Reinforcement Learning-based Computing and Transmission Scheduling for LTE-U-Enabled IoT

Hongli He*, Hangguan Shan*, Aiping Huang*, Qiang Ye†, Weihua Zhuang†
*College of Information Science & Electronic Engineering, Zhejiang University, Hangzhou, China
{hongli_he, hshan, aiping.huang}@zju.edu.cn
†Department of Electrical & Computer Engineering, University of Waterloo, Canada
{q6ye, wzhuang}@uwaterloo.ca

***Abstract*—To facilitate the private deployment of industrial Internet-of-Things (IoT), applying LTE in unlicensed spectrum (LTE-U) is a promising approach, which both tackles the problem of lacking licensed spectrum and leverages an LTE protocol to meet stringent quality-of-service (QoS) requirements via centralized control. In this paper, we investigate the computing offloading problem in an LTE-U-enabled network, where the task on an IoT device is carried out either locally or is offloaded to the LTE-U base station (BS). The offloading policy is formulated as an optimization problem to maximize the long term discounted reward, considering both task completion profit and the task completion delay. Due to the stochastic task arrival process at each device and the Wi-Fi's contention-based random access, we reformulate the computing offloading problem into a Q-learning problem and solve it by a deep learning network-based approximation method. Simulation results show that the proposed scheme considerably enhances the system performance.**

***Index Terms*—Mobile edge computing, offloading, IoT, LTE-U, deep reinforcement learning.**

I. Introduction

Over the last decade, the soaring proliferation of Internet-of-Things (IoT) has attracted great attention from both academia and industry. IoT is to integrate a large number of miscellaneous devices, including vehicles, wearable items, home appliances, and sensors to achieve ubiquitous information access and seamless communication interaction [1]. Benefiting from the IoT paradigm, various new applications have been developed, encompassing smart grids, virtual power plants, smart homes, intelligent transportation and smart cities [2][3].

Due to hardware constraints of IoT devices, including limited computational and energy resources, fulfilling computationally intensive tasks for intelligent IoT applications requires remote execution from computationally powerful clouds. However, the cloud computing resources are normally deployed far from the end-devices, resulting in high communication delay and substantial stress on backhaul and fronthaul links [4]. To this end, shifting some of the tasks back to the vicinity of IoT devices is a potential approach to address these issues, which facilitates the advent of the edge computing. By making the computing resources available in the proximity of the end-devices, the computation-intensive data can be processed near the IoT devices, promising computation augmenting services with short latency [5].

However, it is difficult for some non-cellular-carrier enterprises to reap these benefits. In deploying industrial IoT, private organizations want to take advantage of the centralized control pattern of the cellular network; However, they are not authorized to use the licensed spectrum [6]. Therefore, applying the LTE-like technology to a private industrial IoT network is a promising solution, especially when there are no available licensed radio resources. There are existing studies on the deployment of private industrial IoT systems with unlicensed spectrum and the edge computing in LTE-U networks. Two technologies of low-power WAN (LPWAN) operating in unlicensed spectrum, SigFox and LoRa (Long Range) , are evaluated and compared in terms of coverage range, frequency bands and data rates in [7]. In [8], the challenges of applying the edge computing in the unlicensed spectrum are identified, in terms of security, reliability, and network coordination. In [9], an scheduling algorithm tolerant to out-of-date network knowledge is proposed to relieve the tasks' feedback in the edge computing for an IoT network. The framework applying both the perturbed Lyapunov function and the knapsack problem modelling achieves an asymptotically optimality with only partial information. In [9], an online task offloading algorithm with Lyapunov optimization is proposed to handle the tradeoff in terms of average response time, average monetary, and energy costs in the IoT. However, the dynamic channel availability and the On-off channel access scheme in the LTE-U network will have great impact on the deployment of edge computing system for the IoT, which is seldom studied in the existing works.

In this paper, we consider the tasks offloading problem in an LTE-U enabled IoT network, where high dynamics of the channel availability pose significant challenges on the reliable data transmission and delay evaluation. Taking account of the stochastic arrival process of computing tasks and the highly fluctuant channel availability, we formulate the problem as a Markov decision based (MDP)-framework and derive the state evolution equation under different policies. In order to reduce the exponential space complexity due to the high dimension of the states in the traditional tabular method, we utilize a deep learning network to approximate the Q values of different state-action pairs to learn the optimal policy. Simulation results demonstrate that our proposed algorithm considerably improves the system performance.

