Priority-Aware Resource Scheduling for UAV-Mounted Mobile Edge Computing Networks

Wenqi Zhou, Lisheng Fan, Fasheng Zhou, Feng Li, Xianfu Lei, Wei Xu and Arumugam Nallanathan

***Abstract*—In this paper, we investigate the joint impact of task priority and mobile computing service on the mobile edge computing (MEC) networks, in which one unmanned aerial vehicle (UAV) provides mobile computing service to help compute the tasks from users in multiple hotspots where the task priority is time-varying. For such a system, we firstly measure the system performance by the computing utility of multiples users, where the effect of a wide-range task priority is incorporated. We then analyze the impact of the network wireless bandwidth and UAV computational capability on the system performance, from which we optimize the system through UAV hotspot selection and user task offloading. To solve the optimization problem, we further employ deep Q-learning algorithm to learn an effective solution by continuous interaction between the UAV agent and system environment. Simulations are finally conducted to verify the superiority of the proposed studies in this paper.**

***Index Terms*—Mobile edge computing, task priority, resource scheduling, deep Q-learning.**

I. Introduction

With the rapid development of information technologies, many novel mobile applications such as positioning, autopilot, virtual reality and augmented reality have emerged. Most of these applications are computation and latency sensitive tasks, which impose a severe burden on the latency and energy consumption of the conventional cloud computing paradigm. To support these applications, mobile edge computing (MEC) has been proposed to provide computing services at the network edge, which can assist the task computing for the users through wireless offloading [1]–[9]. In this area, the authors in [1] combined local and data computing into a joint computation scheme to optimize the total computation efficiency of users, and [2] devised a reinforcement learning based mobile offloading scheme for edge computing to improve the utility of mobile devices. In addition, various researches have been performed to optimize the offloading strategy for the MEC networks by fully exploiting the system communication and computing resources, such as transmit power [4], [5], [7], wireless bandwidth [8], and computational capability [9]. However, the system performance is often significantly affected by the task characteristics such as priority, and thus it is of vital importance to take into account the priority during the task computing for the system optimization [10].

Most of existing works on the MEC networks considered fixed computing resources with limited coverage, which however cannot flexibly serve users with time-varying task characteristics in practice. To solve this issue, mobile computing service provided by mobile edge server such as unmanned aerial vehicles (UAVs), mobile vehicles and mobile robots has been equipped in the MEC networks, which can flexibly exploit the communication and computing resources to compute the time-varying computing tasks of users. In this aspect, the authors in [11] studied an UAV-mounted MEC network and maximized the system reward through UAV scheduling and user selection to enhance the system performance. In addition, a vehicle-mounted edge computing network was investigated in [12], [13] to reduce the communication overhead, through the path scheduling and computation offloading. To the best of our knowledge, there has been little work on the joint impact of task priority and mobile computing service on the MEC networks, which motivates the work in this paper. However, there are two critical challenges in exploiting the task priority and mobile computing service for optimizing the MEC network. One challenge is how to quantify and evaluate the performance of computing different priority tasks, while the other challenge is how to intelligently make decisions for the mobile server based on the time-varying task characteristics. These two challenges motivate the work in this paper.

In this paper, we investigate the joint impact of task priority and mobile computing service on the MEC networks, in which the UAV provides mobile computing service to help compute the tasks from users in multiple hotspots where the task priority is time-varying. For such a system, we firstly measure the system performance by the computing utility of multiples users, where the effect of a wide-range task priority is incorporated. We then analyze the impact of the network wireless bandwidth and UAV computational capability on the system performance, from which we optimize the system through UAV hotspot selection and user task offloading. To solve the optimization problem, we employ deep Q-learning algorithm to learn an effective solution by continuous interaction between the UAV agent and system environment. Simulations are finally conducted to verify the superiority of the proposed studies in this paper.

W. Zhou, L. Fan, and F. Zhou are with the School of Computer Science, Guangzhou University, Guangzhou, China (e-mail: 2112006056@e.gzhu.edu.cn, lsfan@gzhu.edu.cn, zfs@gzhu.edu.cn).

Feng Li is with the School of Computer Science and Technology, Shandong University, Qingdao 266237, China (e-mail: fli@sdu.edu.cn).

X. Lei is with the Provincial Key Lab of Information Coding and Transmission, Southwest Jiaotong University, Chengdu 610031, China, and also with National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China (e-mail: xflei@swjtu.edu.cn).

