IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 8, 2021, accepted March 25, 2021, date of publication April 5, 2021, date of current version April 13, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3070908

Task Offloading and Trajectory Control for UAV-Assisted Mobile Edge Computing Using Deep Reinforcement Learning

LU ZHANG[1,2,3], **ZI-YAN ZHANG**[1,2,3], **LUO MIN**[5], **CHAO TANG**[1,2,3], **HONG-YING ZHANG**[1,2,3], **YA-HONG WANG**[1,2,3], **AND PENG CAI**[1,2,3,4]

[1]Key Laboratory of Urban Environment and Health, Institute of Urban Environment, Chinese Academy of Sciences, Xiamen 361021, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
[3]Xiamen Key Laboratory of Physical Environment, Xiamen 361021, China
[4]Shanghai Institute of Nutrition and Health, Chinese Academy of Sciences, Shanghai 200031, China
[5]State Key Laboratory of Electronic Thin Films and Integrated Devices, University of Electronic Science and Technology of China, Chengdu 610054, China

Corresponding authors: Zi-Yan Zhang (zyzhang@iue.ac.cn) and Peng Cai (pcai@iue.ac.cn)

ABSTRACT Mobile Edge Computing (MEC) has been widely employed to support various Internet of Things (IoT) and mobile applications. By leveraging the advantages of easily deployed and flexibility of Unmanned Aerial Vehicle (UAV), one of MEC primary functions is employing UAVs equipped with MEC servers to provide computation supports for the offloaded tasks by mobile users in temporally hotspot areas or some emergent scenarios, such as sports game areas or destroyed by natural disaster areas. Despite the numerous advantages of UAV carried with a MEC server, it is restricted by its limited computation resources and sensitive energy consumption. Moreover, due to the complexity of UAV-assisted MEC system, its computational resource optimization and energy consumption optimization cannot be achieved well in traditional optimization methods. Furthermore, the computational cost of the MEC system optimization is often exponentially growing with the increase of the MEC servers and mobile users. Therefore, it is considerably challenging to control the UAV positions and schedule the task offloading ratio. In this paper, we proposed a novel Deep Reinforcement Learning (DRL) method to optimize UAV trajectory controlling and users' offloaded task ratio scheduling and improve the performance of the UAV-assisted MEC system. We maximized the system stability and minimized the energy consumption and computation latency of UAV-assisted the MEC system. The simulation results show that the proposed method outperforms existing work and has better scalability.

INDEX TERMS Computation offloading, deep learning, energy saving, mobile edge computing, reinforcement learning, unmanned aerial vehicle.

I. INTRODUCTION

The increasing popularity of the Internet of Things (IoT) [1] provides a promising platform for sophisticated mobile applications such as automatic driving [2], augmented reality [3], and various cognitive applications. With the rapid growth of IoT and mobile devices, a vast number of devices are connected to wireless networks and rely on online resources to process various tasks. Offloading a large number of tasks through the core networks and processing the tasks on central cloud servers would cause network traffic congestion and increase latency. In addition, most mobile applications are latency-sensitive, computation-intensive, and energy-intensive. To mitigate those issues, Mobile Edge Computing (MEC) [4] is propose to process offloaded tasks in proximity.

Although MEC has numerous advantages, it cannot avoid the limitations of static tower locations. Therefore, it is challenging to deploy MEC anytime and anyplace. In addition, there is a high possibility that the infrastructure might sometimes be destroyed by the natural disasters. Furthermore, it is almost impossible to mount the infrastructure for temporary use or rural areas like hot spots and mountains. In those mentioned scenarios, the IoT devices cannot fully function to serve its users. Thanks to the flexibility of Unmanned Aerial Vehicles (UAV), UAV-assisted MEC [5]–[7] is introduced to serve as a computational server for mobile users at

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenhui Yuan.

flexible positions. The UAV-assisted MEC can extend the mobile devices' operating lifetime and accelerate computation by providing extra computation resources on the MEC servers. Moreover, offloading tasks to the proximity MEC server avoids mobile users frequently communicating with the cloud or uploading their tasks to the cloud, thereby alleviating communication congestion. Due to the limited payload and energy, the UAV has limited computation capability and flying time. Therefore, the UAV-assisted MEC system demands a control model that can maximize the number of tasks processed before expiring and minimize energy consumption.

Conventional optimization methods have been adopted to address abovementioned problems. As presented in [8], task scheduling and trajectory controlling are essential to minimize the energy. Zhang *et al.* [9] adopted the Lagrangian dual method and successive convex approximation (SCA) to optimize task offloading to UAV-assisted MEC server with respect to minimize the energy consumption. The classical optimization methods such as convex (CVX) optimization [10] and mixed-integer optimization (MIP) [11] are also employed to address similar problems. Similarly, Liu *et al.* [12] presented an algorithm based on SCA to minimize the engorge consumption by controlling the CPU frequencies of edge servers and UAV trajectory. However, the UAV-assisted MEC system environments are not fully observable to the optimization models, and it is challenging to formulate the complex MEC environments to the conventional optimization models. Besides, the computational cost of standard optimization methods is growing exponentially with the increasing of parameters.

As one of the most powerful pattern recognition and optimization methods, machine learning [13] has extensively employed to learn and optimize various problems for UAV-assisted MEC. Jiang *et al.* [14] introduced a deep learning method to optimize UAV positions and schedule resources in UAV-assisted MEC to save energy consumption. However, both standard machine learning and deep learning [15], [16] methods require labeled historical data, which demands considerable human effort to label the training data. Reinforcement Learning [17] can learn and optimize for UAV-assisted MEC without training data by interacting with MEC environment. Therefore, reinforcement learning [17] and Deep Reinforcement Learning (DRL) [18] methods are introduced resource allocation and UAV position optimization [19] to minimize energy consumption. DRL [20] employs deep neural networks to capture the complex states of UAV-assisted MEC and reinforcement learning to make decisions. Wang *et al.* [21] presented a Multi-Agent DRL model for trajectory planning and task offloading for UAV-assisted MEC. However, the existing reinforcement learning or DRL methods require further optimization steps to achieve joint optimization purposes. The extra steps introduced additional computational costs and compromised the performance of the models.

This paper aims to develop an end-to-end DRL model to optimize UAV trajectory planning and task offloading for UAV-assisted MEC. Specifically, the model is to jointly optimize computation offloading and UAV trajectory controlling to minimize the time and energy consumption of the whole system and maximize the stability of the system. To maximize stability means to balance the computation of the system workload, extend the system operation time, and maximize the number of completed tasks. The main contributions of this paper are summarized as the following:

- The UAV-assisted MEC system has been formulated Markov Decision Process (MDP) so that we can solve the optimization problem with DRL models. The details of the MDP elements and UAV-assisted MEC environment are well defined with explanations.
- A novel end-to-end DRL mode is developed to address the optimization problem. Specifically, the DRL model is proposed to optimize the offloading task ratio and UAV trajectory to minimize the overall energy consumption in UAV-assisted MEC.
- The proposed method has been verified extensive simulations. The simulation results show the proposed model outperforms the existing methods, including greedy algorithm, reinforcement learning, and DRL methods.

The remainder of the paper is organized as follows. Section II presents the related works, and Section III shows the system model and problem formulation. In Section IV, we explain how to solve the optimization problem in the UAV-assisted MEC system with the proposed DRL model. After that, we present the simulation results and concluded our work in Section V and Section VI, respectively.

II. RELATED WORKS

In the literature, many research works on computational offloading [22] and trajectory controlling [23] in MEC and UAV-assisted MEC. Since MEC can increase computation capacity and save energy for mobile devices, it has been extensively researched in recent years as a critical technology towards 5G. In [24], the authors provide high-quality surveys, where the definition, the computation and communication modeling of MEC, and its advantages and applications are discussed. Chen and Hao [25] formulated the task offloading problem of MEC as a mixed-integer non-linear program to reduce the computation delay and save the battery life. Zhang *et al.* [26] explored the computation offloading mechanisms for MEC in 5G networks, where energy consumption is minimized.

