A Q-learning based Method for Energy-Efficient Computation Offloading in Mobile Edge Computing

Kai Jiang*, Huan Zhou*, Dawei Li†, Xuxun Liu‡, Shouzhi Xu*
*College of Computer and Information Technology, China Three Gorges University, Yichang, China
†Department of Computer Science, Montclair State University, Montclair, NJ, USA
‡College of Electronic and Information Engineering, South China University of Technology, Guangzhou, China
jiangkai0112@gmail.com, zhouhuan117@gmail.com, dawei.li@montclair.edu, liuxuxun@scut.edu.cn, xsz@ctgu.edu.cn

***Abstract*—Mobile Edge Computing (MEC) has emerged as a promising computing paradigm in 5G networks, which can empower User Equipments (UEs) with computation and energy resources offered by migrating workloads from the UEs to the MEC servers. Although the issues of computation offloading and resource allocation in MEC have been studied with different optimization objectives, they mainly investigate quasi-static system environments, without considering the different resource requirements and time-varying system conditions in a dynamic system. In this paper, we exploit a multi-user MEC system, and investigate the task execution scheme for dynamic joint optimization of offloading decision and resource assignment. Our objective is to minimize the energy consumption of all UEs, with considering the delay constraint as well as the dynamic resource requirements of heterogeneous computation tasks. Accordingly, we formulate the problem as a mixed integer non-linear programming problem (MINLP), and propose a value iteration based Reinforcement Learning (RL) approach, named Q-Learning, to obtain the optimal policy of computation offloading and resource allocation. Simulation results demonstrate that the proposed approach can significantly decrease UEs' energy consumption in different scenarios, compared with other baseline methods.**

***Index Terms*—Mobile Edge Computing; Computation Offloading; Resource Allocation; Energy Consumptions; Q-Learning.**

I. Introduction

WITH the advent of many new wireless services in the emerging 5G networks, mobile applications, especially more and more computation-intensive tasks such as online interactive game, face recognition, and augmented/virtual reality (AR/VR), have led to an explosive growth of data traffic [1] [2]. In general, those emerging applications always have intense requirements on Quality of Service (QoS) and sensitive latency, which brings higher energy consumption than traditional applications. However, considering the User Equipments' (UEs') physical size and cost constraints, current UEs have suffered from the limitation of computation resources and energy power, which may become a new bottleneck to address the accompanying challenges in processing extensive applications or providing persistent energy [3] [4].

In order to relieve the conflict between application requests and resource-constrained UEs, Mobile Cloud Computing (MCC) has been emerged as an effective approach considering the cloud server is deployed with higher capacity of computation and storage than the UEs [5]. The MCC technology can enable convenient access to a shared resource pool in the centralized cloud, which empowers UEs with storage, computation and energy resources offered by migrating workloads from the UEs to the cloud servers. Nevertheless, MCC faces the inevitable problem that the cloud servers are spatially far from UEs, which may contribute to extra transmission cost for transmitting data from UEs to the cloud server. In addition, long-distance transmission cannot guarantee the QoS of latency-sensitive applications [6] [7]. Meanwhile, although wireless spectrum resource has been sufficiently utilized with the techniques such as Ultra Dense Networks (UDN) and Dynamic Spectrum Access (DSA) in MCC [8], the big data traffic brought by computation-intensive tasks increases exponentially and there often lacks of a reasonable resource assignment scheme to deal with the tremendous service traffic.

Therefore, Mobile Edge Computing (MEC) is introduced to bring partial network functions to the network edges. MEC emerges as an important component to cope with the computation-intensive tasks in the 5G architecture [9], it empowers MCC by extending cloud computing services from the centralized cloud to the edges of networks. Specifically, MEC enables UEs to migrate workloads to nearby MEC servers by leveraging Base Stations (BS) or Access Points (AP), along with improving the QoS of mobile applications with considerably reduced latency and power consumption [10].

In recent years, the study of computation offloading and resource allocation in MEC has received great attention in both academia and industry, and the optimization objectives are always designed as reducing latency [11] [12], reducing energy consumption [13] [14] or balancing a tradeoff between energy and latency [15] [16]. In practice, however, the above studies are not very suitable for the dynamic systems, due to the fact that most of the previous studies only focus on the performance in quasi-static system, and seldomly consider the sensitive latency property and time-varying system conditions in the time domain. In addition, the impact of different resource requirements and the limited resource capacity in the MEC system have usually been ignored. Accordingly, in order to solve the limitations of existing studies, emerging reinforcement learning has been regarded as an efficient solution to the problems in MEC. Without any prior knowledge of the environment information, Li et al. [17] achieved a RL based adaptive framework to tackle the computation offloading decision in MEC, that framework was also used in [18]

978-1-7281-6607-0/20/$31.00 ©2020 IEEE

to study the dependency-aware task offloading. Furthermore, in [19], a DRL based approach was designed in a three-layer framework in Internet of Vehicles to minimize the energy consumption. Specifically, they considered the delay constraint of users and emulated based on real-world traces of taxis.