W. Xu is with the National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China (e-mail: wxu@seu.edu.cn).

A. Nallanathan is with the School of Electronic Engineering and Computer Science, Queen Mary University of London, London, U.K (e-mail:a.nallanathan@qmul.ac.uk).

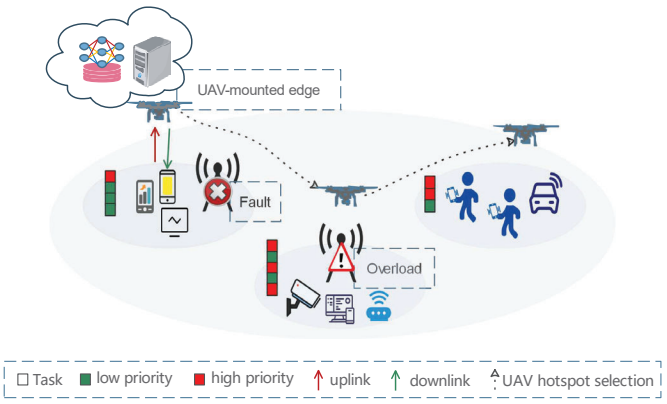

Fig. 1. UAV-mounted mobile edge computing network.

II. System Model

Fig. 1 depicts an UAV-mounted mobile edge computing network, where one UAV exists in the system and provides some computational services to L hotspots indexed by $\mathcal{L} = \{1, 2, ..., L\}$ through wireless links. For hotspot $l \in \mathcal{L}$, its location is denoted by the coordinate $p_l = \{x_l, y_l\}$, and there are M user devices (UDs) indexed by $\mathcal{M} = \{1, 2, ..., M\}$ in each hotspot. In practice, the task characteristics of users may be time-varying due to dynamical requirements, and accordingly, the UAV should select one proper hotspot to serve and design the associated task offloading strategies for the users. In the following, we will discuss the task offloading and computing model, as well as the task utility model for the considered network.

A. Task Offloading and Computing Model

Let $c_l \in \{0, 1\}$ indicate the hotspot selection, where hotspot l is selected to be served by the UAV if $c_l = 1$, or not otherwise. As the UAV assists only one hotspot to compute, the indicator c_l should satisfy the constraint of $\sum_{l=1}^{L} c_l = 1$. After the hotspot selection, the UAV will fly to the selected hotspot[1] and assist the users in the selected hotspot to compute tasks. In this case, the users in the selected hotspot can offload the tasks to the UAV via the wireless links or compute the tasks locally. We use $b_{m,l} \in \{0, 1\}$ to denote the offloading decision, where $b_{m,l} = 1$ indicates that the task of user m in hotspot l is offloaded to the UAV for computing, while $b_{m,l} = 0$ indicates the local computing.

When user m in hotspot l selects to offload its task to the UAV for computing, the data rate and offloading latency are

$$r_{m,l} = B \log_2 \left(1 + \frac{P_{m,l}|h_{m,l}|^2}{\sigma^2}\right), \tag{1}$$

$$T_{m,l}^{trans} = \frac{s_{m,l}}{r_{m,l}}, \tag{2}$$

where B is the wireless bandwidth of each user, $h_{m,l} \sim \mathcal{CN}(0, \xi)$ is the channel parameter of the wireless link from user m to the UAV, $P_{m,l}$ denotes the transmit power at user m in hotspot l, σ^2 is the AWGN variance at the UAV, and $s_{m,l}$ is user task size in Mbits.

After the task offloading, the UAV helps compute the tasks from user m, and the computational latency is

$$T_{m,l}^{uav} = \frac{s_{m,l}\zeta}{f_{uav}}, \tag{3}$$

where ζ is the CPU cycles required to compute per bit of the task, f_{uav} is the computational capability that the UAV assigns to the tasks offloaded to the UAV for computing.

On the other hand, when user m of hotspot l selects to compute its task at local, the computational latency is

$$T_{m,l}^{local} = \frac{s_{m,l}\zeta}{f_{m,l}}, \tag{4}$$

where $f_{m,l}$ is the computational capability of user m.