Generally, we can category UAV-assisted MEC literature into several types based on the proposed methods and their research objectives. The first type is to minimize the energy consumption of the whole system or the energy consumption of mobile users [27]–[34]. Hua *et al.* [27] proposed a resource allocation strategy for a single UAV-assisted edge server for an individual mobile user system, where a UAV is designed to fly from the predefined initial location to the final location with the computation power. Instead of only providing the offloaded service for a single mobile user, Jeong *et al.* [28] researched how a single UAV provides the computation service for multiple users. In [27] and [28], the authors explored the UAV-assisted MEC system that keeps the UAV fly and provides offloaded computation service simultaneously. In order to save propulsion energy, Xiong *et al.* [30] and Diao *et al.* [32] proposed that the UAV only provides offloaded computation services in a specific time or a specific positions.

Besides considering the energy consumption of the MEC system, the computation rate is also considerable for UAV assisted MEC system [35]–[37]. There are two kinds of offloading computation in MEC, including binary offloading and partial offloading. In the binary offloading mode, the users only choose to perform computation tasks all in the local or select to offload all computation tasks to the MEC server. For partial offloading mode, the users can perform computation tasks partially in local and MEC server, where the local computation and offloaded computation are performed in parallel. In [36], a two-stage alternative algorithm and a three-stage alternative algorithmare employed to tackle the partial offloading and binary offloading problem respectively. Hu *et al.* [35] researched partial offloading in UAV-assisted MEC system with a penalty dual decomposition-based and *L0* norm algorithm, which minimized the total processing time, including the transmission time, computation time, and local computation time. Of most interesting, the simulation results indicated that a better performance could be achieved while UAV keeps stationary in a set of time intervals to collect data.

The joint optimization methods are extensively investigated in recent literature because energy consumption and computation rate are both overcritical because of the limited onboard energy of the UAV. Chen *et al.* [37] utilize the DRL methods to schedule the offloading to improve the satisfaction of perceived delay and energy consumption of mobile users. Zhan *et al.* [38] first studied the energy minimization without the time constraints, followed by the minimization problem of the task completion time. After that, it jointly optimized the UAV energy and completion time with the Pareto-optimal solution. It is noteworthy that the existing studies, such as the mentioned above, researched the UAV-assisted MEC system with the objective of energy consumption minimizing and task completion time minimizing separately without considering the balance of two aspects. Although Zhan *et al.* [38] take the trade-off between energy consumption and task completion time into account, the authors did not take the long-term stability of the whole system into consideration. Deep Reinforcement Learning (DRL) [18] methods are naturally introduced to make decisions for UAV-assisted MEC [39] to achieve high performance. However, they did not consider the partial offloading, where the computation task can be only processed solely on a local device or the UAV; therefore, it has much less freedom to control and optimize for the offloading tasks.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. SYSTEM MODEL

In this work, we consider a MEC system with a single UAV and K mobile users in a 3-D Cartesian coordinate, as illustrated in Fig. 1. A UAV carries a MEC server and flies at fixed altitude to provide computation service for IoTs and mobile users. Since the MEC server has greater computing capability than mobile users, mobile users can offload their computation-intensive and latency-sensitive tasks to the UAV so that mobile users to reduce their energy costs and speed up computation. The set of mobile users is described by $k \in \mathcal{K} = \{1, 2, \ldots, K\}$. Concretely, a user k can offload d_τ of the current task to the UAV, and $1 - d_\tau$ of the task is processed on the local device. A control agent can plan the trajectory of the UAV and the offloading proportion d_τ of the task.

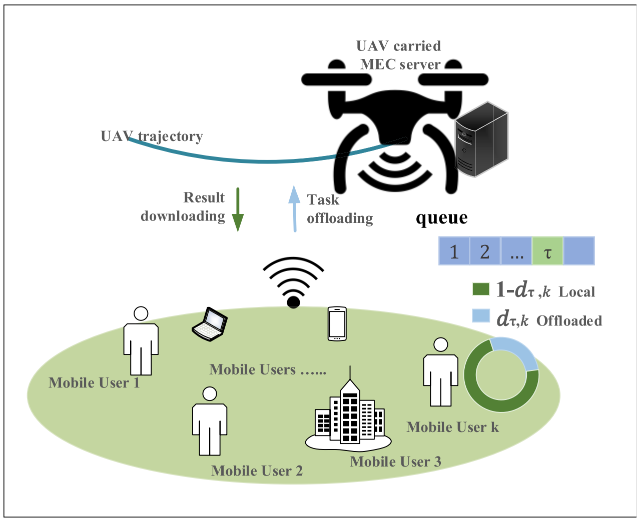


FIGURE 1. System model of UAV-assisted MEC system.

To minimize the total energy consumption of the MEC system and maximize the number of completed tasks over the time T, we need to define models in the MEC system, including communication model, offloaded computation model, and local computation model.

1) COMMUNICATION MODEL

In the communication model, users transmit their computation tasks to the UAV-assisted MEC server. We divide time T into N time slots, where $N > K$ and τ^{th} time slots is defined as $\tau \in \mathcal{T} = \{1, 2, \ldots, N\}$ and the length of time slots is sufficiently small. Assume there is up to 1 task generated in each time slot. The position of mobile user k is given by $u_k = [x_k, y_k, 0], k \in \mathcal{K}$. The trajectory of the UAV in a horizontal plane at altitude H can be indicated by UAV's discrete locations in each time slots, which is defined as $h_\tau = [x_\tau, y_\tau, H], \tau \in \mathcal{T}$. Assume that the UAV has the capability to return to its initial position after completing missions. Thereby, we can derive the following constraints

TABLE 1. Notation Summary.

u_k	the k^{th} online user
f_k	devices CPU frequency of k^{th} user
h_τ	the UAV position at the τ^{th} time step
$\Omega_{\tau,k}$	the τ^{th} tasks from user k
$r_{\tau,k}$	current speed of transfer data from the τ^{th} user to the UAV
$t^{com}_{\tau,k}$	transfer time cost for task $w_{\tau,k}$
$E^{com}_{\tau,k}$	transfer energy consumption for task $w_{i,k}$
f_{UAV}	the CPU frequency of UAV frequency
Q_{UAV}	the current state of task queue in the UAV
$t^{off}_{\tau,k}$	offloading computing time cost
$E^{off}_{\tau,k}$	offloading computing energy cost
a_τ	action has been taken at time slot τ
$R_\tau(s_\tau, a_\tau)$	reward function base on state s_τ and action a_τ

for the UAV's flight as equation (1).

$$\frac{\|h_{\tau+1} - h_\tau\|N}{T} \le v_{max}, \tau \in \mathcal{T}, \tag{1}$$

where equation (1) represents that the speed of UAV has to satisfy the maximum speed constraints of UAV v_{max}.

Since UAV flies at the proximity of mobile users, there are Line-of-Sight (LoS) link and Non-line-of-Sight (NLoS) link between the UAV and mobile users communications. Define P^{LoS} as the LoS link probability between the UAV and mobile users, the calculation of which can be retrieved from [40], given as equation (2). The environment-related variables α and β also can be obtained from paper [40].

$$P^{LoS}_{\tau,k} = \frac{1}{1 + \alpha \exp(-\beta(arctan(\frac{H}{\sqrt{(y_k - y_\tau)^2 + (x_k - x_\tau)^2}}) - \alpha))}, \quad \tau \in \mathcal{T}, k \in \mathcal{K}. \tag{2}$$

Based on [40], the pass-loss between UAV and the mobile user k at τ^{th} time slots is indicated as equation (3).

$$L_{\tau,k} = 20log(\frac{4\pi f_c}{c}\|u_k - h_\tau\|) + P^{LoS}_{\tau,k}\eta_{LoS} + (1 - P^{LoS}_{\tau,k})\eta_{NLoS}, \quad \tau \in \mathcal{T}, k \in \mathcal{K}, \tag{3}$$

where c is the light speed and f_c is the carrier frequency. The parameters η_{LoS} and η_{NLoS} denotes environment-dependent losses for LoS and NLoS links, respectively. Therefore, the data transmit rate of mobile users to UAV is given as (4)

$$r_{\tau,k} = \mathcal{B}log_2(1 + \frac{p_{\tau,k}10^{-L_{\tau,k}/10}}{\sigma^2}), \quad \tau \in \mathcal{T}, \ k \in \mathcal{K}, \tag{4}$$

where $\mathcal{B}$ denotes the bandwidth, $p_{\tau,k}$ denotes the transmission power of mobile user k at τ^{th} time slots, and σ^2 denotes noise power.