In this paper, considering the practical computation offloading and resource allocation property in time-varying MEC system, we focus on jointly optimizing the offloading decision and resource assignment for task execution in MEC, and our goal is to minimize the energy consumptions of all UEs. We formulate the corresponding problem as a mixed integer non-linear programming problem (MINLP), and propose a value iteration based Reinforcement Learning (RL) approach, named Q-Learning, to determine the optimal scheme of joint computation offloading and resource allocation. The key contributions can summarize as follows:

1) We investigate the issue of joint optimization of computation offloading and resource allocation in MEC, and formulate the corresponding problem as a mixed integer non-linear programming problem (MINLP) for minimizing the energy consumption of all UEs.
2) To address the formulated energy consumption minimization problem, we define the state space, action space and reward function, and introduce a Markov decision process for our proposed solution. Then, we propose a value iteration based Reinforcement Learning (RL) approach named Q-learning.
3) Simulation results demonstrate that our proposed method greatly outperforms other baselines in terms of energy consumption in different scenarios.

The remainder of this paper is organized as follows. In Section II, we describe the MEC system model, including the network architecture, communication model, computing model and the energy comsumption of UEs. In Section III, we formulate the problem as a mixed integer non-linear programming problem (MINLP) with the objective to minimize the energy consumption of all UEs. In Section IV, we propose the strategy of offloading and resource allocation, i.e., a Q-learning based method. Furthermore, we analyze the simulation results in Section V. Finally, Section VI concludes this paper.

II. System Model

In this section, the system model of the multi-user MEC network is presented. We first introduce the network architecture adopted in this paper. Then, we give the detail of the analytical model, including the communication model, computation model and the energy consumption.

A. Network Architecture

In the MEC network as shown in Fig. 1, we consider a single-cell scenario, which consists of one Access Point (AP) and n UEs (denoted as a set $\mathcal{I} = \{1, 2, \dots, n\}$). In order to provide the UEs with MEC services, a MEC server is deployed at the AP for computation offloading, and multiple UEs can decide to offload their workloads to the MEC server via the wireless link. Furthermore, we assume that the system operates in fixed length of time slots $t = \{0, 1, 2, \dots, T\}$, each UE has a computation-intensive task which needs to be processed during a computation offloading period t. Meanwhile, the computation task is conceived as atomic and is unable to be divided into partitions, which means that the computation tasks of the UEs cannot be executed on different devices, they can either be executed locally by UE's computation resources, or be offloaded to the MEC server in the AP. When multiple tasks need to be offloaded concurrently, the manager in the MEC server determines how to allocate the spectrum and computation resource to each UE, based on the time-varying system conditions, the heterogeneity of tasks, and the energy overhead of all UEs.

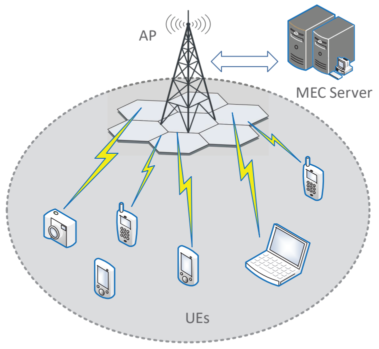

Fig. 1. Multi-User Mobile Edge Network Model.

Without loss of generality, we adopt a widely used task model to describe the arrived tasks. For any computation task on UE_i, it can be denoted as $H_i \triangleq \{s_i, c_i, D_i^{\max}\}$, where s_i stands for the size of computation task H_i, c_i reflects the amount of computation required to accomplish the task H_i. For example, c_i can be quantified as the total number of CPU cycles required to process the task H_i. The variable c_i and s_i are independently and identically distributed over the slots and may have an arbitrary probability distribution which is unnecessary to know. D_i^{max} denotes the maximum time delay tolerance of the task H_i, which means that the task execution time should not exceed D_i^{max}, whether the task is executed either locally or by computation offloading.

Moreover, we assume that the UEs remain covered by the AP during a computation offloading period. In this paper, we only concentrate on executing locally or offloading the task to the MEC serve connected to the AP, without further offloading to the remote cloud or other macro base stations. Let the integer variable $x_i \in \{0, 1\} \quad (\forall i \in \mathcal{I})$ represent the offloading decision of UE_i in time slot t, where $x_i = 0$ means that task H_i is decided to be executed at UE_i's CPU locally, and $x_i = 1$ indicates UE_i decides to offload its task to the MEC server. Consequently, we can define the offloading decision vector of our MEC system as $\boldsymbol{\eta} = \{x_1, x_2, x_3, \dots, x_n\}$.