From (2)-(4), we can obtain the processing latency of user m in hotspot l as

$$\begin{aligned} T_{m,l} = c_l \left[b_{m,l}\left(T_{m,l}^{trans} + T_{m,l}^{uav}\right) + (1 - b_{m,l})\, T_{m,l}^{local}\right] \\ + (1 - c_l) T_{m,l}^{local}. \end{aligned} \tag{5}$$

B. Task Utility Model

Different tasks may have various priorities in the practical communication and computing. Let v_m denote the task priority of user m, where $v_m = 1$ and $v_m = 0$ indicate the tasks of high priority and low priority, respectively. In addition to these two priorities, the medium priority is characterized by $v_m \in (0, 1)$. The high-priority tasks occur in some communication scenarios such as vehicle navigation and road-sensing, where a strict latency constraint is imposed on the tasks. On the contrary, the low-priority tasks occur in the communication scenarios such as entertainment applications and value-added services, where the computing latency is tolerant. Besides these two priorities, the medium priority exists in some communication scenarios such as many public safety. In this case, the tasks need to be processed in time if possible, whereas the urgency is not as high as the high-priority tasks.

For the high-priority tasks with $v_m = 1$, a logarithmic function can be used to represent the utility of task processing [10],

$$\begin{aligned} \mathcal{U}_{m,l}^{H} = \log(1 + \tau_{m,l} - T_{m,l}) I(T_{m,l} \le \tau_{m,l}) \\ - \Gamma_H I(T_{m,l} > \tau_{m,l}), \end{aligned} \tag{6}$$

where $\tau_{m,l}$ is the tolerant latency threshold of user m in hotspot l, and $-\Gamma_H$ is a negative reward that represents the penalty of failure to complete the high-priority task within the latency threshold $\tau_{m,l}$. Notation $I(x)$ is an indicator function, equal to 1 if x is true or 0 otherwise.

For the low-priority tasks with $v_m = 0$, the utility of the task processing is [10]

$$\begin{aligned} \mathcal{U}_{m,l}^{L} = \Gamma_L e^{-\alpha(T_{m,l} - \tau_{m,l})} I(T_{m,l} > \tau_{m,l}) \\ + \Gamma_L I(T_{m,l} \le \tau_{m,l}), \end{aligned} \tag{7}$$

where Γ_L is a positive constant to measure the reward of completing a low-priority task within the given latency threshold $\tau_{m,l}$, and $\alpha > 0$ is an exponential decay factor. Besides the above two priorities with $v_m = 1$ and $v_m = 0$, a much wider range of priority, i.e., $v_m \in (0, 1)$, exists to

[1] In this paper, we assume that the UAV connects to the nearby hotspots only, as a long-distance service would impose a severe burden of latency and cost on the system. Therefore, the flying latency of UAV among different hotspots can be ignored.

measure the practical task processing. To measure the task processing utility of the medium priority, a linear interpolation method is used for user m in hotspot l as,

$$\mathcal{U}_{m,l} = v_{m,l}\mathcal{U}^H_{m,l} + (1 - v_{m,l})\mathcal{U}^L_{m,l}. \tag{8}$$

In particular, when the task priority becomes high, the utility $\mathcal{U}_{m,l}$ converges to $\mathcal{U}^H_{m,l}$, while the utility $\mathcal{U}_{m,l}$ converges to $\mathcal{U}^L_{m,l}$ when the task priority becomes low. From the above utilities, we can summarize the system utility of the M users to process the L tasks as

$$\mathcal{U}_{sys} = \sum_{m=1}^{M}\sum_{l=1}^{L}\mathcal{U}_{m,l}. \tag{9}$$

III. Performance Analysis and Problem Formulation

From (9), we can find that there are many factors that affect system utility, such as the wireless bandwidth, computational capability at the UAV, as well as the hotspot selection and task offloading strategy. In this section, we firstly analyse the impact of the wireless bandwidth and computational capability at the UAV on the system utility, based on which we design the system optimization problem by optimizing the hotspot selection and offloading strategy.