Assume that $D_{\tau,k}$ bits data is needed to compute for mobile user k at τ^{th} time slot. $d_{\tau,k}$ is introduced to indicate the ratio of the tasks offloaded via user k to UAV at τ^{th} time slot. Thereby, in communication model, the transmission time and the transmission energy consumption is calculated by equation (5) and (6), respectively.

$$t^{com}_{\tau,k} = \frac{D_{\tau,k}d_{\tau,k}}{r_{\tau,k}}, \tag{5}$$

$$E^{com}_{\tau,k} = p_{\tau,k}\frac{D_{\tau,k}d_{\tau,k}}{r_{\tau,k}}. \tag{6}$$

where $p_{\tau,k}$ is the transmission power.

2) OFFLOADED COMPUTATION MODEL

After transmitting the offloaded computation tasks, UAV and mobile users perform the offloaded computation and local computation. Denote $C_{\tau,k}$ as the required CPU cycles to compute per bit data. Therefore, the computation time $t^{off}_{k,c}$ and energy consumption $E^{off}_{k,c}$ can be calculated by:

$$t^{off}_{\tau,k} = \frac{D_{\tau,k}d_{\tau,k}C_{\tau,k}}{f_{UAV}}, \tag{7}$$

$$E^{off}_{\tau,k} = \kappa D_{\tau,k}d_{\tau,k}C_{\tau,k}f^2_{UAV}, \tag{8}$$

where f_{UAV} represents CPU frequency of MEC server mounted on UAV. $\kappa = 10^{-26}$ is a hardware-related constant. The UAV has an idle state to save energy and a running state to process the tasks.

Since multiple tasks are transmitted to UAV by many mobile users, there is a waiting time for transmitted tasks. Suppose that there is a virtual queue Λ in UAV. The virtual queue holds the offloaded tasks, and the edge server computes the tasks with first-in-first-served order. The proposed model algorithm decides the offloaded ratio of the queue head task and UAV position at next time slot. Suppose that there are $a - 1$ tasks waiting at this virtual queue, then the current task is added into the queue as a^{th} element. Therefore, we can calculate the waiting time of user k as follows:

$$t^{wait}_{\tau,k} = \sum_{(\tau,k)=1}^{a-1} t^{off}_{\tau,k}. \tag{9}$$

In the offloaded computation model, the total time cost is given as equation (10).

$$t^{uav}_{\tau,k} = t^{wait}_{\tau,k} + t^{off}_{\tau,k} = \sum_{(\tau,k)=1}^{a-1} t^{off}_{\tau,k} + t^{off}_{\tau,k}, \quad (\tau,k) = a. \tag{10}$$

3) LOCAL COMPUTATION MODEL

Similar to offloaded computation model, given the frequency of CPU f_k for the mobile user k, we can derive the local computation time and local computation energy consumption in equation (11) and (12).

$$t^{loc}_{\tau,k} = \frac{D_{\tau,k}(1 - d_{\tau,k})C_{\tau,k}}{f_k}, \tag{11}$$

$$E^{loc}_{\tau,k} = \kappa D_{\tau,k}(1 - d_{\tau,k})C_{\tau,k}f^2_k. \tag{12}$$

4) OVERALL TIME AND ENERGY CONSUMPTION

The total time cost is equal to maximum value of the summarize of transmission time and offloaded time and local computation time, as equation (13).

$$t_{\tau,k}^{total} = max(t_{\tau,k}^{com} + t_{\tau,k}^{uav}, t_{\tau,k}^{loc}), \quad t_{\tau,k}^{total} \leq \Delta_{\tau,k}, \tag{13}$$

where $\Delta_{\tau,k}$ denotes maximum tolerant of task $t_{\tau,k}^{total}$. If $t_{\tau,k}^{total}$ exceeds this expiration time, we consider the algorithm is failed to complete this task because the task is expired.

The total of energy consumption of UAV assisted MEC system includes communication energy consumption, offloaded computation energy consumption, local computation energy consumption, and the propulsion energy consumption of the UAV. The former three types of energy consumption have been listed in the above, and the propulsion energy consumption can be given,

$$E_{\tau,UAV}^{pro} = \xi(\frac{\|h_{\tau+1} - h_\tau\|N}{T})^2, \tag{14}$$

where $\xi = 0.5MT/N$ and M is the mass of UAV including its payload [35].

Therefore, the total of energy consumption at τ^{th} time slots, can be computed as,

$$E_{\tau,k} = E_{\tau,k}^{com} + E_{\tau,k}^{off} + E_{\tau,k}^{loc} + E_{\tau,UAV}^{pro}. \tag{15}$$

B. PROBLEM FORMULATION

In the proposed UAV-assisted MEC system, we aim to minimize the total energy consumption and time consumption simultaneously; meanwhile, the number of completed task before their expiration time is maximized. Therefore, we formulate our target problem as follows:

$$\begin{aligned} &max \sum_{\tau=0}^{N} \mathbb{E}[\lambda_1 F_{\tau,k} - \lambda_2 \log E_{\tau,k} - \lambda_3 \log t_{\tau,k}^{total}], \\ &\text{s.t.} \sum_{\tau}^{N} E_{\tau,k} \leq E_{uav} \\ &t_{\tau,k}^{total} \leq \Delta_{\max} \end{aligned} \tag{16}$$

where λ_1, λ_2 and λ_3 are the normalization factors that covert the terms into about the same magnitude values. The total energy consumption is constrained by the total consumable energy E_{uav} in the UAV, and the tolerance of the tasks constrain the computation time. Again, $E_{\tau,k}$ and $t_{\tau,k}^{total}$ are defined as the energy and time cost at τ^{th} time step. And $F_{\tau,k}$ is a flag value describes whether the tasks processed before its expiration time; it can be assigned with value 1 as if the total time cost less than the maximum tolerant and 0 otherwise, which given by:

$$F_{\tau,k} = \begin{cases} 1, & \text{if} t_{\tau,k}^{total} \leq \Delta_{\max}; \\ 0, & \text{otherwise}, \end{cases} \tag{17}$$

where the $t_{\tau,k}^{total}$ is total consumption shown in equation (7).

Note that we can remove the $E_{\tau,k}$ term if we only care about whether the task has been processed before its maximum tolerant time and response speed. However, we may also want to extend UAV life time by sacrificing the response time. Therefore, it is more reasonable to both minimize the time cost and energy consumption.

IV. PROPOSED METHOD

This section presents a Markov Decision Process (MDP) for UAV-assisted MEC and the proposed DRL model to optimize the abovementioned target problem.

A. MARKOV DECISION PROCESS SETTINGS

DRL employs deep neural networks to capture the complex states of UAV-assisted MEC and reinforcement learning [17] to make decisions. Therefore, the DRL settings are the same as reinforcement learning configuration. Specifically, there is a learning agent interacts with environment and learns to accumulate long-term expected rewards. The environment describes the real-world with a Markov Decision Process (MDP). In this work, we consider the MDP has a finite number of states, and terminal states are defined as the MEC server is overloaded and cannot process the newly arrived tasks. In practice, we can consider an episode is terminated when the task waiting time on the MEC server is longer than a threshold. For the sake of simplicity of the argument, we assume that the UAV has remained the sufficient energy to return to its initial state after completed all tasks.