B. Communication Model

When the computation task is hard to be executed locally under restrained conditions, the UE can migrate the task to the MEC server by a wireless cellular link. In this paper, we consider the UEs communicate with the AP using orthogonal spectrum, and we ignore the communication overhead between the MEC server and AP. Since there is only one AP in a cell, overlapping coverage between neighboring cells is also not considered, the interference among UEs can be neglected. Now we assume that if multiple UEs decide to upload the tasks to the AP simultaneously, the MEC system can allocate bandwidth based on UEs' demands by using Dynamic Spectrum Access (DSA). We define $\theta_i \in [0,1]$ as the percentage of the spectrum allocated to the UE_i. Then, the uplink transmission rate of UE_i served by AP, R_i can be calculated as:

$$R_i = \theta_i W \log_2\left(1 + \frac{p_i \cdot g_i}{\sigma}\right) \tag{1}$$

where W denotes the channel bandwidth of the available spectrum between UE_i and the AP. p_i is the transmission power of UE_i for uploading tasks, g_i is the channel gain of wireless propagation channel and σ is the power of intricate white Gaussian noise.

C. Computation Model

The computation task H_i can be executed either locally on UE_i's own computation resources or on a MEC server via computation offloading.

1) Local Execution Model: For $x_i = 0$, the task H_i will be processed locally on the UE_i. We denote f_i^{loc} and δ_i^{loc} as UE_i's computation capability (in CPU cycles per second) and the consumed energy per CPU cycle of task H_i, respectively. Thus, the computation processing time of task H_i in such case can be given as:

$$T_i^{loc} = \frac{c_i}{f_i^{loc}}, \quad \forall i \in \mathcal{I} \tag{2}$$

and the corresponding energy consumption of UE_i is calculated as:

$$E_i^{loc}(t) = c_i \delta_i^{loc}, \quad \forall i \in \mathcal{I} \tag{3}$$

where $\delta_i^{loc} = 10^{-27}(f_i^{loc})^2$, hinging on the practical chip architecture. Here we refer this coefficient from measurement in [12].

2) Mobile Edge Execution Model: For $x_i = 1$, UE_i chooses to offload the computation task H_i to the MEC server connected to the AP, and then the task will be computed on the MEC server. After handling the computation task, MEC server returns the computing results back to UE. Note that we neglect the transmission time and energy consumption for sending back the result in this paper, due to the tiny amount of outcome data and the higher downlink rate from AP to UE in most instances. Therefore, the total processing time of the task H_i mainly contains two parts, the first one is the time for transmitting H_i from the UE to the MEC server via the wireless access, and the second part is the computation execution time of task H_i on MEC server.

The corresponding energy consumption for transmitting H_i from the UE_i to the MEC server is associated with the transmitting delay directly, so we have:

$$E_i^{trans}(t) = p_i \cdot t_i^{comm} = p_i \frac{s_i}{R_i} \tag{4}$$

We define β_i as the percentage of computation resource assigned to complete task H_i by MEC server, and define f_{mec} as the entire computation resource of the MEC server, so $\beta_i f_{mec}$ represents the computation resource allocated to UE_i. When a higher proportion of computation resources are allocated to a UE, the task execution time becomes smaller, but the energy consumption may increase. Meanwhile, the variable β_i must satisfy the constraint of $\sum_{i=1}^{l} x_i \beta_i \leq 1$. When the MEC server executes the computation task for UE_i, UE_i is supposed to wait the return outcome of the finished task. During this period, we assume that UE_i operates in standby mode and we defined the power consumption in standby state is p_i^w. Thus, the corresponding energy consumption of UE_i is:

$$E_i^{com}(t) = p_i^w \cdot t_i^{comp} = \frac{p_i^w c_i}{\beta_i f_{mec}} \tag{5}$$

Accordingly, combined with the above calculating process, the total execution time and the energy consumption of the task of UE_i in computation offloading consist of the communication process and the computation process, which is easily formulated as:

$$T_i^{offl} = t_i^{comm} + t_i^{comp} = \frac{s_i}{R_i} + \frac{c_i}{\beta_i f_{mec}} \tag{6}$$

and

$$E_i^{offl}(t) = E_i^{trans}(t) + E_i^{com}(t) = \frac{p_i s_i}{R_i} + \frac{p_i^w c_i}{\beta_i f_{mec}} \tag{7}$$

D. Energy Consumption

In the MEC system, UE_i is supposed to decide one computation approach to execute the task H_i, so the delay for each UE_i is:

$$\begin{aligned} T_i &= (1 - x_i)\, \mathrm{T}_i^{loc} + x_i T_i^{offl}, \ \forall i \in \mathcal{I} \\ &= (1 - x_i)\, \frac{c_i}{f_i^{loc}} + x_i \left(\frac{s_i}{R_i} + \frac{c_i}{\beta_i f_{mec}}\right) \end{aligned} \tag{8}$$