From (1)-(5), we can rewrite the task processing latency of user m in hotspot l as

$$T_{m,l} = c_l\Bigg[b_{m,l}\left(\frac{s_{m,l}}{B\log_2\left(1+\frac{P_{m,l}|h_{m,l}|^2}{\sigma^2}\right)} + \frac{s_{m,l}\zeta}{f_{uav}}\right) + \underbrace{(1-b_{m,l})\frac{s_{m,l}\zeta}{f_{m,l}}}_{G_1}\Bigg] + \underbrace{(1-c_l)\frac{s_{m,l}\zeta}{f_{m,l}}}_{G_2}, \tag{10}$$

which decreases with a larger wireless bandwidth B and computational capability f_{uav} at the UAV. In particular, when f_{uav} becomes sufficiently large, $T_{m,l}$ can be asymptotically approximated by

$$T_{m,l} \simeq c_l\left[\frac{b_{m,l}s_{m,l}}{B\log_2\left(1+\frac{P_{m,l}|h_{m,l}|^2}{\sigma^2}\right)} + G_1\right] + G_2. \tag{11}$$

Then, by substituting (11) into (9), the asymptotic expression of the system utility can be obtained in (12), which shows that the impact of f_{uav} on the system utility of task processing almost disappears. Therefore, we can obtain the following insights,

Remark 1: When f_{uav} is sufficiently large, the task computational latency at the UAV can be negligible and the offloading latency becomes dominant in task processing latency, leading to that the impact of f_{uav} on the system utility of task processing becomes marginal.

When B becomes sufficiently large, $T_{m,l}$ can be asymptotically approximated by

$$T_{m,l} \simeq c_l\left(\frac{b_{m,l}s_{m,l}\zeta}{f_{uav}} + G_1\right) + G_2. \tag{13}$$

By substituting (13) into (9), the system utility can be asymptotically approximated by (14), which shows that the impact of B on the utility of task processing can be approximately ignored. Hence, we have the following insights,

Remark 2: When B is large enough, the task offloading latency is very small, and the task processing time mainly depends on the computational latency, thus the impact of B on the system utility of task processing can be ignored.

When f_{uav} and B both become sufficiently large, $T_{m,l}$ can be asymptotically approximated by

$$T_{m,l} \simeq c_lG_1 + G_2, \tag{15}$$

and we can obtain the asymptotic expression of the utility by substituting (15) into (9), as shown in (16), which shows that the impact of f_{uav} and B on the utility of task processing can be approximately ignored, and then we have the following insight,

Remark 3: When f_{uav} and B become sufficiently large, the task offloading and computational latency at the UAV is small. Hence, the impact of f_{uav} and B on the system utility can be almost ignored, and the system performance is thus dependent on the hotspot selection and the task offloading.

According to the above analysis result, we further design the system optimization problem by maximizing the utility $\mathcal{U}_{sys}$, through optimizing the hotspot selection and offloading strategy, formulated by

$$\mathbf{P1}: \max_{\{b_{m,l},c_l|1\le m\le M,1\le l\le L\}} \mathcal{U}_{sys}, \tag{17}$$

$$\text{s.t.} \quad C_1: b_{m,l}\in\{0,1\}, \forall m\in\mathcal{M}, \forall l\in\mathcal{L}, \tag{17a}$$

$$C_2: c_l\in\{0,1\}, \forall l\in\mathcal{L}, \tag{17b}$$

$$C_3: \sum_{l=1}^{L} c_l = 1, \tag{17c}$$

where C_1 and C_2 are the constraints on the task offloading and hotspot selection, respectively, and constraint C_3 ensures that the UAV can only select one hotspot to serve.

Note that in practice, the task characteristics and wireless channels may dynamically vary with time, and the hotspot selection and task offloading are both discrete actions, which gives much difficulty to solve the optimization problem **P1** by using the traditional method such as convex optimization. Although we can use the conventional branch and bound (BnB) method to obtain the optimal solution, it will produce huge computational complexity in practical applications. Hence, we turn to use the deep Q-learning to optimize the problem **P1**, as it can provide an effective solution through the interaction between the UAV agent and the system environment, and makes the UAV to adaptively complete the tasks to further realize the purpose of commercial use. In the next section, we will detail how to optimize the problem **P1** by employing the deep Q-learning algorithm.

IV. Optimization with Deep Q-learning Algorithm

In this section, we use the deep Q-learning to optimize the system problem **P1** in (17). In particular, we firstly use the Markov decision process (MDP) to model the process between the agent and system environment, based on which we employ the deep Q-learning to tackle the problem **P1**.

