb, 4.6 Mb, 3.0 Mb, and 4.2 Mb, respectively. In further, the total bandwidth of the wireless links is set to 10 MHz, so that $B_{total} = 10$ MHz.

Fig. 4 shows the convergence of the proposed DQN algorithm, where M = 2, N = 5, and $B_{total} = 10$ MHz. For the convinced of notation, we used "R-DQN", "L-DQN", and "E-DQN" to represent the DQN with the bandwidth allocation based on the offloading ratio in the iteration, bandwidth allocation by the sub-task length and equal bandwidth allocation, respectively. From (4), we can see that the DQN with several bandwidth allocation schemes converge swiftly, and after 8000 iterations, system can achieve stable performance. Moreover, the performance of L-DQN is better than that of E-DQN, as L-DQN in corporates the length of sub-tasks into the bandwidth allocation process. In further, we can find that R-DQN outperforms E-DQN and L-DQN, indicating that the offloading ratio in the iterative process can help allocation the wireless bandwidth very effectively.

In Fig. 5, we show the relationship between the system cost $\Phi$ and several offloading strategies and the weight factors $\lambda$, where M = 2, N = 5, and $\lambda$ varies from 0.1 to 0.9. In Fig. 5 we use 'All-Local' and 'All-CAP' to represent that the tasks are computed locally and by the CAPs, respectively. The equal bandwidth allocation scheme is adopted in Fig. 5. From this figure, we can find

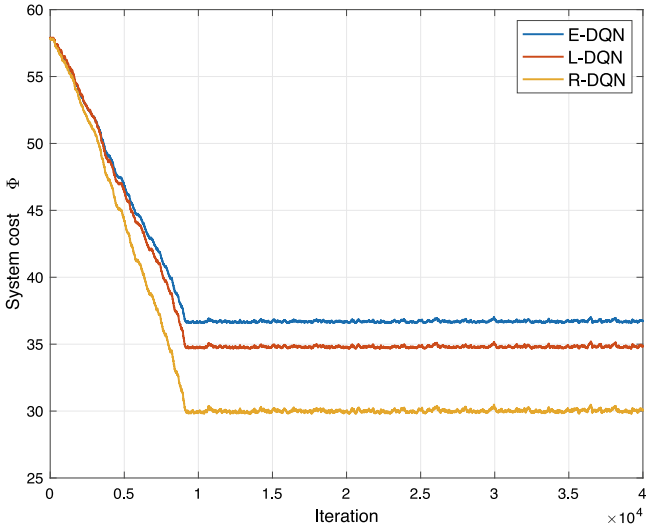



**Fig. 4.** Convergence of the DQN algorithm versus iteration.

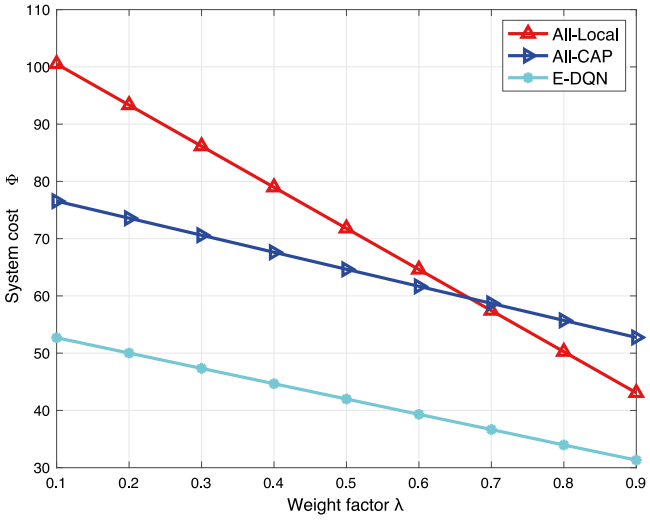



**Fig. 5.** Comparison of the three offloading strategies versus the weight factor $\lambda$.

that the proposed E-DQN outperform the 'All-Local' and 'All-CAP' for various values of $\lambda$, indicating that the proposed scheme can efficiently utilize the computational resources among the users and CAPs. Moreover, the cost of 'All-CAP' is smaller then that of 'All-Local' when $\lambda$ is small, as using the CAPs to compute the tasks can help reduce the energy consumption. On the contrary, when $\lambda$ is large, 'All-CAP' becomes course them 'All-Local', simple the transmission latency becomes the bottle neck of the system cost.

Fig. 6 shows the impact of number users on the system cost with several offloading strategies, where M = 2 and N varies from 1 to 6. For performance comparison, we present the cost of the proposed 'E-DQN', 'All-Local' and 'All-CAP' in this figure. From Fig. 6, we can find that the system costs increases with a larger value of N, as more users give more burden of the computational tasks on the system. Moreover, for various values of N, the proposed 'E-DQN' outperforms the 'All-Local' and 'All-CAP', which further validates the effectiveness of the proposed scheme in scheduling the computational resources in the system.