Similarly, in order to accomplish the task H_i, the energy consumption of an individual UE_i can be calculated as:

$$\begin{aligned} E_i(t) &= (1 - x_i)\, E_i^{loc}(t) + x_i E_i^{offl}(t), \ \forall i \in \mathcal{I} \\ &= (1 - x_i)\, c_i \delta_i^{loc} + x_i \frac{p_i s_i}{R_i} + \frac{p_i^w c_i}{\beta_i f_{mec}} \end{aligned} \tag{9}$$

Finally, we can formulate the total energy consumption of all UEs in the MEC system, which is given as:

$$E(t) = \sum_{i=1}^{\mathcal{I}} E_i(t) = \sum_{i=1}^{\mathcal{I}} \left[(1 - x_i)\, E_i^{loc}(t) + x_i E_i^{offl}(t)\right] \tag{10}$$

III. Problem Formulation

In this paper, we investigate the joint optimization of computation offloading and resource allocation in the considered MEC system, and our objective is to minimize the long-term energy consumption of all UEs. Considering the maximum tolerant delay constraints, the corresponding problem can be formulated as:

$$\begin{aligned}
&\min_{x_i,\beta_i,\theta_i} \lim_{t\to\infty} \frac{1}{t}\sum_{t=1}^{T} E(t) && \\
\text{s.t.}\quad & C1: x_i \in \{0,1\}, && \forall i \in \mathcal{I}, \\
& C2: T_i \le D_i^{\max}, && \forall i \in \mathcal{I}, \\
& C3: 0 \le \sum_{i=1}^{I} x_i\beta_i \le 1 && \forall i \in \mathcal{I}, \\
& C4: 0 \le \beta_i \le 1 && \forall i \in \mathcal{I}, \\
& C5: 0 \le \Sigma_i^I x_i\theta_i \le 1 && \forall i \in \mathcal{I}, \\
& C6: 0 \le \theta_i \le 1 && \forall i \in \mathcal{I}.
\end{aligned} \tag{11}$$

Here, C1 denotes that each UE only chooses local execution model or edge execution to process the computation task; C2 guarantees that the execution time of local or offloading computation models cannot exceed the maximum tolerant delay of the task; C3 denotes that the computation resource allocated to all UEs cannot exceed the total amount of computation resource provided by the MEC server; C4 ensures that the allocated computation resource to a singe UE_i is not greater than the total amount of computation resource provided by the MEC server; C5 guarantees that the used spectrum by all UEs should be less than the total available spectrum resource of the AP; C6 ensures that the spectrum used by a singe UE_i cannot exceed the total available spectrum resource of the AP.

To address the above problem in (11), it is necessary to find the optimal results of offloading decision variables $\{x_i|i \in \mathcal{I}\}$, computation resource allocation variables $\{\beta_i|i \in \mathcal{I}\}$ and communication resource allocation variables $\{\theta_i|i \in \mathcal{I}\}$, which can be used to minimize the overall computation energy consumption with the given delay constraint. However, the offloading decision variables x_i are binary variables and the communication resource allocation variables β_i as well as the computation resource allocation variables θ_i are dynamically changing. As a result, the system needs to collect a large amount of network state information, and makes the global decision on offloading operation and resources allocation for each UE based on the network's current state. Therefore, the objective function is a mixed integer non-linear programming problem (MINLP), the feasible set of the problem is not convex and the complexity of this method always increases exponentially with the increase of the number of UEs.

IV. Problem Solving

In this section, we first define the state space, action space, reward function and a Markov decision process for our proposed solution. Second, we introduce the classical Q-learning based method to solve the optimization problem.

A. State, Action and Reward Definition

Three critical elements are identified in the reinforcement learning based method, i.e., state, action and reward, which are defined as:

- ***State Space*** **:** The state of the available computation resources, the available spectrum resources at time slot t is determined by the realization of the states $1-\sum_i^I x_i\theta_i$, $1-\sum_{i=1}^N x_i\beta_i$, where the former is the percentage of the available computational capacity of the MEC server, and the latter is the percentage of the available spectrum resources of the wireless channel at present. The purpose of observing them is to keep the constraint of computation capacity and communication channel capacity. Furthermore, we also need to observe the energy consumption $E(t)$ in each time slot in order to compare whether the optimal state is reached. Hence, the state vector at time slot t can be presented as $\boldsymbol{z_i(t)} = \Big(E(t), 1-\sum_{i=1}^I x_i\theta_i, 1-\sum_{i=1}^I x_i\beta_i\Big)$.
- ***Action Space*** **:** In our proposed MEC system, the MEC server needs to determine the computation task offloading strategy for selecting local execution or edge execution. Besides, it also determines the corresponding percentage of the communication and computation resources that are allocated to UE_i at time slot t. Therefore, the system action consists of three parts: the offloading decision vector of UEs $\boldsymbol{\eta} = \{x_1, x_2, x_3, \ldots, x_n\}$, the computation resource allocation variables $\{\beta_1, \beta_2, \ldots, \beta_n\}$ and the communication resource allocation variables $\{\theta_1, \theta_2, \ldots, \theta_n\}$ at time slot t, respectively. Accordingly, the current action can be given as $\boldsymbol{d_i(t)} = \{x_1, x_2, \ldots, x_n, \theta_1, \theta_2, \ldots, \theta_n, \beta_1, \beta_2, \ldots, \beta_n\}$ with the combination of some possible values of these three parts.
- ***Reward Function*** **:** In general, the immediate network reward function is related to the objective function. In the considered optimization problem, our objective is to obtain the minimum total energy consumption, and the goal of RL is to achieve the maximum reward. Thus, the value of reward needs to be negatively correlated to the value of the total energy consumption. The agent obtains $r(\boldsymbol{z_i(t)}, \boldsymbol{d_i(t)})$ in state $\boldsymbol{z_i(t)}$ when action $\boldsymbol{d_i(t)}$ is performed at time slot t. To minimize the energy consumption of all UEs, we define the immediate reward as normalized $r(\boldsymbol{z_i(t)}, \boldsymbol{d_i(t)}) = -E(z_i(t), d_i(t))$, where $E(z_i(t), d_i(t))$ gives the actual total energy consumption of the current state.

B. Markov Decision Process

Markov Decision Process (MDP) is the basis of the reinforcement learning, and almost all programming problems can generally be described in terms of MDP. In this paper, we approximate the computation offloading optimization problem as a MDP, where the agent continually learns and makes decisions through the repeated interaction with the unknown environment in discrete time steps. Specifically, the agent observes current state of the environment as $z_t \in \mathcal{Z}$, then selects and executes an admissible action $d_t \in \mathcal{D}$ according to a policy π in each time step. The policy π is defined as a mapping from the current state to the corresponding action, for which a specific policy π can guide the next decision action

$d_{t+1} = \pi(z_t, d_t)$ under different current states z_t. After that, the agent will obtain an immediate reward $r_t = r(z_t, d_t)$ and the system is transferred into the next new state.

For the long-term consideration, a state value function $V^\pi(z_t)$ is defined by the expected long-term discounted reward value and a discount fact for agent in the state z_t under policy π, which can be used to indicate the long-term effect by making a policy π in the current state z_t (i.e., measuring the quality of a certain state or an available state-action pair). Thus, the state value function based on any initial state z_0 is defined as follows:

$$V^\pi(z_t) = \mathbb{E}_\pi \left[U_t | z_0 = z_t\right] \quad = \mathbb{E}\left[\sum_{t=0}^{\infty} \varphi^t \cdot r_t | z_0 = z_t\right] \tag{12}$$

where $\mathbb{E}$ denotes the expectation, and $\varphi \in (0,1)$ is the discounting factor indicating the importance of the predicted future rewards.

We now use $z_{t+1} \in \mathcal{Z}$ to represent the next state after executing an action d_t at any current state z_t, and the transition probability from z_t to z_{t+1} is given as $p_{z_t z_{t+1}}(d_t)$. As we have formulated the system environment as a MDP, so the state value function $V^\pi(z_t)$ can be transformed to the temporal difference form according to the Bellman Equation as follows:

$$\begin{aligned} V^\pi(z_t) &= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \varphi^t \cdot r_t | z_0 = z_t\right] \\ &= \pi(z_t) \sum_{z_{t+1} \in Z} p_{z_t z_{t+1}}(d_t)\left[r(z_t, d_t) + \varphi V^\pi(z_{t+1})\right] \end{aligned} \tag{13}$$

According to the process above, the goal of the RL agent is to make the optimal control policy $\pi^*(z_t) = d_t^* \in \mathcal{D}$ for maximizing the expected long-term discounted reward under the current state z_t. Therefore, under the optimal policy $\pi^*(z_t)$, the optimization problem can be converted to a recursive optimal state value function $V^{\pi^*}(z_t)$, which is expressed as:

$$\begin{aligned} V^{\pi^*}(z_t) = \max_\pi &\left[r(z_t, d_t) + \varphi \textstyle\sum_{z_{t+1} \in Z} p_{z_t z_{t+1}}(d_t) V^\pi(z_{t+1})\right] \\ &\text{s.t. constraints in (C1)} - \text{(C6)} \end{aligned} \tag{14}$$

Then, the optimal action d_t^* for state z_t under the policy $\pi^*(z_t)$ is easily obtained as:

$$d_t^* = \underset{d_t}{\operatorname{argmax}}\ V^\pi(z_t, d_t), \forall z_t \in \mathcal{Z} \tag{15}$$

C. Q-Learning Based Solution

Q-learning is an effective model-free RL approach, for which both current environment and the state transition probability are not explicit and even time-varying. As the dynamic environment in our proposed network model, Q-learning algorithm attempts to enable the agent to automatically learn the optimal behaviour separately within a specific context in each time step. It directly approximates the optimal Q-value of each state-action pair instead of modeling the dynamic knowledge of MDP, and then updates the Q value in a maintained dimension Q-table after each interaction. Finally, the corresponding policy is derived by selecting the action with the highest Q value under each state.