The remainder of the paper is organized as follows. System model is presented in Section II. In Section III, we formulate the edge computing offloading problem into a deep Q-learning

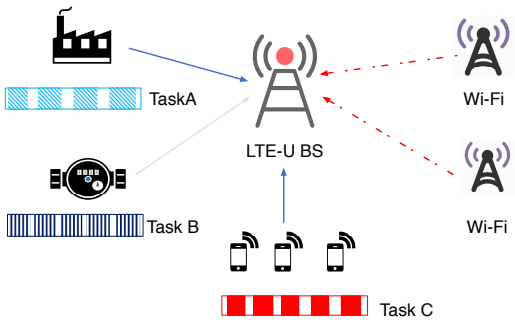

Fig. 1. The deployment scenario.

(DQL) framework to decide where the new task should be executed. Numerical results are given in Section IV, followed by conclusions in Section V.

II. System Model

A. Deployment Scenario

We consider a mobile edge computing network, consisting of one LTE-U BS and K LTE-U-enabled IoT devices, illustrated in Fig. 1. Let $\mathcal{K} = \{1, 2, ..., K\}$ be the set of indexes of the K devices, and the BS be indexed by 0. The BS operates in an unlicensed spectrum and there is a band with bandwidth B dedicated for the edge computing offloading service. The BS has both edge computing and wireless transmission functions to serve the devices. The transmission and computing process evolves over time according to a slotted structure indexed by t ($t > 0$) and the slot length is τ. The devices need to keep performing some tasks based on the steady stream of data collected from the environment. Because the devices are usually designed for some specific services, we consider each device only performs one certain kind of task [10]. The data structure of different tasks to be processed keeps the same in the temporal order for one device, but varies among different devices. Due to limited computing capacity of the devices and the various delay requirements for different real-time applications, some computing tasks should be offloaded to the BS which has a much more powerful computing capability. As LTE-U is a technology deployed in an unlicensed spectrum, it is required to fairly share with the other legacy networks, e.g. Wi-Fi, which introduce some inevitable interference to the LTE-U networks. Let $\vartheta(t)$ denote an abstract of the external effect (including the number of of active Wi-Fi STAs, their transmission power, and their traffic load etc.) on the transmission of the LTE-U network in the tth slot. The interference at the BS receiver is represented as a stochastic function $f(\vartheta(t))$. The value of $f(\vartheta(t))$ is known at the beginning of each slot via the reporting mechanism in LTE and remains the same in the whole slot.

B. Task Model

The task generated from device k is denoted as $M_k \triangleq (L_k, R_k, X_k)$, where L_k is the task input-data size (in the unit of bit), R_k is the task completion reward, and X_k is the computation workload/intensity (normally in CPU cycles per bit) [11]. We consider a linear dependancy between two neighbouring tasks generated from a device, i.e., a subsequent task can be performed upon the completion of its previous task. We assume that at most one task is generated at a device within each time slot, since the duration of each time slot is set to a small value. The probability of a new task arriving at device k within slot t follows a Bernoulli distribution with parameter $p_k(t)$. We denote the computing capacity (in the unit of cycles per second) of device k as C_k ($k \in \mathcal{K}$), and C_0 represents the computing capacity of the BS, and the computing capacity of the BS is much more computationally powerful than its associated devices, i.e., $C_0 >> C_k, \forall k \in \mathcal{K}$.

C. Transmission Model

Since the unlicensed spectrum is shared between LTE-U and Wi-Fi networks to guarantee the fairness of the channel utilization [12], the listen-before-talk (LBT) mechanism is adopted for the LTE-U network to mitigate the transmission interference with Wi-Fi users. Therefore, the category 4 channel access scheme (LBT with random back-off and a contention window of variable size) is adopted in the LTE-U network to sense and occupy the unlicensed spectrum resources [13]. The channel access process of the LTE-U network is initiated by the BS and is described by a two-stage backoff process, $s_C = (a(t), b(t))$, where $a(t)$ is the contention stage and $b(t)$ is the backoff counter ($t > 0$). When the LTE-U intends to access the unlicensed channel, the BS first generates a backoff counter $b(t)$ randomly between 0 and the initial backoff window size W_0, and then begins to sense the channel at the beginning of each slot. If the channel is idle, its backoff counter $b(t)$ is reduced by 1; Otherwise the counter keeps frozen. Only when the backoff counter $b(t)$ is decreased to 0, can the LTE-U BS access the unlicensed channel and schedule the devices to do their uplink transmission. Once transmission collisions happen with other networks' devices, the backoff stage of the BS $a(t)$ is increased by one, and the backoff counter is randomly re-selected in a doubled window size. When the channel is available for the LTE-U network, the bandwidth resources are equally shared among the associated devices and the uplink transmission rate of the device k at slot t is given by