1) STATE SPACE

From the system description, the states are considerably complicated as it contains the states of user devices, task profiles, network channel distribution, and various parameters of UAV. We first define that MDP in each episode is described by a set of states, given by:

$$\mathcal{S} = \{s_1, s_2, \ldots, s_\tau, \ldots, s_N\} \tag{18}$$

where $s_\tau \in \mathcal{S}$ is a general state at time slot τ, and s_τ equal to $\{f_k, f_{UAV}, \Omega_{\tau,k}, r_{\tau,k}, \mathbf{h}_\tau, \Lambda_\tau\}$, where f_k and f_{UAV} are devices CPU frequency of k^{th} user and UAV, respectively; $\Omega_{\tau,k}$ is defined as the information of τ^{th} task on mobile user k, represented by $\Omega_{\tau,k} = \{D_{\tau,k}, C_{\tau,k}, \Delta_{\tau,k}\}$; $r_{\tau,k}$ is current speed of transfer data from mobile user k to the UAV; h_τ is UAV position at time step τ; Λ_τ tasks queue states in the UAV at time slot τ.

2) ACTION SPACE

The states are transited according to the actions that have been taken and the MEC network's internal transition probability. Each action contains two parts, including deciding the percentage of a task to offload and control the next position of UAV. And a general action can be defined as:

$$a_\tau = \{d_{\tau,k}, \mathbf{h}_{\tau+1}\}, \tag{19}$$

where $d_{\tau,k}$ is the percentages of the tasks to offloading at time step τ. In other words, the task is offloaded $d_{\tau,k}$ of the task to the UAV and $1 - d_{\tau,k}$ process on local devices.

The $\mathbf{h}_{\tau+1}$ decides the next position of UAV. Note that the MDP environment state is changed whenever an action has been taken conditional upon the current state.

3) TRANSITION FUNCTION

We assume the channel gain does not change during the short time slot τ. That means that the data transfer speed $r_{i,k}$ change when the UAV has been moved to the new position because $r_{i,k}$ depends on the distance and pass-loss. We do not consider that the channel gain change while UAV is flying. Thus, the transition probability of the MEC network can be given by:

$$p(s'|s,a) \doteq \Pr\{s_\tau = s'|s_{\tau-1} = s, a_{\tau-1} = a\}, \tag{20}$$

where, s and s' are the current state and next state, and the transitions depended are the actions taken by the DRL agent and the current states. Also, the sum of the probabilities is equal to one:

$$\sum p(s'_\tau, R_\tau|s_\tau, a_\tau) = 1. \tag{21}$$

4) REWARD FUNCTION

The feedback from the MEC network to the DRL model can be computed when an action has been taken for a set of offloading tasks, which often called rewards in DRL. Specifically, a reward described as the number of completed tasks before expired minus the corresponding energy and time consumption, formulated as:

$$R_\tau(s_\tau, a_\tau) = (1-\omega)\lambda_1 F_{\tau,k} - \omega\lambda_2 \log_5(E_{\tau,k}) - \log_3(t^{total}_{\tau,k}) + \mathcal{C}. \tag{22}$$

This equation indicates that the agent is awarded when the offloaded task has been processed before it has expired denoted F_τ. The agent is punished through energy and time consumption term. The energy E_τ and time $t^{total}_{\tau,k}$ consumption values have been smoothed with logarithm function because the learning model can suffer from the feedback of fluctuation of the energy and time consuming if we use the original value. Moreover, $\mathcal{C}$ is a small constant value to encourage the model to keep running and accumulate rewards over time steps. The explanation of other parameters have been defined in equation (16).

To maximize the long-term accumulated rewards of the proposed model, each action has been evaluated with expected long term reward, which can be given by:

$$\max \mathbb{E}\left[R_\tau + \gamma G_{\tau+1}\right], \tag{23}$$

where R_τ is the immediate reward, and $\gamma G_{\tau+1}$ is the discounted long-term reward that can be calculated by equation (24). γ represents the discount for future reward, $\gamma \in [0, 1]$.

$$G_\tau = \sum_{k=0}^{\infty} \gamma^k R_{\tau+k}, \tag{24}$$

when $k = 0$ the feedback is the immediate reward R_τ. While $\gamma = 1$ and $k > 0$, the future reward is not discounted.

Since the environment only returns an immediate reward to the learning agent during the interaction and learning, the expected future rewards are often generated with a policy π, which is a sequence of actions corresponding to a set of states. The expected value of state s take action a called action-value function $Q(s,a)$, and the maximum $Q(s,a)$ is known as optimal action-value function $Q^*(s,a)$ given by:

$$Q^*(s,a) = \max_\pi \mathbb{E}\left[R_\tau + \gamma G_{\tau+1}|s_\tau = s, a_\tau = a, \pi\right]. \tag{25}$$

In other words, the target problem is equivalent to find an optimal policy π^* that can maximize the expected long-term reward. In practice, there are more than one optimal policy but we do not need find all of the optimal polices.

$$\pi^*(a \mid s) = \begin{cases} 1 & \text{if} a = \underset{a \in \mathcal{A}}{\operatorname{argmax}} Q^*(s,a) \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

B. PROPOSED MODEL

Deep Reinforcement Learning (DRL) introduced a deep neural network to replace the Q-function (Deep Q-Network or DQN) [18] as the approximator. It seems intuitive to replace the Q-table in the RL algorithm with a deep neural network [41]. However, the RL has to learn the right answer from continuous, evaluative, and sequential feedback. In other words, the DRL is different from supervised learning, the label data (feedback) of RL comes from RL iterative update. Therefore, the feedback is changing in each iterative. The feedback is a score for evaluating the action made based on the current state. The model can suffer from oscillation during exploitation and exploration because of the noisy feedback from the environment. To address this challenge, DQN using a target network to back up the Deep Q-Network and fixed the weights (aka, Fixation Method) during certain episodes.

The general process of the offloading system of UAV-assisted MEC and DRL agent learning process has been presented in Fig. 2. First, the users send the profile of the ready offloading tasks to the offloading control system denoted as Environment in the offloading system. Second, the Environment collects the mobile edge computing network's current states whenever it receives a task profile. Third, the DRL agent also resides at the UAV, and it takes actions based on the observation information from the Environment. Fourth, the Environment provides feedback and the next states of the MEC network to the DRL agent corresponding to the action, the feedback, and can be considered the evaluation of the action, and the feedback also knows as a reward. Note that there are two deep neural networks in the DRL agent called the local network and target network, respectively. The local network takes action for the Environment. Finally, the control agent (Environment) executes the action, which decides the proportion of the task allowing offloading to the UAV and driving it to the new location.

The training algorithm has been shown in Alg.1. The training process has shown in the following:

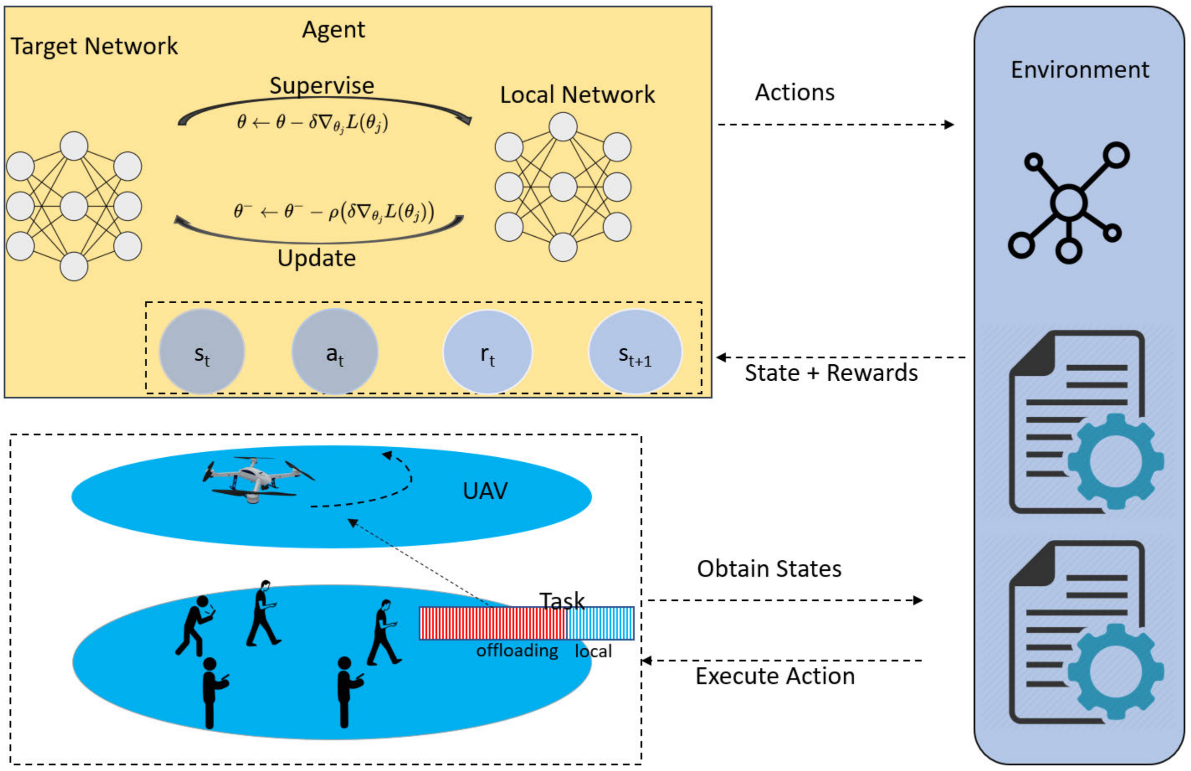


FIGURE 2. **UAV Offloading System.**

First, the experience replay buffer is to store the collected data. As shown in Fig. 2, the system generates a record whenever the agent takes action and interaction with the environment. And each record contains the current state s_τ, and the action a_τ has been made, and the reward r_τ, and the next state $s_{\tau+1}$, formed as tuple $< s_\tau, a_\tau, r_\tau, s_{\tau+1} >$. The experience reply buffer is a queue-like buffer with a fixed length; the new records are stored to the buffer, and it discards the oldest the records and keep the latest records when the buffer is full. Experience reply buffer is crucial to DRL learning and converges to robust policies because it is wasteful to use the sample only once and through away in conventional RL. In addition, training the deep learning model with multiple deep learning passes is very common; the number of epochs in deep learning defines train the model with the same training samples. The model can converge faster and learn from the rare samples that are considerably important to converge to a robust policy. Moreover, although we formulate the MEC network environment with an MDP framework, we prefer to decouple the sequential dependencies during the learning and interaction. Finally, we can reduce the noise of the training samples by using draw batches of the samples from experience buffer instead of using a single training sample at one time.

Second, the deep neural network (Deep Q-Network) for representing the Q-Value function has been defined. Note that the inputs and outputs sizes of the network are set as equal to state space and action space because the input will be the state and the outputs are the probabilities of the available actions. The target network is created by copying from the Deep Q-Network. The two copies of the networks have different purposes for training. The first copy is often called a local network and responsible for interacting with the environment and generating the training data samples. The target network is crucial for the training because it can prevent the learning model in suffering oscillation caused by the noise feedback from the environment. During the training, the model tries to minimize the loss between the Q-value from the target network and the local network. Let $Q^*\left(s', a'; \theta_\tau^-\right)$ be the optimal value from target network, and the outputs can be given by:

$$\bar{Y} = r + \gamma \max_{a'} Q^*\left(s', a'; \theta_\tau^-\right), \tag{27}$$

In other words, the model uses the target network values to supervise the local network to prevent suffering from oscillation. Therefore, the actual loss function Deep Q-Network can be given:

$$L_\tau(w_\tau) = \mathbb{E}_{s,a,r}\left[\left(\mathbb{E}_s[\bar{Y}|s, a] - Q\left(s, a; \theta_\tau\right)\right)^2\right], \tag{28}$$

Third, a score window has been internalized to smooth the reward score. With the complexity of the mobile edge computing network system and noisy feedback, the rewards are still considerably noisy; therefore, it is more reliable to evaluate the model with an average score of all the rewards in the current window. The score window is a queue, and it updates along with the training; the score window discards the old score and keeps the latest score when the queue is full.

Algorithm 1 E2DQN-Learning for UAV-MEC

```
Input: $epoch\_no$, $\epsilon_{start}$, $\epsilon_{end}$
Output: $loss$, $gradients$
//1. Initialization Section:
Initialize replay memory $D$ with capacity $N$;
Initialize action-value $Q$ with random weights $\theta$;
Initialize target action-value $\hat{Q}$ weights $\theta^- \leftarrow \theta$;
Initialize rewards window size;
$\epsilon \leftarrow \epsilon_{start}$;
for $episode \leftarrow 1$ to $M$ do
    Reset the UAV-MEC environment;
    Initial input raw data $s_1$;
    Prepossessing: $S \leftarrow \phi(< s_1 >)$;
    for time step: $\tau \leftarrow 1$ to $T_{max}$ do
        // 2. Generate training data section:
        Select action $a$ from based on $s_\tau$ using:
        $\pi \leftarrow \epsilon - Greedy(Q(s_\tau, a, \theta))$;
        Take action $a$, to get feedback and next state
        $s_{\tau+1}$;
        Prepossessing next state: $s_{\tau+1} \leftarrow \phi(s_{\tau+1})$;
        Save experience tuple $(s_\tau, a, r, s_{\tau+1})$ into replay
        memory $D$;
        // 3. Training Section:
        Draw mini-batch of experiences $(s_j, a_j, r_j, s_{j+1})$
        from $D$;
        if $s_{terminal} == s_{j+1}$ then
            Set target $\bar{Y}_j \leftarrow r_j$;
        else
            Set target $\bar{Y}_j \leftarrow r_j + \gamma \max_{a'} \hat{Q}\left(s_j, a, \theta^-\right)$;
        Update local network:
        $\theta \leftarrow \theta - \delta \nabla_{\theta_j} L\left(\theta_j\right)$ with $Adam$;
        Soft-update the target network:
        $\theta^- \leftarrow \theta^- - \rho(\delta \nabla_{\theta_j} L\left(\theta_j\right))$;
    $\epsilon \leftarrow max(\epsilon_{end}, \epsilon * decay)$;
    Store score for current episode;
```

Fourth, the algorithm starts to train the model by starting an episode. We defined an episode ends when the UAV server has full, which indicates the waiting time is longer than a threshold. We have to reset the MEC network environment whenever we start a new episode. As described in the previous section, we want the model to keep the UAV server running without an explosion. Before the state features have been fed into the model, the features have been prepossessed. The feature value scale is considerably different that can miss-lead the model may bias on feature with significant numerical values. However, less critical features, and ignore the critical feature represented by small numerical values. Therefore, we normalize the components of the state before concatenating together as one state and input to the model.

Fifth, the agent starts to interact with the mobile edge computing environment and the local network by incorporating a $\epsilon - greedy$ algorithm. Each interacts to produce an experience tuple, including current state, action, reward, and next state, denoted as (S, A, R, S'); these experience tuples are collected and stored in the experience buffer for training the local network. Note that the learning agent chooses the best action given the current state and policies with probability $1 - \epsilon$ and takes random action with probability ϵ. To balance the exploration and exploitation, the ϵ is decay over time because we want the model to spend more time exploring the environment at the beginning of the training that the later episodes. As the model has more knowledge about the environment, we want the learning agent to leverage more about the experience than exploration.

Finally, the agent draws sample batches of the experience, tuples the reply buffer, and train the local network. As mentioned above, the learning agent tries to minimize the loss (errors) between the outputs of the local network and the target network. The weights of the local network update every step of the gradient descent. In the original DQN method, the algorithm only updates the target network every $\mathcal{N}$ steps by overwriting the weights with the weights of the local network. In this work, we adopt the soft-update method introduced by Lillicrap *et al.* [42] update the target network smoothly instead of the Fixation method, which is a significant update every $\mathcal{N}$ steps. Concretely, the target network is updated with a small portion of change local weights discount ρ.We adopt the Adam [43] algorithm to optimize the loss function and update the weights of the local network. Note that generating training data block in the above step and training steps have to one step followed by others. Therefore, we can run server times of the training and a single time data generation steps; they also can run simultaneously.