Then, we investigate the impact of the computational capability of the CAP on the cost. From Fig. 7, we can observe that the maximum and minimum CPU frequency of the set to CAP are $6.3 \times 10^8$ cyc/sed and $4.1 \times 10^8$ cyc/se, respectively. The

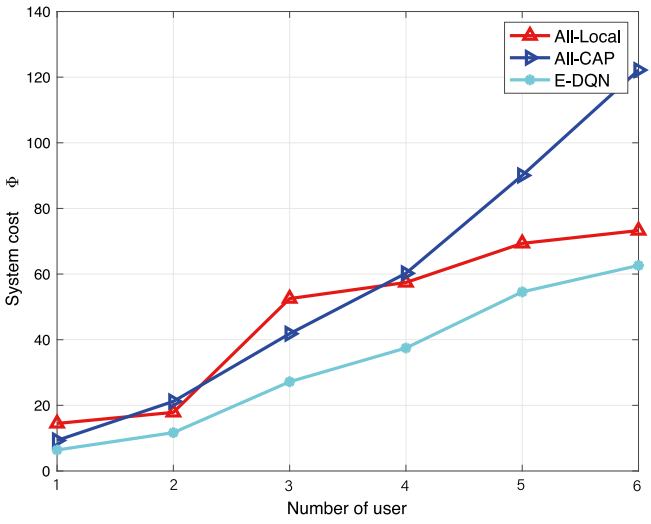



**Fig. 6.** Comparison of the three offloading strategies versus the number of users.

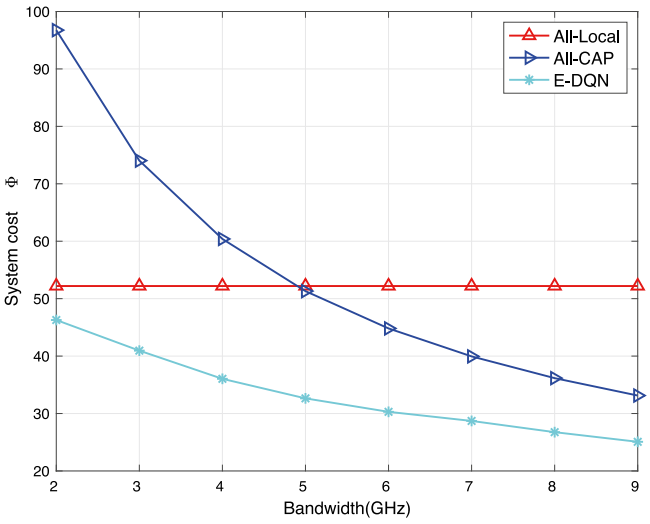



**Fig. 8.** Comparison of the three offloading strategies versus the bandwidth $W_{total}$.

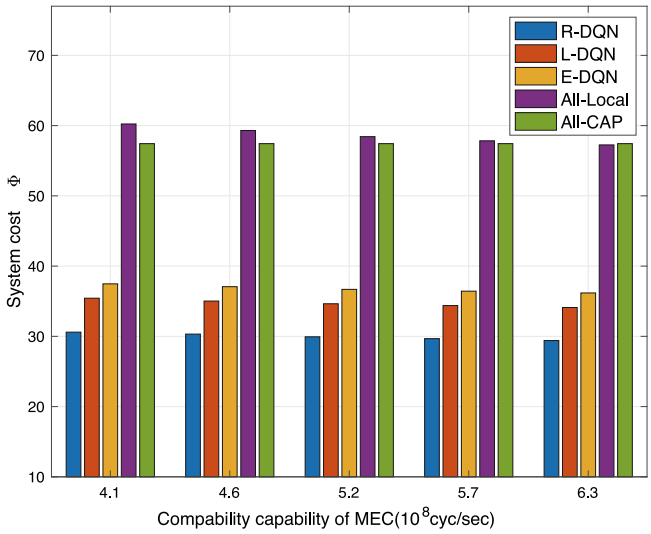



**Fig. 7.** The computing capability of MEC.

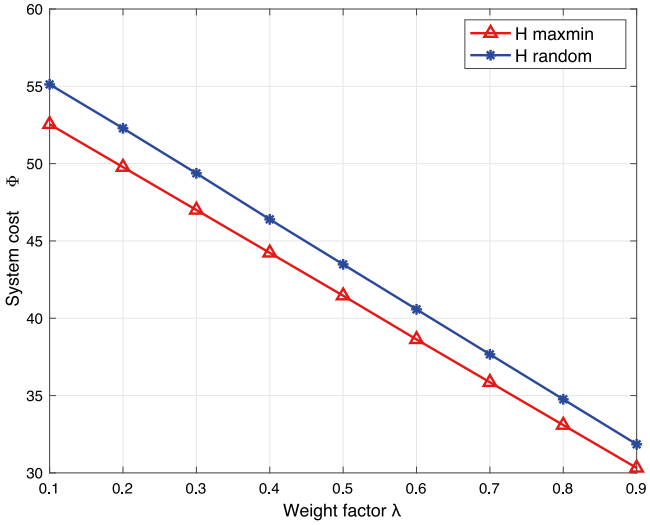



**Fig. 9.** Comparison of the CAP selection method with E-DQN versus the weight factor λ.

performance of 'All-Local' computing and 'All-CAP' computing are still the worst. We can observe that when E-DQN, E-DQN or E-DQN algorithm is adopted, the average energy consumption of the system decreases when the computing capability of the servers increases, because these three schemes can upload more tasks to the CAPs. The results indicate that offloading tasks to the CAP server can be completed faster, thereby reducing the cost. We can also observe that user's offloading rates increase as the computational power of the edge server increases. This indicates that when the user finds that the computing resources of the CAP are sufficient, the user is more willing to upload more tasks to the CAP.

In Fig. 8, we compare the cost performances of the several offloading strategy versus the wireless bandwidth, where M = 2, N = 5, and the bandwidth $B_{total}$ varies from 2 GHz to 9 GHz. We can find from Fig. 8 that for various values of $B_{total}$, the proposed E-DQN scheme outperforms the 'All-Local' and 'All-CAP', which further validates the effectiveness of the proposed offloading strategy. Moreover, unlike the 'All-local', the proposed E-DQN and 'All-CAP' have better performances when the value of $B_{total}$ becomes larger. This is because greater bandwidth can help reduce transmission delays and transmission energy consumption.