$$r_k(t) = \frac{B}{K} \log \left(1 + \frac{p_k |h_k(t)|^2}{\sigma^2 + f(\vartheta(t))} \right) \quad (1)$$

where p_k is the fixed transmission power of device k, $h_k(t)$ is the channel gain, including the path loss and Rayleigh fading, from the device k to the BS, and σ^2 is the noise power.

D. Offloading Policy and System Status

When a new task is generated at device k, the task is either computed locally or offloaded to the BS, which is decided by the controller in the BS. Let $A_k(t) = 0$ denote the action that the new task is to be processed by the device k in the slot $t + 1$ and $A_k(t) = 1$ denote the action that the new task is to be processed by the BS. The decision epochs for task processing will be discussed in detail in Section II-E. Based on the different task offloading policies, the device-dependent task status at slot t is described as vector $\boldsymbol{z}(t) = (z_1(t), z_2(t), ..., z_k(t), ..., z_K(t))$ where $z_k(t) \in \{0, 1, 2, 3, 4\}, k \in \mathcal{K}$. $z_k(t) = 0$ indicates that the task of device k is scheduled locally for processing (including the case that there is no task to process) in slot t, $z_k(t) = 1$

indicates that the input data of the task of device k is being forwarded to the BS, to the BS in slot t, $z_k(t) = 2$ denotes that the task of device k is waiting in the mobile edge computing queue in slot t, $z_k(t) = 3$ denotes that the task of device k is being computed at the BS in slot t, $z_k(t) = 4$ means that the task of device k is completed at the BS and is waiting for transmitting the output data in slot t. Note that task status of all devices are updated at the beginning of each time slot. The status transitions depend on the availability of transmission resources and computing resources, which will be elaborated in the following subsections.

E. Task Buffer

Each device has a task buffer containing all the tasks to be scheduled. We use $B_k(t)$ to represent the number of tasks in the task buffer at the beginning of time slot t. The evolution of the buffer size of device k depends on both the new task arrival and the in-service task processing and its update is described as

$$B(t+1) = B'_k(t) + O_k(t) \tag{2}$$

where the $B'_k(t)$ is the updated buffer size due to the task processing and $O_k(t)$ is is a random binary variable indicating whether a new task arrives at device k within slot t, where $O_k(t) = 1$ with probability $p_k(t)$ represents that a new task is generated within slot t, and $O_k(t) = 0$ with probability $(1 - p_k(t))$ represents that no task is generated. The update of the buffer size due to in-service task processing at device k is given by

$$B'_k(t) = \begin{cases} \max\left\{B_k(t) - \dfrac{C_k\tau}{L_k X_k}, \lfloor B_k(t)\rfloor^-\right\}, & \text{if } z_k(t) = 0 \\ B_k(t) - 1, & \text{if } z_k(t) = 4 \text{ and } b(t) = 0 \\ B_k(t), & \text{otherwise} \end{cases} \tag{3}$$

where $\lfloor B_k(t)\rfloor^-$ is the modified floor function maps to the greatest integer less than $B_k(t)$. The top branch of (3) represents the case that when the task of device k is locally processed, the buffer size of device k is reduced by the normalized fraction at the end of slot t, but the subsequent task at device k cannot be processed in the same slot, i.e., $B'_k(t) \geq \lfloor B_k(t)\rfloor^-$. The middle branch of (3) indicates the case that when the task of device k is completed at the BS, the buffer size of device k is only updated after transmitting the output data back when the channel is available.

The decision epochs for device k can be divided into two cases. The first case is that there is some task completed (either is completed locally or obtain the output data from the BS) and the task buffer size is still larger than 0, which is denoted by $\mathbb{E}_k = \{t | t \in \mathcal{T}_k^0,\ B_k(t+1) > 0\}$, where $\mathcal{T}_k^0$ is the set of time slots in which some tasks of device k are completed, i.e., $\mathcal{T}_k^0 = \{t | B'_k(t) \in \mathbf{Z},\ B'_k(t) < B_k(t)\}$; The second case is that the task buffer of device k is empty at the beginning of slot t but some new task arrives within slot t, which is denoted by $\mathscr{E}_k = \{B_k(t) = 0, O_k(t) > 0\}$. Therefore, the overall decision epochs for device k can be denoted as $\mathcal{E}_k = \mathbb{E}_k \bigcup \mathscr{E}_k$. Note that the task scheduling decision epochs are logically after the new task arrival when they are in the same time slot.