In summary, the algorithm starts to initialize the replay buffer, the local network, target network, and score window size. Moreover, the proposed DRL model resets the MEC network every episode, and the learning agent interacts with the environment and generates training data and stores into the experience buffer. The DRL agent can draw sample data from the experience buffer to train the local network. The target network will be overwritten by the local network at every $\mathcal{N}$ step.

V. SIMULATION RESULTS

In this section, we present the details of the simulation configuration and results. We adopt Python[1] as the programming language and PyTorch[2] to build the proposed DRL model. The simulator has three main components. including the system environment, UAV-assisted MEC, and the DQN agent. The system environment describes the whole MEC offloading environment as MDP. The UAV-assisted MEC simulates the MEC network. The DRL model takes actions for the MDP environment of UAV-assisted MEC. The proposed DRL model has been compared the proposed methods with the greedy algorithm, reinforcement learning (Q-learning) and

[1]https://www.python.org
[2]https://pytorch.org

existing DRL model. Note that the existing reinforcement learning models and DRL models have to incorporate traditional optimization to achieve joint optimization.

A. SIMULATION SETUP

The UAV-assisted MEC is constructed by various MEC entities, including the UAV, users, and network environments. Further, the environment wraps up the UAV-assisted MEC and provides the interface to the DRL agent. For the simplicity of the argument, we call the UAV-assisted MEC and its environment wrapper as MEC environment in the following section. The initial position of the UAV is located at [10, 10, 25] with the frequency of 16×10^9 Hz. The expiration queue time and the task number constraint are 10 seconds and 500, respectively. The MEC environment is considered as overloaded when the task waiting time in the UAV task queue is longer than 10 seconds, or the number of tasks in the UAV queue Λ exceeds 500. The simulator terminates the current episode and start a new episode when MEC environment is overload. The key parameters settings are listed in the TABLE.2

TABLE 2. Parameter Setting.

Notation	The definition of notation	Value Range/Value
$D_{\tau,k}$	the task data size	$[2 \times 10^7, 2 \times 10^8]$
f_k	devices CPU frequency of k^{th} user	$[8 \times 10^8, 8 \times 10^9]$
f_{UAV}	the CPU frequency of UAV frequency	32×10^9
Δ_{max}	the max waiting time	$[10s, 30s]$
Λ_{max}	the max number of task in the queue	500
α	the environment related variable 1	9.61
β	the environment related variable 2	0.16
f_c	the carrier frequency	2000
c	the light speed	3×10^5km/s
η_{Los}	the environment dependent path loss for LoS	1.0
η_{NLos}	the environment dependent path loss for NLoS	20
$p_{i,k}$	the transmission power of mobile user	36
B	the bandwidth	10^7
c	the light speed	3×10^5km/s
κ	the hardware-related constant	10^{-26}
C	the required CPU cycles to calculate per bit data	$[5 \times 10^{10}, 6 \times 10^{10}]$
ω	the proportional term for task completion time and energy consumption	0.5
γ	the discount rate	0.9
eps	the number of episode	15000

Next, we present the neural network settings used for the DRL agents to optimize the target problem. Since our goal is to maximize the number of completed tasks before expiring and minimize the energy and time consumption by optimizing the UAV trajectory and the offloading ratio. We input the state of UAV, users, and environment into the neural network and output the combination of the UAV position and the offloading ratio.The input parameters include the task transmission speed, UAV current queue waiting time, UAV current position, UAV frequency, the position of current served user, its frequency, and the data size, computation cycles, and expiration time of the task. There are nine action options to move UAV position to control the UAV trajectory; further discretize the ratio offloading [0, 1] into five intervals. Therefore, the state space (neurons in the input layer) and the action space (neurons in the output layer) are, 11 and 45 respectively. Besides the input layer and output layer, we also build 5 hidden layers, which have 256, 512, 1024, 512,, and 256 nodes in the corresponding layer. During the training the process, the batch size and the buffer size are set as 256 and 10^5, respectively.

Moreover, the DQN has to balance exploration and exploitation to find the optimal policies. In this simulation, we adopt the ϵ-greedy algorithm, where the agent randomly generates a number which is ranged at [0, 1]. When the generated number is greater than ϵ, the agent exploits the learned knowledge, which means the agent takes the best action according to the current optimal policy. On the other hand, the agent takes a random action to explore the environment if the generated number is less than or equal to ϵ. The ϵ started from 1.0 and decay to 0.001 gradually. Finally, the future expected rewards are discounted with γ, which is defined as 0.9.

B. RESULTS

To demonstrate the efficiency and robustness of the proposed DRL model, we compare the proposed with existing works given the aforementioned parameter settings. The UAV-assisted MEC system performance is impacted by various factors, such as the number of mobile users, the MEC environment traffic, and different parameter settings of the DRL models. Therefore, we also show the performance of the model in different parameter settings.

To figure out the heuristic settings and performance improvements of the proposed end-to-end DRL (E2DQN), we run 2, 000 episodes to compare the performance of the model with different ϵ values. Fig. 3 shows the small ϵ has less exploitation the faster the model converges to its local optimal. However, the model with $\epsilon = 0.99$ can achieve

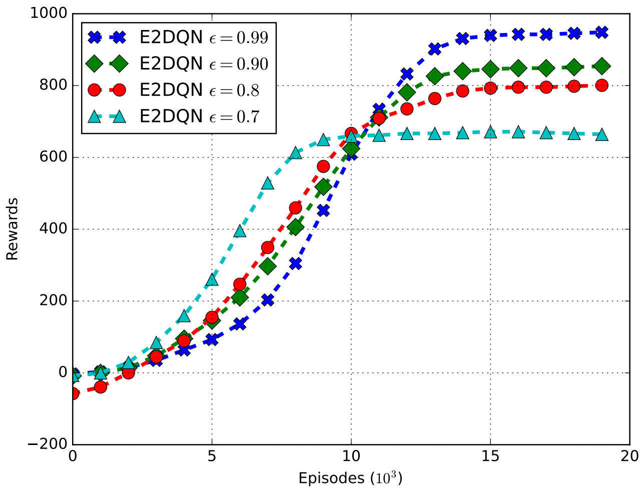


FIGURE 3. Exploration and Exploitation (ϵ Settings).

the highest, and $\epsilon = 0.7$ end up the lowest performance in this simulation. The models with large ϵ values take more time to explore the environment than those with small ϵ. Although the model with larger ϵ can achieve higher performance, the large ϵ is not always the best option because the exploration costs computational power and take a longer time to converge its optimal policies. Nevertheless, we choose to set $\epsilon = 0.99$ because it can collect more rewards than others.

To verify the performance of the proposed DRL model, we compare the DQN with the greedy algorithm (Greedy), reinforcement learning (Q-learning), and the existing DRL model (DQN) by collecting rewards. In this work, each reward contains weighted values of the completed task number and the energy consumption. As shown in Fig. 4, the proposed model (E2DQN) achieves the highest performance because it can lean to optimize the target problem in an end-to-end manner. Contrarily, the exiting DRL model (DQN) and reinforcement learning model (Q-learning) require further steps to optimize multiple objectives. Specifically, the DQN and Q-learning agents learn to control trajectory and classical optimizer to optimize the task offloading ratio. The classical optimization steps often have less room to optimize based on the outputs (actions) generated by the learning agent, which compromise the performance of the models. The learning agents are learning "knowledge" and exploring the system environment in the early period, and the greedy always selects the optimal local actions to collect rewards. Therefore, the learning agents have lower performance than the greedy algorithm in the early period. As they accumulate sufficient "experiences", the learning agents consider the long-term reward and take optimal actions, but the greedy always selects the local and short-term optimal actions. Gradually, the learning agents perform the greedy algorithm.

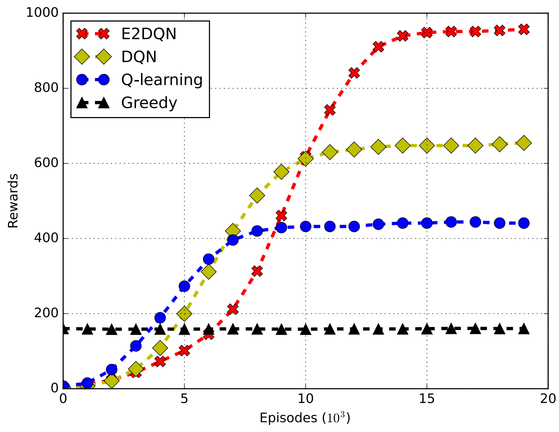


FIGURE 4. Rewards.