$$\mathcal{U}_{sys} \simeq \sum_{l=1}^{L}\sum_{m=1}^{M}\left[v_{m,l}\log\left(1+\tau_{m,l}-\frac{c_l b_{m,l}s_{m,l}}{B\log_2\left(1+\frac{P_{m,l}|h_{m,l}|^2}{\sigma^2}\right)}-c_lG_1-G_2\right)+(1-v_{m,l})\Gamma^L\right]\times I\left[\frac{c_l b_{m,l}s_{m,l}}{B\log_2\left(1+\frac{P_{m,l}|h_{m,l}|^2}{\sigma^2}\right)}+c_lG_1+G_2\le\tau_{m,l}\right]+\left[(1-v_{m,l})\Gamma^L\times\exp\left(-\alpha\left(c_l\left(\frac{b_{m,l}s_{m,l}}{B\log_2\left(1+\frac{P_{m,l}|h_{m,l}|^2}{\sigma^2}\right)}+G_1\right)+G_2-\tau_{m,l}\right)\right)-v_{m,l}\Gamma^H\right]\times I\left[c_l\left(\frac{b_{m,l}s_{m,l}}{B\log_2\left(1+\frac{P_{m,l}|h_{m,l}|^2}{\sigma^2}\right)}+G_1\right)+G_2>\tau_{m,l}\right]. \tag{12}$$

$$\mathcal{U}_{sys} \simeq \sum_{l=1}^{L}\sum_{m=1}^{M}\left[v_{m,l}\log\left(1+\tau_{m,l}-\frac{c_l b_{m,l}s_{m,l}\zeta}{f_{uav}}-c_lG_1-G_2\right)+(1-v_{m,l})\Gamma^L\right]\times I\left[c_l\left(\frac{b_{m,l}s_{m,l}\zeta}{f_{uav}}+G_1\right)+G_2\le\tau_{m,l}\right]+\left[(1-v_{m,l})\Gamma^L\exp\left(-\alpha\left(c_l\left(b_{m,l}\frac{s_{m,l}\zeta}{f_{uav}}+G_1\right)+G_2-\tau_{m,l}\right)\right)-v_{m,l}\Gamma^H\right]\times I\left[c_l\left(\frac{b_{m,l}s_{m,l}\zeta}{f_{uav}}+G_1\right)+G_2>\tau_{m,l}\right]. \tag{14}$$

$$\mathcal{U}_{sys} \simeq \sum_{l=1}^{L}\sum_{m=1}^{M}\left[v_{m,l}\log(1+\tau_{m,l}-c_lG_1-G_2)+(1-v_{m,l})\Gamma^L\right]\times I(c_lG_1+G_2\le\tau_{m,l})+\left[(1-v_{m,l})\Gamma^L\times\exp\left(-\alpha(c_lG_1+G_2-\tau_{m,l})\right)-v_{m,l}\Gamma^H\right]\times I(c_lG_1+G_2>\tau_{m,l}). \tag{16}$$

We firstly design the state space, action space and reward according to the MDP model, which can be characterized by the $S(t)$, $A(t)$ and $R(t)$. Specifically, we define the state space as

$$S(t)=\{s(t)|s(t)=\{p_l(t),b_{m,l}(t)\}\}, \tag{18}$$

for $l\in\{1,2,...,L\}$, and $m\in\{1,2,...,M\}$, where $p_l(t)$ is the coordinate of the hotspot chosen by the UAV at time slot t, and $b_{m,l}(t)$ is the offloading strategy at time slot t. Then, we define the action space, consisting of the hotspot selection of the UAV and the task offloading strategy of users, given by

$$A(t)=\{a(t)|a(t)=\{a_l^s(t),a_{m,l}^o(t)\}\}, \tag{19}$$

where $a_l^s(t)\in\{0,1\}$ is the action of hotspot selection, $a_{m,l}^o(t)\in\{0,1\}$ indicates the task offloading action. We further describe how to select an action. In general, ϵ-greedy algorithm is used to discover which action yields the most reward through trial and error, given by

$$a(t)=\begin{cases}random, & w.p.\epsilon,\\ \arg\max\limits_a Q(S(t),a;\boldsymbol{\theta}), & w.p.1-\epsilon.\end{cases} \tag{20}$$

where $\epsilon\in[0,1]$ is the probability of randomly selecting an action, $\boldsymbol{\theta}$ is the weighted parameter, and $Q(S,a;\boldsymbol{\theta})$ is the Q function, which represents the accumulative reward of performing an action a. When the action $a(t)\in A(t)$ is performed, the state $S(t)$ changes to $S(t+1)$, and we can obtain a reward from the utility accordingly, given by