F. Local Transmission Queue

When a task of device k is decided to be offloaded to the BS, the device k should transmit its input data with size L_k to the BS. The normalized transmission queue length of device k at the beginning of time slot t is denoted by $Q_k(t)$. Since the tasks at the same device are processed sequentially and device k only starts to transmit the input data of a subsequent task after receiving the completely processed output data of the previous task (from the BS or locally), the value of the transmission queue length cannot exceed 1, i.e., $Q_k(t) \in [0, 1]$. The transmission queue size of device k is updated given by

$$Q_k(t+1) = \begin{cases} \max\left\{Q_k(t) - \dfrac{r_k(t)\tau}{L_k}, 0\right\}, & \text{if } z_k(t) = 1 \text{ and } b(t) = 0 \\ Q_k(t) + \mathbf{1}(A_k(t) = 1), & \text{if } t \in \mathcal{E}_k \\ Q_k(t), & \text{otherwise} \end{cases} \tag{4}$$

where $\mathbf{1}(\cdot)$ is the indicator function. The top branch of Eq. (4) indicates the case that the task of device k is forwarding its input data to the BS when the channel is available and its transmission queue size is reduced by the normalized fraction; The middle branch of Eq. (4) indicates the case that when the slot is a decision epoch of device k and the task is to be offloaded to the BS, the transmission queue of device k should be added by one.

Upon completing the transmission of the device k's input data, the task of device k is put into the edge computing queue and we denote the transmission completion epochs as $\mathcal{T}_k^1 = \{t | Q_k(t+1) = 0,\ Q_k(t) > 0\}$ and $\mathcal{T}^1 = \bigcup_{k \in \mathcal{K}} \mathcal{T}_k^1$.

G. Edge Computing Queue

The tasks in the edge computing queue of the BS are served in a first-come-first-serve order, and therefore the priority value $P_k(t)$ is used to specify the service order of the tasks in the edge computing queue at the beginning of slot t. When a new task is put into the edge computing queue, all the priority values of the tasks in the queue (including the new task) are added by one, which guarantees that the earlier tasks and can be served with a higher priority. This process is denoted by

$$\overline{P}_k(t) = \begin{cases} P_k(t) + 1, & \text{if } t \in \mathcal{T}^1 \text{ and } k \in \mathscr{K}(t) \\ P_k(t), & \text{otherwise} \end{cases} \tag{5}$$

where $\mathscr{K}$ is the set of devices whose tasks are in the edge computing queue, i.e., $\mathscr{K}(t) = \{k | Q_k(t) > 0 \text{ or } t \in \mathcal{T}_k^1\}$. Moreover, in order to model the edge computing process for the task of device k, the decimal part of $P_k(t)$ is denoted as the normalized remaining fraction of device k's task being processed at the BS. The impact of edge computing on the priority values of different devices is described as

$$\hat{P}_k(t) = \begin{cases} \max\left\{\overline{P}_k(t) - \dfrac{C_0\tau}{L_k X_k}, \lfloor \overline{P}_k(t)\rfloor^-\right\}, & \text{if } z_k(t) = 3 \\ \overline{P}_k(t), & \text{otherwise.} \end{cases} \tag{6}$$

where the upper branch indicates the case that the task of device k is being processed at the BS, and its remaining fraction of the task is reduced by the normalized completed part. In addition, once a task of device k is completed by the BS in slot t, its corresponding priority is reset to zero, indicating that the release of the computing resources, which is denoted by

$$P_k(t+1) = \begin{cases} 0, & \text{if } \hat{P}_k(t) \in \mathbf{Z} \text{ and } z_k(t) = 3 \\ \hat{Q}_k, & \text{otherwise} \end{cases} \tag{7}$$

After the BS completes a task, the output data is required to be transmitted back to the device and we denote these time slots as set $\mathcal{T}_k^2 = \left\{ t | \hat{P}_k(t) \in \mathbf{Z},\ z_k(t) = 3 \right\}$. Since the output data size is normally much smaller than the input data size, the transmission of the output data for all tasks is assumed to take only one time slot [14].