As shown in Fig. 4, Fig. 5 and Fig. 6, the Q-learning and existing DQN models converges to their optimal policies faster than the proposed model. The existing models take fewer time steps to learn because Q-learning cannot learn to capture complex state space; moreover, the exiting models only learn to optimize for a single target, and the rest of the optimization steps are processed by classical optimization methods such as CVX and MIP methods. Therefore, the exiting learning methods have relatively small action spaces. However, Q-learning and exiting DQN methods end up with relatively lower performance than the proposed method because it is challenging to assure the collaboration learning agents and optimization methods converge global optimal policies for joint optimization problems. As we can see from Fig. 4, Q-learning and existing DQN are converged about 750 and 1000 episodes, whereas the proposed converge to its optimal policies around 1,500 episodes.

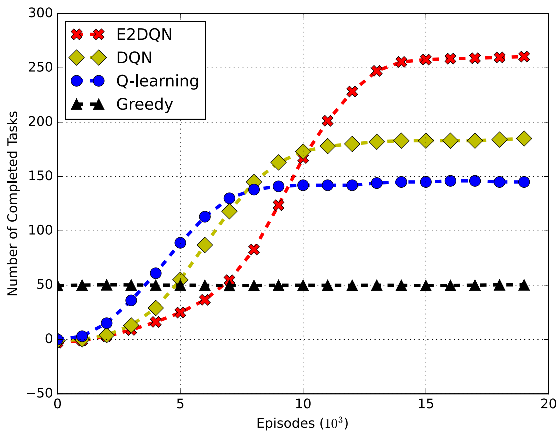


FIGURE 5. Completed Tasks.

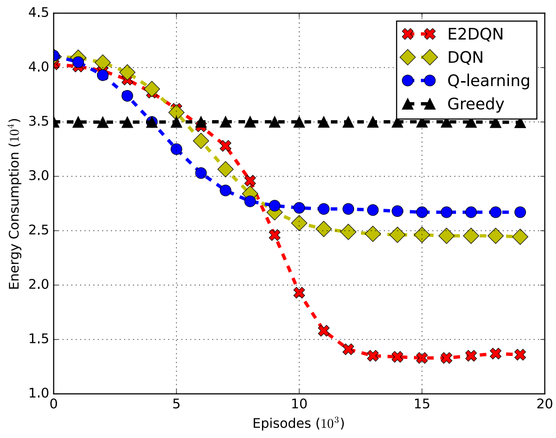


FIGURE 6. Energy Consumption.

Further, we compare the proposed model with other methods in terms of the reward components (sub-objectives). As shown Fig. 5, the proposed model can possess more tasks than existing DQN models, Q-learning, and the greedy algorithm. Moreover, Fig. 6 shows the proposed model consumes less energy than the other models. Again, the proposed method can optimize the objectives in an end-to-end manner to generate actions that optimize the UAV trajectory and offloading task ratio simultaneously. The proposed method can process more than 250 tasks, and the existing DQN method can process about 175 tasks; the Q-learning and greedy algorithm can about 148 and 50 tasks, respectively. The proposed model can save energy and increase number of processed tasks by optimizing the UAV trajectory and offloading task ratio simultaneously.

Finally, Fig. 7 shows the scalability of the models, and it shows the proposed model outperforms the DQN model and Q-learning models. The learning models can process more tasks with the increasing of users; on the contrary, the greedy algorithm performs extremely poorly in terms of scalability because it has no mechanism to plan to accumulate expected long-term rewards. Note that the greedy rewards are decreased when more tasks are offloading in a short time, which can lead to the server overloaded and end the current episode; the earlier the episode ends, the less reward the agent can collect. The greedy algorithm probably maintains the same rewards if there is no reset episode in the simulation. On the other hand, the learning algorithms can collect more rewards in an episode because they optimize the offloading ratio with respect to long-term rewards and maintain the stability of the MEC server.

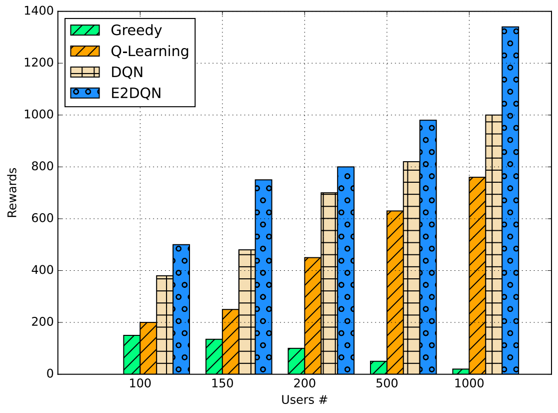


FIGURE 7. Scalability.

VI. CONCLUSION

In this paper, we have investigated the deployment of UAV-assisted MEC networks in a short time and temporally serve multiple users. Further, we proposed a deep reinforcement learning model to learn and optimize task offloading and UAV trajectory control in an end-to-end fashion. The proposed model controls the proportion of offloading tasks and UAV trajectory; the proposed model takes actions to optimize multiple targets, including computing latency, the energy consumption of the UAV-assisted MEC network. Specifically, the agent tries to maximize the number of tasks processed before they expire and simultaneously minimize energy and time consumption. The proposed model is an end-to-end model that does not require further optimization based on the output. The results show that the proposed model outperforms the existing methods. For future work, we will investigate multiple UAV collaboration with task partitioning.

REFERENCES

[1] H. Tran-Dang, N. Krommenacker, P. Charpentier, and D.-S. Kim, "Toward the Internet of Things for physical Internet: Perspectives and challenges," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4711–4736, Jun. 2020.

[2] Y. Zhang, J. Wang, X. Wang, and J. M. Dolan, "Road-segmentation-based curb detection method for self-driving via a 3D-LiDAR sensor," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 12, pp. 3981–3991, Dec. 2018.

[3] J. Grubert, T. Langlotz, S. Zollmann, and H. Regenbrecht, "Towards pervasive augmented reality: Context-awareness in augmented reality," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 6, pp. 1706–1724, Jun. 2017.

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.

[5] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energy-efficient UAV-assisted mobile edge computing: Resource allocation and trajectory optimization," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3424–3438, Mar. 2020.

[6] W. Zhang, L. Li, N. Zhang, T. Han, and S. Wang, "Air-ground integrated mobile edge networks: A survey," *IEEE Access*, vol. 8, pp. 125998–126018, 2020.

[7] Z. Yuan, J. Jin, L. Sun, K.-W. Chin, and G.-M. Muntean, "Ultra-reliable IoT communications with UAVs: A swarm use case," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 90–96, Dec. 2018.

[8] X. Diao, J. Zheng, Y. Cai, Y. Wu, and A. Anpalagan, "Fair data allocation and trajectory optimization for UAV-assisted mobile edge computing," *IEEE Commun. Lett.*, vol. 23, no. 12, pp. 2357–2361, Dec. 2019.

[9] X. Zhang, Y. Zhong, P. Liu, F. Zhou, and Y. Wang, "Resource allocation for a UAV-enabled mobile-edge computing system: Computation efficiency maximization," *IEEE Access*, vol. 7, pp. 113345–113354, 2019.

[10] H. Mei, K. Yang, Q. Liu, and K. Wang, "Joint trajectory-resource optimization in UAV-enabled edge-cloud system with virtualized mobile clone," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5906–5921, Jul. 2020.

[11] Q. Wang, A. Gao, and Y. Hu, "Joint power and QoE optimization scheme for multi-UAV assisted offloading in mobile computing," *IEEE Access*, vol. 9, pp. 21206–21217, 2021.

[12] Y. Liu, K. Xiong, Q. Ni, P. Fan, and K. B. Letaief, "UAV-assisted wireless powered cooperative mobile edge computing: Joint offloading, CPU control, and trajectory optimization," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2777–2790, Apr. 2020.