$$r(t)=\begin{cases}\delta_1, & \text{If } U_{sys}(t+1)>U_{sys}(t),\\ -\delta_2, & \text{If } U_{sys}(t+1)=U_{sys}(t),\\ -\delta_3, & \text{If } U_{sys}(t+1)<U_{sys}(t),\end{cases} \tag{21}$$

where δ_1, δ_2 and δ_3 are three positive values. Note that $U_{sys}(t)$ and $U_{sys}(t+1)$ are the system utility at time slot t and $t+1$, respectively. We further employ the deep Q-learning to solve the problem by two deep neural networks. One network is the evaluation network, which can obtain the action-state value $Q(S,a;\boldsymbol{\theta})$ by inputting the current state, where $\boldsymbol{\theta}$ is the weighted factor of the evaluation network. The other network is target network, which is copied from the evaluation network and helps update the evaluation network. In the deep Q-learning, we use temporal different (TD) approach to update the network, and the loss function is

$$L(t)=(Z(t)-Q(S(t),a(t);\boldsymbol{\theta}))^2, \tag{22}$$

where the target value function $Z(t)$ is given by

$$Z(t)=r(t)+\eta\max_a\hat{Q}(S(t+1),a;\hat{\boldsymbol{\theta}}), \tag{23}$$

where η is a discount parameter and $\hat{\boldsymbol{\theta}}$ is the weighted factor of the target network. In addition, the target network copies the weighted parameter from the evaluation network every T_u time slot to update itself. After several epochs of training, the system will obtain an effective hotspot selection and offloading strategy. The whole procedure of deep Q-learning for the hotspot selection and task offloading is summarized in Algorithm 1.

V. Simulation Results and Discussions

In this part, we provide simulation results to verify the proposed studies. The wireless links in the network experience Rayleigh fading with the average channel gain of unity. The transmit power of users and the variance of AWGN are set to 2W and 1×10^{-2}W, respectively. Moreover, we set the

Algorithm 1 Deep Q-learning based hotspot selection and task offloading scheme.

Input: $B, f_{m,l}, f_{uav}$

Initialize experience replay buffer E;
Randomly initialize the weight $\boldsymbol{\theta}$ of the evaluation network;
Initialize the weight $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}$ of the target network;
for Epoch $= 1 : K$ **do**
  Initialize $v_{m,l}$, $s_{m,l}$ $h_{m,l}$;
  Randomly initialize state $S(t)$;
  for t=1:T **do**
    Obtain state $S(t)$ and select an action by (20);
    Take action $a(t)$ to obtain new state $S(t+1)$ and reward $r(t)$;
    Save transition $(S(t), S(t+1), a(t), r(t))$ in E;
    Randomly sample mini-batch $(S(t), S(t+1), a(t), r(t))$ from E;
    Calculate loss function $L(t)$ by (22);
    Set $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}$ every T_u time slot;
  end for
end for

Output: $p_l, b_{m,l}$

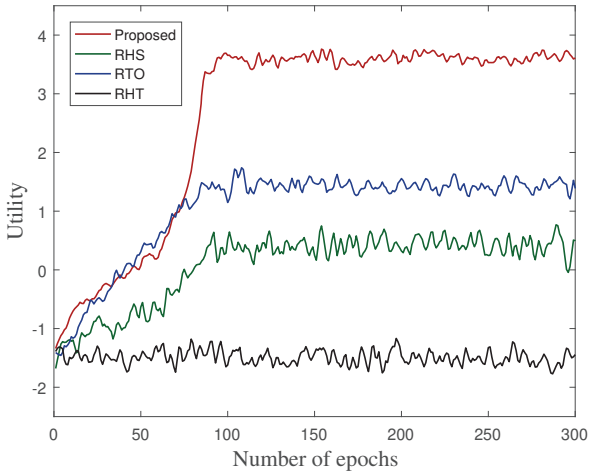

Fig. 2. Utility convergence of four schemes versus the training epoch.

computational workload ζ to 5cycle/bit, and the computational capability of each user is 1×10^7cycle/s. In further, the task size of each user in hotspot l follows uniform distribution of $s_{m,l} \sim \mathrm{U}(5l, 5(l+1))$, and the priority of tasks in hotspot l follows uniform distribution of $v_{m,l} \sim \mathrm{U}(l/L, (l+1)/L)$. If not specified, we set the number of hotspots to 6, the number of users in each hotspot to 5, the computational capability of the UAV to 6×10^8cycle/s, and the wireless bandwidth of each user to 30MHz.