H. Scheduling Epochs

One of the our objective is to minimize the average delay of each task. Since the delay from a task's arrival to its scheduling cannot controlled directly, we only take into account the delay from the task scheduling to its completion (when locally computed) or the receiving of the output data (when offloaded to the BS). Therefore, the scheduling epoch of the current task of device k in slot t is denoted by $D_k(t)$ and it evolves as

$$D_k(t+1) = \begin{cases} t+1, & \text{if } t \in \mathcal{E}_k \\ D_k(t), & \text{otherwise} \end{cases} \tag{8}$$

where the upper branch indicates the case that a new task of device k is scheduled at the end of slot t and therefore the index of the next time slot is recorded in the task scheduling epoch memory.

III. Problem formulation

In order to capture the highly dynamic of the tasks' states at different devices, we formulate the computing offloading problem as an MDP-based framework which aims at maximizing the system reward. Basically an MDP problem can be cast into four elements, i.e., the state space $\mathcal{S}$, the decision space $\mathcal{A}$, the state transition probabilities function $P := \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, and the reward function $R := \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

The system state is described by a tuple, $\boldsymbol{s}(t) = (\boldsymbol{z}(t), \boldsymbol{B}(t), \boldsymbol{Q}(t), \boldsymbol{P}(t), \boldsymbol{D}(t), a(t), b(t))$, where $\boldsymbol{B}(t), \boldsymbol{Q}(t)$, $\boldsymbol{P}(t)$ and $\boldsymbol{D}(t)$ are the vectors respectively representing the values of the buffer size, transmission queue size, edge computing priority and the scheduling epochs of all K devices. Note that the state $\boldsymbol{s}(t)$ just captures their values at the beginning of slot t.

A decision is made only in a decision epoch, i.e., $t \in \mathcal{E} = \bigcup_k \mathcal{E}_k$, and is denoted as $\boldsymbol{A}(t) = (A_1(t), A_2(t), ..., A_K(t))$. Its feasible space is related to the current state and is calculated as

$$\mathcal{A}(t) = \prod^{\otimes} \mathcal{A}_k(t) \tag{9}$$

where $\prod^{\otimes}$ is the cumulative Cartesian products of each task's feasible action space $\mathcal{A}_k(t)$. And $\mathcal{A}_k(t) = \mathbf{1}(t \in \mathcal{T}_k^{\mathrm{A}})\{0,1\} \bigcup \mathbf{1}(t \notin \mathcal{T}_k^{\mathrm{A}})\{0\}$, which indicates that when it is a decision epoch for device k, it can choose $A_k(t) \in \{0,1\}$ and when it is not a decision epoch, it can only use the defaulted value, i.e., $A_k(t) \in \{0\}$.

The state transitions include three aspects: The first one is the update of $\boldsymbol{B}(t)$, $\boldsymbol{Q}(t)$, $\boldsymbol{P}(t)$, and $\boldsymbol{D}(t)$, discussed in the preceding section; The second one is updating the channel access process and the transition probability $P(a_{t+1}, b_{t+1} | a_t, b_t)$ decided by the external environment; The third part is the transitions of the task status and is given by

$$z_k(t+1) = \begin{cases} A_k(t) & \text{if } t \in \mathcal{E}_k \\ 2 & \text{if } z_k(t) = 1 \text{ and } t \in \mathcal{T}_k^1 \text{ and } k \neq \arg\max_{k \in \mathcal{K}} P_k(t+1) \\ 3 & \text{if } z_k(t) = 1 \text{ and } t \in \mathcal{T}_k^1 \text{ and } k = \arg\max_{k \in \mathcal{K}} P_k(t+1) \\ 3 & \text{if } z_k(t) = 2 \text{ and } t \in \mathcal{T}^2 \text{ and } k = \arg\max_{k \in \mathcal{K}} P_k(t+1) \\ 4 & \text{if } z_k(t) = 3 \text{ and } t \in \mathcal{T}_k^2 \\ z_k(t) & \text{otherwise.} \end{cases} \tag{10}$$

The first case in (10) indicates that the task status transits into the local computing status (if $A_k(t) = 0$) or the transmission status (if $A_k(t) = 1$) at the beginning of slot $t+1$ after deciding where to process the next task; The second case denotes that the task status transits from transmission status into the edge computing queue waiting status when there are other tasks waiting in the edge computing queue; The third case indicates that the task directly transits into the edge computing status after the transmission because the edge computing queue is empty; The forth case indicates that the task status transits from edge computing queue waiting status to the edge computing status when other tasks are completed (i.e., $t \in \mathcal{T}^2 = \bigcup_{k \in \mathcal{K}} \mathcal{T}_k^2$), and has the highest queue priority (i.e., $k = \arg\max_{k \in \mathcal{K}} P_k(t+1)$); The fifth case indicates that the completed task of device k begins to wait at the BS for the channel availability for transmitting its output data back to the device k.