In Fig. 9, we show the effect of CAP selection on the system cost performance versus the weight factor λ, where M = 2, N = 5, and $B_{total}$ = 10 MHz. For comparison, we plot the result of random CAP selection in Fig. 9 as a benchmark. As observed from Fig. 9, we can find that for various of λ, the CAP selection scheme outperforms the random selection scheme, since the former can exploit the wireless links for improving the transmission latency and energy consumption. The validity of the proposed research in this work is further verified.

## 5. Conclusions

This paper studied MEC networks for intelligent IoT, where multiple users have some computational tasks assisted by multiple CAPs. We devised the system by proposing the offloading strategy intelligently through the deep reinforcement learning algorithm. In this algorithm, Deep Q-Network was used to automatically learn the offloading decision in order to optimize the system performance, and a neural network (NN) was trained to predict the offloading action, where the training data was generated from the environmental system.

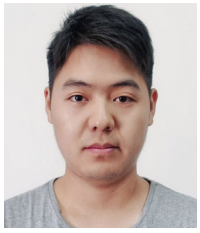
Moreover, we employed the bandwidth allocation in order to optimize the wireless spectrum for the links between the users and CAPs, where several bandwidth allocation schemes were proposed. In further, we used the CAP selection in order to choose one best CAP to assist the computational tasks from the users. Simulation results were finally presented to show the effectiveness of the proposed reinforcement learning offloading strategy. In particular, the system cost of latency and energy consumption could be reduced significantly by the proposed deep reinforcement learning based algorithm.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References