Fig. 2 shows the utility convergence of several schemes versus the training epoch, where the number of epochs changes from 0 to 300. For comparison, we plot the utility of three competing schemes: random hotspot selection (RHS) which employs the deep-Q learning to optimize the task offloading while randomly selects the hotspot for the UAV, random task offloading (RTO) which employs the DQN algorithm to optimize the UAV hotspot selection while randomly offloads the tasks for users, and random hotspot selection and task offloading (RHT) which randomly selects the hotspot for the UAV and offload tasks of users. We can find from the figure that, except for the RHT, the utility of the other three schemes increases as the number of epochs increases until it converges. Specifically, the utility of the proposed scheme increases sharply from -1.3 to 3.6 and converges at epoch around 85, the utility of the RHS scheme increases from -1.5 to 0.3 and converges at epoch about 90, the utility of the RTO scheme increases from -1.4 to 1.5 and converges at epoch around 85, while the utility of RHT remains unchanged around -1.5. This is because that the deep Q-learning algorithm of RHS, RTO and the proposed scheme can exploit effective hotspot selection and the offloading strategies to improve the task utility of the system, while the RHT can not output an useful strategy and results in a poor system performance. Moreover, we can see that the utility of the proposed scheme is higher than that of the three competing schemes, as the proposed scheme can explore both the effective actions of hotspot selection and offloading decision. In particular, the performance of the proposed scheme is at least 58.3% superior to that of the three competing schemes, which verifies that the proposed mathod can effectively improve the system performance.

Fig. 3 illustrates the impact of the user number on the utility of the four schemes, where the number of users ranges from 2 to 12. Observing from the figure, we can find an interesting phenomenon that the utility of the proposed scheme and RTO scheme increases with the number of users, while that of RHS and RHT decreases along with the number of users. The major reason is that the proposed scheme and RTO can exploit the effective hotspot selection policy to select the hotspot that needs to be served by the UAV. Once the UAV determines the urgent hotspot that need to be served, the average system utility is positive, so that the average utilities of the proposed scheme and RTO increase with a larger number of users. In constrast, the other two schemes cannot effectively select the emergency hotspot, which results in the failure of computing the emergency tasks due to time-out and leads to a large negative penalty. Therefore, the average system utilities of RHS and RHT decreases with a larger number of users. Moreover, we also see that the proposed scheme is superior to the three competing schemes. For example, when the task number is 12, the utility of the proposed scheme is at least 48.9% larger than that of the three competing schemes, which proves the superiority of the proposed method to explore the hotspot selection and offloading strategy.

Fig. 4 presents the effect of the UAV computational capability on the utility of the four schemes, where the computational capability at the UAV ranges from 2×10^8cycles/s to 10×10^8cycles/s. The results of this figure show that the system utility grows with the UAV computational capability, but the improvement rate gradually vanishes. The reason is that, when the UAV computational capability is small, the task computational latency becomes dominant in the task execution time, and thus the utility increases significantly

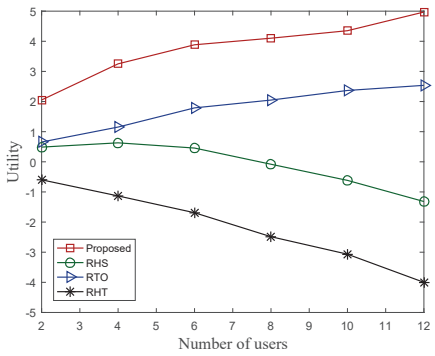

Fig. 3. Impact of number of users on the utility of the four schemes.

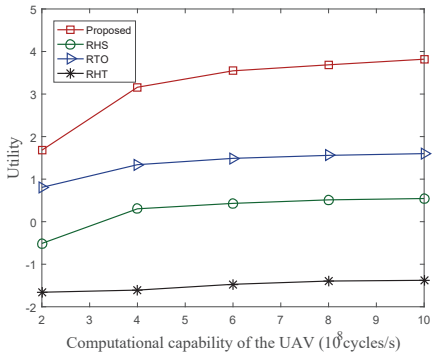

Fig. 4. Effect of the UAV computational capability on the utility of the four schemes.