After each task is completed, a reward is achieved for the network, and the delay [1] is expected to be low. Therefore the reward function is defined as

$$R(\boldsymbol{s}(t)) = \sum_{k \in \mathcal{K}} \left[\eta R_k + (1-\eta)(t - Q_k^{\mathrm{D}}(t)) \right] \mathbf{1}(t_k \in \mathcal{T}_k^0) \tag{11}$$

where weight η is a normalized parameter to adjust the importance preference between the task reward and the delay. Our objective is to maximize the cumulative discounted reward by choosing the optimal action in each time slot based on the given state. Therefore, the problem is formulated as

$$\max_{\boldsymbol{A}_0} \sum_{t=0}^{\infty} \gamma^t R(\boldsymbol{s}_t, \boldsymbol{A}_t) \tag{12}$$

where γ is the discounted factor, and for brevity, the time

[1] The delay is defined as the duration from the instant that a task arrives at the task queue till the instant that the task processing is completed and received by the device.

index t is written as the subscript of the state and action in the following context.

If necessary network information was known as a priori knowledge, the computing offloading problem can be formulated as an MDP with four complete elements and be solved via traditional value iteration algorithms or policy iteration algorithms [15]. Although we can describe the temporal interactions among these states, the stochastic task arrival process at each device and the uncontrollable random access from Wi-Fi devices to the unlicensed channel make the overall state transition probabilities inaccessible. Therefore, we leverage the reinforcement learning algorithm to solve this problem with partially known network information.

Q-learning is one of the most common model-free method of reinforcement learning which can learn the optimal policy via successively interacting with the environment and updating its knowledge from the reward feedback [16]. The basic idea is to improve the evaluations of the quality of particular actions at specific states based on the Bellman equation, given by,

$$Q(\boldsymbol{s}_t, \boldsymbol{A}_t) = (1-\beta)Q(\boldsymbol{s}_t, \boldsymbol{a}_t) + \beta \left(R(\boldsymbol{s}_t, \boldsymbol{A}_t) + \gamma \max_{\boldsymbol{A}} Q(\boldsymbol{s}_{t+1}, \boldsymbol{A}) \right) \quad (13)$$

where β is the learning rate controling the learning speed and accuracy. However, the traditional tabular Q-learning has the curse of dimension problem, resulting in the large memory requirement which increases exponentially with the state and action dimensions. To overcome this high space complexity, we adopt a deep Q-learning method [17] to approximate the corresponding Q values of state-action pairs, i.e.,

$$Q(\boldsymbol{s}, \boldsymbol{A}) \approx \hat{Q}(\boldsymbol{s}, \boldsymbol{A}; \theta) \quad (14)$$

where $\hat{Q}(\cdot; \theta)$ is the deep learning network function with structure and parameter denoted by θ. The detailed algorithm is invoked in Algorithm 1.

The proposed algorithm is executed by the LTE-U BS and we assume it can collect the system states via the reporting scheme which cost little transmission resources. There are two deep learning networks with parameter θ and θ', respectively. The first one is used to select action by comparing their Q values. The second one is used to calculate the target Q values y_j via the Bellman equation and these two separated deep learning networks can reduce the oscillations or divergence of the policy. Variable ϵ is set to balance the exploitation and exploration for the DQL method. When the system reaches a decision epoch, it selects the random policy with probability ϵ, or selects the policy that maximizes the corresponding prediction Q value with parameter θ . After the replay memory is updated, the deep learning neural network for approximation is also updated based on the gradient decent of the random sampled minibatch, aiming at minimizing the square error between the approximated values and the updated values from the Bellman function.