We adopt the Q-learning method to address the proposed Markov decision problem. Particularly, the Q-learning method can estimate the optimal action values $Q(z,d)$ of the state and admissible action pairs for each time step, which are stored or updated in a Q-table. Here we define the Q value of any admissible action d_t under the state z_t as state-action Q function, then the expected cumulative return after conducting an action d_t is:

$$Q^\pi(z_t, d_t) = r(z_t, d_t) + \varphi \sum_{z_t \in \mathcal{Z}} p_{z_t z_{t+1}}(d) V^\pi(z_{t+1}) \tag{16}$$

Then, it is easy to obtain the relationship between the optimal state value function $V^{\pi^*}(z_t)$ and the state-action Q function, which can be expressed as:

$$V^{\pi^*}(z_t) = \max_\pi Q^\pi(z_t, d_t) \tag{17}$$

Therefore, we can rewrite Eq. (16) with the following form by incorporating Eq. (16) with Eq. (17):

$$Q^\pi(z_t, d_t) = r(z_t, d_t) + \varphi \sum_{z_t \in Z} p_{z_t z_{t+1}}(d) \max_{d_{t+1}} Q^\pi(z_{t+1}, d_{t+1}) \tag{18}$$

At last, we try to update the state-action function by using the recursive method, and now our goal is to estimate the best Q-value instead of finding the best policy. The iterative formula of Q-value of each-step can be obtained as follows:

$$\begin{aligned} Q(z_t, d_t) = Q(z_t, d_t) + \varepsilon\Big[& r(z_t, d_t) + \varphi \max_{d_t} Q(z_{t+1}, d_{t+1}) \\ & - Q(z_t, d_t)\Big] \end{aligned} \tag{19}$$

where $\varepsilon \in (0,1)$ is the learning rate parameter. The Q-value can definitely converge to the optimal value $Q^{\pi^*}(z_t, d_t)$ when an appropriate ε is designated.

More details of our proposed Q-learning method is given in Algorithm 1. In the multi-user MEC scenario, each UE has no way of knowing the information about others except leveraging repeating observation of system, and the system gradually learns to update actions with the corresponding functions to optimize the configuration of various offloading decisions and dynamic resources allocation. The agent computes for each step and stores value $Q(z_t, d_t)$ in a two-dimension table, in which the Q-value can be considered as a long term cumulative reward. However, it should be noticed that the optimal policy π^* from the Q-table has sometimes the disadvantage of being vulnerable to limited search area, and they heavily depend on the quality and the quantity of the training data. In order to provide the trade-off between exploitation and the exploration in Q table, we can choose the action d_t with ϵ-greedy strategy, where ϵ is a diminishing value to provide exploring

Algorithm 1 Q-learning Based method

```
Input: state space $\mathcal{Z}$, action space $\mathcal{D}$, learning rate $\varepsilon$, discount factor $\varphi$
Output: the $Q$-Values for every state-action pair
1:  Initialize $Q(z, d)$ arbitrarily for $\forall\ z \in \mathcal{Z},\ d \in \mathcal{D}$
2:  for each episode do
3:      Initialize $z_t$
4:      for each steps of episode do
5:          Choose an action $d_t$ from the set of possible actions in the current $z_t$ $\forall t$
6:          Execute action $d_t$, observe the reward $r_t$ and the next state $z_{t+1}$
7:          Update new $Q(z_t, d_t) := Q(z_t, d_t) + \varepsilon[r(z_t, d_t) + \varphi \max_{d_{t+1}} Q(z_{t+1}, d_{t+1}) - Q(z_t, d_t)]$
8:          Let $z_t \leftarrow z_{t+1}$;
9:          Terminate When the expected state $z_{terminal}$ reaches
10:     end for
11: end for
```

in reinforcement learning. The UE chooses the action that maximizes the Q-value with the probability $1-\epsilon$ (exploiting), and the other action with the tiny probability ϵ (exploring), which can be expressed as follows:

$$d_t = \begin{cases} d, \epsilon \\ \text{argmax}_d Q\left(z_t, d\right), 1-\epsilon \end{cases} \tag{20}$$

After the algorithm conducts an action d_t, the environment grants the agent a reward $r(z_t, d_t)$ and the current state z_t will transfer to the next state z_{t+1}. With these information, the corresponding Q value for z_t and d_t in the Q table then can be updated using Eq. (19).