[13] K. Kim, Y. M. Park, and C. Seon Hong, "Machine learning based edge-assisted UAV computation offloading for data analyzing," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2020, pp. 117–120.

[14] F. Jiang, K. Wang, L. Dong, C. Pan, W. Xu, and K. Yang, "Deep-learning-based joint resource scheduling algorithms for hybrid MEC networks," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6252–6265, Jul. 2020.

[15] Z. Zhou, H. Liao, B. Gu, K. M. S. Huq, S. Mumtaz, and J. Rodriguez, "Robust mobile crowd sensing: When deep learning meets edge computing," *IEEE Netw.*, vol. 32, no. 4, pp. 54–60, Jul. 2018.

[16] L. Ale, N. Zhang, H. Wu, D. Chen, and T. Han, "Online proactive caching in mobile edge computing using bidirectional deep recurrent neural network," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5520–5530, Jun. 2019.

[17] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[18] M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[19] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and A. Nallanathan, "Deep reinforcement learning based dynamic trajectory control for UAV-assisted mobile edge computing," *IEEE Trans. Mobile Comput.*, early access, Feb. 16, 2021, doi: 10.1109/TMC.2021.3059691.

[20] B. Gu, X. Zhang, Z. Lin, and M. Alazab, "Deep multiagent reinforcement-learning-based resource allocation for Internet of controllable things," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3066–3074, Mar. 2021.

[21] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multi-agent deep reinforcement learning-based trajectory planning for multi-UAV assisted mobile edge computing," *IEEE Trans. Cognit. Commun. Netw.*, vol. 7, no. 1, pp. 73–84, Mar. 2021.

[22] H. Liao, Z. Zhou, W. Kong, Y. Chen, X. Wang, Z. Wang, and S. A. Otaibi, "Learning-based intent-aware task offloading for air-ground integrated vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, early access, Oct. 29, 2020, doi: 10.1109/TITS.2020.3027437.

[23] Z. Zhou, J. Feng, B. Gu, B. Ai, S. Mumtaz, J. Rodriguez, and M. Guizani, "When mobile crowd sensing meets UAV: Energy-efficient task assignment and route planning," *IEEE Trans. Commun.*, vol. 66, no. 11, pp. 5526–5538, Nov. 2018.

[24] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A s," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, 2017.

[25] M. Chen and 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.

[26] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[27] M. Hua, Y. Wang, Z. Zhang, C. Li, Y. Huang, and L. Yang, "Optimal resource partitioning and bit allocation for UAV-enabled mobile edge computing," in *Proc. IEEE 88th Veh. Technol. Conf. (VTC-Fall)*, Aug. 2018, pp. 1–6.

[28] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2049–2063, Mar. 2018.

[29] X. Hu, K.-K. Wong, K. Yang, and Z. Zheng, "UAV-assisted relaying and edge computing: Scheduling and trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4738–4752, Oct. 2019.

[30] J. Xiong, H. Guo, and J. Liu, "Task offloading in UAV-aided edge computing: Bit allocation and trajectory optimization," *IEEE Commun. Lett.*, vol. 23, no. 3, pp. 538–541, Mar. 2019.

[31] J. Zhang, L. Zhou, Q. Tang, E. C.-H. Ngai, X. Hu, H. Zhao, and J. Wei, "Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3688–3699, Apr. 2019.

[32] X. Diao, J. Zheng, Y. Wu, Y. Cai, and A. Anpalagan, "Joint trajectory design, task data, and computing resource allocations for noma-based and UAV-assisted mobile edge computing," *IEEE Access*, vol. 7, pp. 117448–117459, 2019.

[33] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, "Energy efficient resource allocation in UAV-enabled mobile edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 9, pp. 4576–4589, Sep. 2019.

[34] F. Zhou, Y. Wu, H. Sun, and Z. Chu, "UAV-enabled mobile edge computing: Offloading optimization and trajectory design," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[35] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, and G. Y. Li, "Joint offloading and trajectory design for UAV-enabled mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1879–1892, Apr. 2019.

[36] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1927–1941, Sep. 2018.

[37] X. Chen, T. Chen, Z. Zhao, H. Zhang, M. Bennis, and Y. Ji, "Resource awareness in unmanned aerial vehicle-assisted mobile-edge computing systems," in *Proc. IEEE 91st Veh. Technol. Conf. (VTC-Spring)*, May 2020, pp. 1–6.

[38] C. Zhan, H. Hu, X. Sui, Z. Liu, and D. Niyato, "Completion time and energy optimization in the UAV-enabled mobile-edge computing system," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7808–7822, Aug. 2020.

[39] K. Li, W. Ni, E. Tovar, and M. Guizani, "Joint flight cruise control and data collection in UAV-aided Internet of Things: An onboard deep reinforcement learning approach," *IEEE Internet Things J.*, early access, Aug. 24, 2020, doi: 10.1109/JIOT.2020.3019186.

[40] R. I. Bor-Yaliniz, A. El-Keyi, and H. Yanikomeroglu, "Efficient 3-D placement of an aerial base station in next generation cellular networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–5.

[41] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

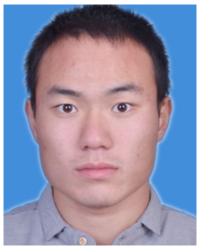
[42] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–14.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, pp. 1–14, Dec. 2014.

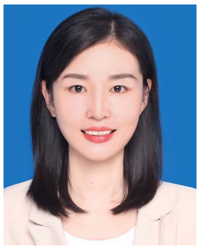
LU ZHANG received the B.S. degree in integrated circuit design and integrated systems, in 2013, and the master's degree in software engineering domain engineering, in 2015. He is currently pursuing the Ph.D. degree in environmental science with the Institute of Urban Environment, Chinese Academy of Sciences, China. His research interests include intelligent computing, intelligent sensing, and complex urban environment.

ZI-YAN ZHANG received the Bachelor of Science degree in life science from Fujian Normal University, Fuzhou, China, in 2006, the master's degree in life science from Xiamen University, Xiamen, China, in 2009, and the Ph.D. degree in environmental sciences from the University of Chinese Academy of Sciences, Beijing, China, in 2017. She has been working as an Assistant Researcher with the Key Laboratory of Urban Environment and Health, Institute of Urban Environment, Chinese Academy of Sciences, China. Her research interests include environmental monitoring and environmental health effects.

LUO MIN received the B.S. degree in integrated circuit design and integrated systems, in 2013, and the master's degree in electronic science and technology, in 2016. He is currently pursuing the Ph.D. degree in electronic information with the University of Electronic Science and Technology of China. His research interests include intelligent computing, intelligent sensing, and terahertz devices.


CHAO TANG received the B.S. degree in information and computing science from Ludong University, Yantai, Shandong, China, in 2018. He is currently pursuing the M.S. degree in environmental engineering with the University of Chinese Academy of Sciences, China. His research interests include the Internet-of-Thing application and electromagnetic environmental monitoring.

HONG-YING ZHANG received the Bachelor of Agriculture degree in agricultural resource and environment from Zhejiang University, China, in 2015. She is currently pursuing the Ph.D. degree in environmental science with the Institute of Urban Environment, Chinese Academy of Sciences, China. Her research interests include life sciences in space, simulated microgravity, and behavior of drosophila melanogaster.


YA-HONG WANG received the Bachelor of Engineering degree in environment engineering from the Zhengzhou University of Aeronautics, in 2016. She is currently pursuing the Ph.D. degree in environmental science with the Institute of Urban Environment, Chinese Academy of Sciences, China. Her work experiences include motion-sleep, growing development, and reproductive capacity of drosophila melanogaster under long-term radiofrequency radiation.

PENG CAI received the Ph.D. degree from the Shanghai Institute of Biochemistry, Chinese Academy of Sciences, in 1995. He is currently a Researcher, a Ph.D. Supervisor, and a Group Leader of the Physical Environment Research Group, Institute of Urban Environment, Chinese Academy of Sciences. He focuses on the research of environment and health, especially the influence of special operating environment on the physical and mental health of personnel.