Algorithm 1 DQL algorithm for computing offloading

Initialize:
Set replay memory D to capacity N;
Randomly set action-value function Q with weight θ;
Set action-value function $\hat{Q}$ with weight $\theta' = \theta$;
Initialize state $\boldsymbol{s}_0$;
while 1 **do**
  if $t \in \bigcup_k \mathcal{T}_k^{\mathrm{A}}$ **then**
    With probability ϵ to select a random action $\boldsymbol{A}_t$, otherwise select $\boldsymbol{A}_t = \arg\max_{\boldsymbol{A}} Q(\boldsymbol{s}_t, \boldsymbol{A}; \theta)$;
    Execute the action and observe the reward the next state $\boldsymbol{s}_{t+1}$;
  else
    Set $\boldsymbol{A}_t = \mathbf{0}$;
  end if
  Store the transition $(\boldsymbol{s}_t, \boldsymbol{A}_t, R_t, \boldsymbol{s}_{t+1})$ in D;
  Sample random minibatch of transitions$(\boldsymbol{s}_j, \boldsymbol{A}_j, R_j, \boldsymbol{s}_{j+1})$ from D;
  Set $y_j = R_j + \gamma \max_{\boldsymbol{A}'}(\hat{Q}(\boldsymbol{s}_{j+1}, \boldsymbol{A}'; \theta'))$;
  Perform a gradient decent step on $|y_j - Q(\boldsymbol{s}_j, \boldsymbol{A}_j; \theta)|^2$ with respect to θ ;
  $t = t + 1$;
  Every C steps set $\theta' = \theta$;
end while

IV. Performance Evaluation

In this section, simulation results are presented to investigate the performance of the proposed computing offloading scheme, which is also compared with the benchmark. We consider an LTE-U network consisting of an BS with the coverage radius of 100m. There are 10 randomly distributed IoT devices associated with the BS. The LTE-U network is operated on the band of 5GHz and the distance-dependant pathloss model is given by [18]

$$\mathrm{PL}(R) = 38.46 \log_{10}(R) + 20 \log_{10}(R) + 0.7R \quad (15)$$

and the fast fading gain is also considered, which follows an exponential distribution with unit mean. The noise power spectrum density is -174dBm/Hz. The dedicated bandwidth for the computing offloading service is 5MHz and is equally allocated to all devices. Meanwhile, there are also 10 Wi-Fi STAs are randomly located in the network, employing the IEEE 802.11ac DCF protocol for channel access. The task input data size of different devices (L_k) follows a uniform distribution from 15 Kbits to 20 Kbits. The workloads for all devices are set as 200 cycles/bit. The length of each slot is 1ms. The reward of each device's completed task is a random integer between 1 and 3. We assume that the average task arrival rates at all devices are 0.18 task/s. The defaulted weight η is set as 0.9. We apply a four-layer deep learning network with (48, 48, 24, 24) neuro units, to approximate the Q values. The size of the minibatch for the stochastic gradient decent in the deep learning networks is set as 32. All the simulation results are performed for 76800 time slots.

We first compare our algorithm with the random policy which indicates that a new task is either executed locally or offloaded to the BS with probability 0.5. Fig. 2 shows the performance comparison between the DQL with different learning rate α (which is the learning rates in the deep learning network, instead of that in Eq. (13)) and the random policy. It

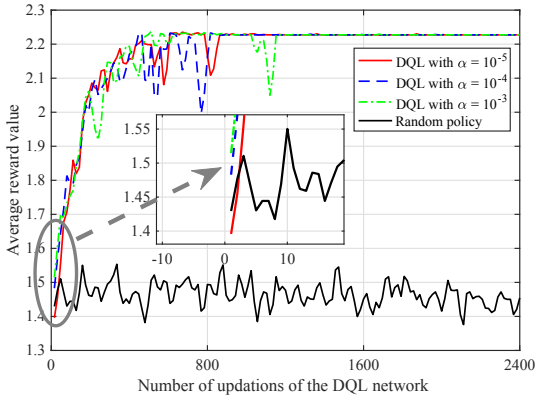

Fig. 2. The convergence process of DQL.