V. Numerical Results

To illustrate the performance of the proposed method for dynamic MEC scenarios, we consider a small cell in radius, where one AP with MEC server is located at the center of a small cell. At each time slot, 5 UEs with computation tasks are randomly distributed within AP's coverage. Similar to the ref. [20], the channel power gain g_i is modeled as $127 + 30 \cdot log(L)$, where L is the distance between UE and MEC. At each time slot, the data size of each computation task s_i (in Mbit) obeys uniform distribution between (12, 16), and the corresponding computating load c_i (in Megacycles) obeys uniform distribution between (2000, 2500). Detailed evaluation parameters are listed in Table I.

Then, we compare the proposed Q-learning based method with two baseline methods. Here, we use "Local First" to denote the method that the UEs try to execute their task locally under the maximum tolerate delay. In contrast, we use "Offloading First" to denote the method that the UEs will offload the task to the MEC server preferentially, the whole communication resource and computational resource of MEC server are allocated equally to each UE in Offloading First method. It is worth noting that the resource requirements of computation tasks are dynamic at each time slot t, a certain UE may be incapable of executing the arrived task locally due to the excessive required computation resources under the limitation of the maximum tolerate delay.

TABLE I
Simulation parameters

Parameter	Definition	Value
p_i	The transmission power of UE_i	1.2W
σ^2	The power of White Gaussian Noise	-100dBm
W	The transmission bandwidth	5MHz
f_i^{loc}	The computation capability of UE_i	0.8GHz
f_{mec}	The computation capability of MEC	6GHz
p_i^w	The standby power of UE_i	0.8W
D_i^{max}	The maximum time delay of task H_i	3s

Fig. 2(a) displays the total energy consumption versus the increase of the number of UEs, where computation capability of the UE and the MEC server are 0.8 GHz and 6 GHz, respectively. It can be observed that the energy consumptions of the three methods all increase accordingly with the increase of the number of UEs. By comparing these three methods, we can find that the proposed Q-learning based method performs best, as its energy consumption is smallest. The energy consumption of the Offloading First method is smaller than the Local First method when the number of UEs is relatively small. Moreover, the energy consumption of the Offloading First method exceeds the Local First method when the number of UE is 4 and continues to grow substantially with the increasing number of UEs. This is because the computation and communication resource budget get relatively tight when more tasks need to be executed at a time slot. Once the available resource allocated to a single UE decreases, the delay of transmission and computation for UE will increase significantly, as well as the energy consumption.

Fig. 2(b) demonstrates the total energy consumption versus different computation capacity f_{mec} of MEC server, where the number of UEs is 5. It can be observed that the energy consumption in all methods decreases as the computation capacity of MEC server increases. Similar to the results in Fig. 2(a), the proposed Q-learning based method still performs best, which means that the proposed method is superior to the other two methods. When the computation capacity of MEC server is small, the Offloading First method has a much higher energy consumption than other two methods, and the difference between the Offloading First method and other methods becomes smaller with the increase of the computation capacity of MEC server. The main reason is that as the computation capacity of MEC server increases, each UE can be allocated with more computation resource, thus the computation delay will decrease significantly, as well as the energy consumption.

Fig. 2(c) shows the energy consumption versus different computing load c_i of tasks (the same for all tasks in a time slot), where the number of UEs is 5. It can be observed that the energy consumption grows with the increase of the computing load by each task for all methods due to the higher delay. Similarly, the proposed Q-learning based method consume

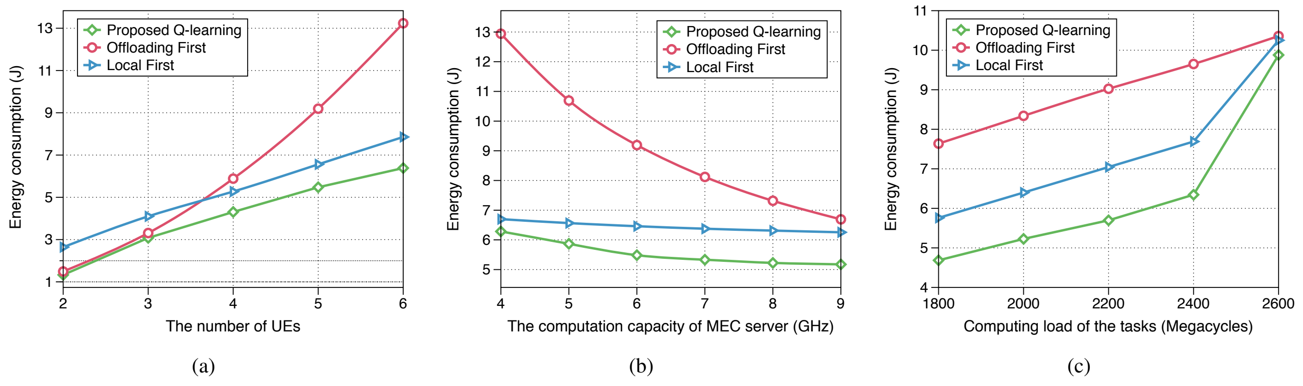

Fig. 2. (a) Energy consumption versus the number of UEs. (b) Energy consumption versus the computation capacity of MEC server. (c) Energy consumption versus the computing load of tasks.