[1] N. Zhao, X. Liu, Communications, caching, and computing oriented small cell networks with interference alignment, IEEE Commun. Mag. 54 (9) (2016) 29–35.
[2] K. He, Ultra-reliable mu-mimo detector based on deep learning for 5g/b5g-enabled iot, Physical Communication PP (99) (2020) 1–8.
[3] J. Zhao, Power allocation based on genetic simulated annealing algorithm in cognitive radio networks, Chin. J. Electron. 22 (1) (2013) 177–180.
[4] J. Xia, D. Deng, A note on implementation methodologies of deep learning-based signal detection for conventional mimo transmitters, IEEE Transactions on Broadcasting PP (2020) 1–2.
[5] J. Zhao, Power control algorithm of cognitive radio based on non-cooperative game theory, China Commun. 10 (11) (2013) 143–154.
[6] J. Xia, Opportunistic access point selection for mobile edge computing networks, IEEE Transactions on Wireless Communications PP (2020) 1–12.
[7] B. Wang, F. Gao, S. Jin, H. Lin, G.Y. Li, Spatial- and frequency-wideband effects in millimeter-wave massive MIMO systems, IEEE Trans. Signal Process. 66 (13) (2018) 3393–3406.
[8] R. Zhao, Intelligent physical-layer secure communications for beyond 5g wireless networks, Physical Communication PP (99) (2020) 1–8.
[9] X. Hu, C. Zhong, X. Chen, W. Xu, Z. Zhang, Cluster grouping and power control for angle-domain MmWave MIMO NOMA systems, IEEE J. Sel. Top. Sign. Proces. 13 (5) (2019) 1167–1180.
[10] G. Gui, M. Liu, F. Tang, N. Kato, F. Adachi, 6G: Opening new horizons for integration of comfort, security and intelligence, IEEE Wirel. Commun. (2020) 1–7.
[11] J. Xia, D. Deng, Cache-aided mobile edge computing for B5G wireless communication networks, EURASIP J. Wireless Commun. Networking 2020 (1) (2020) 15.
[12] Z. Zhao, A novel framework of three-hierarchical offloading optimization for MEC in industrial IoT networks, IEEE Trans. Ind. Inf. 16 (8) (2020) 5424–5434.
[13] Y. Guo, Intelligent offloading strategy design for relaying mobile edge computing networks, IEEE Access 8 (2020) 35127–35135.
[14] Z. Zhao, Intelligent mobile edge computing with pricing in Internet of Things, IEEE Access 8 (2020) 37727–37735.
[15] J. Xia, K. He, A MIMO detector with deep learning in the presence of correlated interference, IEEE Trans. Veh. Technol. 69 (4) (2020) 4492–4497.
[16] K. He, D. Deng, Generic deep learning based linear detectors for MIMO systems over correlated noise environments, IEEE Access 8 (2020,) 29922–29929.
[17] G. Gui, F. Liu, J. Sun, J. Yang, Z. Zhou, D. Zhao, Flight delay prediction based on aviation big data and machine learning, IEEE Trans. Veh. Technol. 69 (1) (2020) 140–150.
[18] G. Gui, Z. Zhou, J. Wang, F. Liu, J. Sun, Machine learning aided air traffic flow analysis based on aviation big data, IEEE Trans. Veh. Technol. 69 (5) (2020) 4817–4826.
[19] Y. Wang, G. Gui, H. Gacanin, T. Ohtsuki, H. Sari, F. Adachi, Transfer learning for semi-supervised automatic modulation classification in ZF-MIMO systems, IEEE J. Emerg. Sel. Top. Circuits Syst. 10 (2) (2020) 231–239.
[20] S. Lai, Intelligent secure mobile edge computing for beyond 5g wireless networks, Physical Communication PP (99) (2020) 1–8.
[21] C. Li, Cache-enabled physical-layer secure game against smart UAV-assisted attacks in B5G NOMA networks, EURASIP J. Wireless Commun. Networking 2020 (1) (2020) 7.
[22] Mastering the game of Go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489.
[23] J.X. Wang, Z. Kurth-Nelson, D. Kumaran, D. Tirumala, H. Soyer, J.Z. Leibo, D. Hassabis, M. Botvinick, Prefrontal cortex as a meta-reinforcement learning system, Nature Neurosci. (2018).
[24] C. Shannon, A mathematical theory of communication, Bell Syst. Tech. J. 27 (4) (1948) 623–656.

**Rui Zhao** received the B.E. degree in computer science and technology from the Bohai University , Jinzhou, China, in 2018. He is currently pursuing the M.S. degree at the Guangzhou University, Guangzhou, China. His current research interests include machine learning , and mobile edge computing resource scheduling algorithms.

**Xinjie Wang** was born in 1980. He received his B.S. Degree in Electrical and Information from Qufu Normal University, China, in 2003, the Master Degree in Signal and Information processing from Sun Yat-sen University, China, in 2005, and his Ph.D. degree in Computer Application Technology from Ocean University of China, China, in 2016. Since 2005, he is an lecturer in school of information and control engineering, Qingdao University of Technology, China. His research interests include cognitive radio networks, physical layer security, energy harvesting, drone communications and heterogeneous networks.

**Junjuan Xia** received the bachelor degree from the department of computer science from Tianjin University in 2003, and obtained the master degree from the department of electronic engineering from Shantou University in 2015. Now she works for the School of Computer Science and Cyber Engineering, Guangzhou University as a laboratory assistant. Her current research interests include wireless caching, physical-layer security, cooperative relaying and interference modeling.

**Liseng Fan** obtained the doctoral degree from Tokyo Institute of Technology, Tokyo, in the year of 2008. He is now a Professor at the School of Computer Science and Cyber Engineering of Guangzhou University. He has published more than 40 papers on the IEEE journal and IEEE conferences. His main research interests include the information security, wireless networks, and the artificial intelligence. His recent research interest is the application of artificial intelligence into the wireless networks.