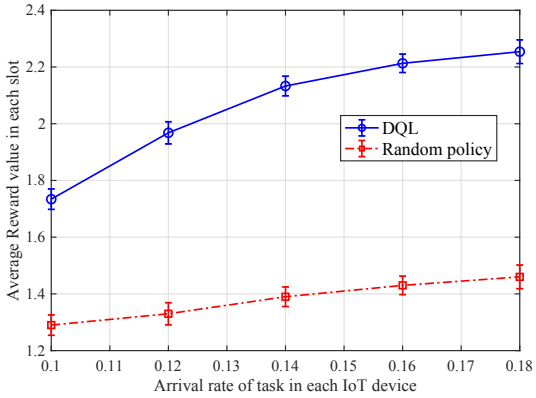

Fig. 3. The reward comparison between DQL and Random policy versus task arrival rate.

can be seen that all the DQL algorithms can nearly converge to a much higher reward level than the random policy, because they exploit the channel status and task status to decide to whether perform the task locally or offload it to the BS. The impact of different learning rates is also shown on the reward curve. When the learning rate is high (e.g., 10^{-3}), it has better performance than that with small learning rates in the initial training phase. However, high learning rate degrades the stability of the learning process even in the converging state, which is also a challenging problem in DQL.

In Fig. 3, given different task arrival rates, we compare the performance of the proposed algorithm with the random policy with a 95% confidence interval. It is found that our proposed algorithm always outperforms the benchmark with a varying task arrival rate. The overall performance of our proposed algorithm increases with the task arrival rate due to a higher probability that the computing resources are occupied by the devices' tasks.

V. Conclusion

In this paper, we propose a Q-learning-based method to solve the computing offloading problem for the LTE-U-enabled IoT networks. Both the task completion reward and the task computing delay are considered in this framework. To resolve the exponential space complexity due to the high dimensions of the states, we utilize a deep learning method to approximate the Q values of different state-action pairs, instead of the traditional tabular method. Simulation results demonstrate that our proposed algorithm considerably improves the system performance.

References

[1] Q. Ye and W. Zhuang, "Distributed and adaptive medium access control for Internet-of-Things-enabled mobile networks," *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 446–460, Apr. 2017.

[2] K. Zhang, J. Ni, K. Yang, X. Liang, J. Ren, and X. S. Shen, "Security and privacy in smart city applications: Challenges and solutions," *IEEE Commun. Mag.*, vol. 55, no. 1, pp. 122–129, Jan. 2017.

[3] Q. Ye and W. Zhuang, "Token-based adaptive MAC for a two-hop Internet-of-Things enabled MANET," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1739–1753, Oct. 2017.

[4] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the edge: A scalable IoT architecture based on transparent computing," *IEEE Network*, vol. 31, no. 5, pp. 96–105, Aug. 2017.

[5] X. Chen, Q. Shi, L. Yang, and J. Xu, "Thriftyedge: Resource-efficient edge computing for intelligent IoT applications," *IEEE Network*, vol. 32, no. 1, pp. 61–65, Jan. 2018.

[6] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, "Long-range communications in unlicensed bands: The rising stars in the IoT and smart city scenarios," *IEEE Wireless Commun.*, vol. 23, no. 5, pp. 60–67, 2016.

[7] W. Yang, M. Wang, J. Zhang, J. Zou, M. Hua, T. Xia, and X. You, "Narrowband wireless access for low-power massive Internet of Things: A bandwidth perspective," *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 138–145, Jun. 2017.

[8] B. P. Rimal, D. P. Van, and M. Maier, "Mobile edge computing empowered fiber-wireless access networks in the 5G era," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 192–200, Feb. 2017.

[9] X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, G. B. Giannakis, and A. Paulraj, "Optimal schedule of mobile edge computing for internet of things using partial information," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2606–2615, Nov 2017.

[10] A. M. Haubenwaller and K. Vandikas, "Computations on the edge in the Internet of things," *Procedia Computer Science*, vol. 52, pp. 29 – 34, May 2015.

[11] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Aug. 2017.

[12] H. He, H. Shan, A. Huang, L. X. Cai, and T. Q. S. Quek, "Proportional fairness-based resource allocation for LTE-U coexisting with Wi-Fi," *IEEE Access*, vol. 5, pp. 4720–4731, Apr. 2017.

[13] 3GPP TR 36.889, "Study on licensed-assisted access to unlicensed spectrum," May 2015.

[14] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[15] H. C. Tijms, *A first course in stochastic models*. John Wiley and sons, 2003.

[16] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, Feb. 2015.

[18] F. Liu, E. Bala, E. Erkip, M. C. Beluri, and R. Yang, "Small-cell traffic balancing over licensed and unlicensed bands," *IEEE Trans. Veh. Technol.*, vol. 64, no. 12, pp. 5850–5865, Dec. 2015.