smaller energy than the other two methods. For the Offloading First method, before the local execution time for each UE exceeds the delay threshold, the energy consumption is always much higher than other two methods, which means that the resource for offloading 5 tasks simultaneously in a time slot is very tight under the current system condition. Then, as the computing load by each task reaches 2400 Megacycles, it can be found that there are sudden leaps of the energy consumption in the Local First and proposed Q-learning based methods. This is because the maximum tolerant delay $D_i^{\max}$ cannot be satisfied by computing locally for each UE when c_i exceeds 2400 Megacycles. Hereafter, all tasks in each time slot have to be offloaded to the MEC server in three methods.

VI. Conclusion

In this paper, we have investigated the joint optimization of computation offloading and resource allocation in a multi-user MEC system, to cope with the different resource requirements and time-varying system conditions. Meanwhile, the delay constraint and the limited resource capacity has also been considered. We have formulated the optimization problem as a MINLP, and used a Markov decision process to describe the relationship between the offloading and resource allocation policies and the system environment. Then, we have proposed a Q-learning based method to obtain the optimal strategy. Numerical results have demonstrated the effectiveness of the proposed method under different scenarios.

VII. Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under Grants NO.61872221 and NO.61671209. The corresponding author is X. Liu.

References

[1] T. Taleb et al., "Coping with emerging mobile social media applications through dynamic service function chaining," *IEEE Trans. Wireless Commun.*, vol. 15, no. 4, pp. 2859-2871, Apr. 2016.

[2] X. Chen, L. Jiao, W. Li, X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795-2808, Oct. 2016.

[3] G. Premsankar, M. D. Francesco, T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet of Things J.*, vol. 5, no. 2, pp. 1275-1284, Apr. 2018.

[4] H. Zhou et al., "Predicting Temporal Social Contact Patterns for Data Forwarding in Opportunistic Mobile Networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10372-10383, Nov. 2017.

[5] Y. Chao, Y. Liu, X. Chen and S. L. Xie, "Efficient Mobility-Aware Task Offloading for Vehicular Edge Computing Networks," *IEEE Access*, vol. 7, pp. 26652 - 26664, Feb. 2019.

[6] Z. Sanaei, S. Abolfazli and A. Gani, "Heterogeneity in mobile cloud computing: taxonomy and open challenges," *IEEE Commun. Surv. Tuts.*, vol. 16, no. 1, pp. 369-392, 1st Quart. 2014.

[7] H. Zhou et al., "DRAIM: A Novel Delay-constraint and Reverse Auction-based Incentive Mechanism for WiFi Offloading," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 4, pp. 711-722, 2020.

[8] W. Zhang et al., "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569-4581, Sep. 2014.

[9] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, W. Wang, "A survey on mobile edge networks: convergence of computing caching and communications," *IEEE Access*, vol. 5, pp. 6757 - 6779, Mar. 2017.

[10] N. Abbas et al., "Mobile edge computing: A survey," *IEEE Internet of Things J.*, vol. 5, no. 1, pp. 450-465, Feb. 2018.

[11] K. Zhang, Y. Mao, S. Leng, S. Mahaijan and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 1-6, 2017.

[12] Y. Mao, J. Zhang and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590-3605, Dec. 2016.

[13] H. Guo, J. Liu, J. Zhang, "Efficient Computation Offloading for Multi-Access Edge Computing in 5G HetNets," *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 1-6, Jul. 2018.

[14] O. Munoz et al., "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738-4755, Oct. 2015.

[15] J. Zhang et al., "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet of Things J.*, vol. 5, no. 4, pp. 2633-2645, Aug. 2018.

[16] M. -H. Chen, B. Liang, M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," *Proc. IEEE Int. Conf. Commun.(ICC)*, pp. 1-6, Jul. 2016.

[17] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, pp. 1-6, Apr. 2018.

[18] S. L. Pan, Z. Zhang, D. Zeng, "Dependency-Aware Computation Offloading in Mobile Edge Computing: A Reinforcement Learning Approach," *IEEE Access*, vol. 7, pp. 134742-134753, Sep. 2019.

[19] Z. Ning et al., "Deep Reinforcement Learning for Intelligent Internet of Vehicles: An Energy-Efficient Computational Offloading Scheme," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 4, pp. 1060-1072, 2019.

[20] M. Chen, Y. Hao, "Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587-597, Mar. 2018.