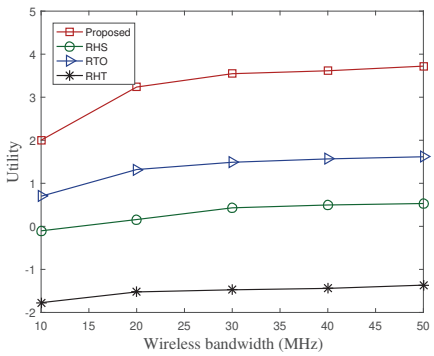

Fig. 5. Utility of the four schemes versus the wireless bandwidth.

with the UAV computational capability. However, when the UAV computational capability becomes large, the task computational latency can be negligible, and the improvement of the system utility with the UAV computational capability becomes marginal. These phenomena justify the insights in *Remark 1*. Moreover, we observe that the utility of the proposed scheme is always better than that of the other three schemes because of its superior ability to output the hotspot selections and offloading strategies. Specifically, the proposed scheme is at least 58.1% better than the three competing schemes when the computational capability of the UAV is 10×10^8cycles/s, which attests the effectiveness of the proposed scheme.

Fig. 5 shows the utility of the four schemes versus the wireless bandwidth, where the bandwidth ranges from 10MHz to 50MHz. By observing this figure, we can find that the system utility improves with the increased bandwidth, but the improvement rate gradually decreases. This is because that, when the system bandwidth is small, the task offloading latency is dominant in the task execution latency, and therefore the system utility enlarges significantly along with the bandwidth. However, when the bandwidth becomes large, the task offloading latency is very small, and the improvement of the system utility from the increased bandwidth becomes marginal. These phenomena justify the insights in *Remark 2*. Moreover, we find that the utility of the proposed scheme is always superior to that of the other three schemes due to the effective hotspot selections and offloading strategies explored by deep Q-learning algorithm. Specifically, the proposed scheme is at least 56.5% better than the other three schemes when the bandwidth is 50MHz, which proves the superior performance of the proposed scheme.

VI. Conclusions

In this paper, we studied the UAV-mounted MEC network, where some tasks with time-varying priority in several hotspots need to be executed by the UAV. For such a system, we analyzed the impact of the wireless bandwidth and computational capability of the UAV on the system performance, from which we optimize the system through UAV hotspot selection and user task offloading. We further employed the deep Q-learning algorithm to solve the system optimization problem to promote the system performance. Simulation results were finally verified the effectiveness of the proposed method that it can exploit the benefits of UAV hotspot selection and task offloading, and improve the system performance compared to the other schemes.

References

[1] H. Sun, F. Zhou, and R. Q. Hu, "Joint offloading and computation energy efficiency maximization in a mobile edge computing system," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 3052–3056, 2019.

[2] L. Xiao, X. Lu, T. Xu, X. Wan, W. Ji, and Y. Zhang, "Reinforcement learning-based mobile offloading for edge computing against jamming and interference," *IEEE Trans. Commun.*, vol. 68, no. 10, pp. 6114–6126, 2020.

[3] L. Chen and X. Lei, "Relay-assisted federated edge learning: Performance analysis and system optimization," *IEEE Trans. Commun.*, vol. PP, no. 99, pp. 1–12, 2022.

[4] W. Wu, F. Zhou, R. Q. Hu, and B. Wang, "Energy-efficient resource allocation for secure NOMA-enabled mobile edge computing networks," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 493–505, 2020.

[5] F. Zhou and R. Q. Hu, "Computation efficiency maximization in wireless-powered mobile edge computing networks," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 5, pp. 3170–3184, 2020.

[6] L. He and X. Tang, "Learning-based MIMO detection with dynamic spatial modulation," *IEEE Trans. Cog. Commun. and Net.*, vol. PP, no. 99, pp. 1–12, 2023.

[7] F. Wang, J. Xu, and S. Cui, "Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 4, pp. 2443–2459, 2020.

[8] Y. Guo, R. Zhao, S. Lai, L. Fan, X. Lei, and G. K. Karagiannidis, "Distributed machine learning for multiuser mobile edge computing systems," *IEEE J. Sel. Top. Signal Process.*, vol. 16, no. 3, pp. 460–473, 2022.

[9] Z. Liang, Y. Liu, T. Lok, and K. Huang, "Multi-cell mobile edge computing: Joint service migration and resource allocation," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 9, pp. 5898–5912, 2021.

[10] J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, "Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16 067–16 081, 2020.

[11] Q. Liu, L. Shi, L. Sun, J. Li, M. Ding, and F. Shu, "Path planning for UAV-mounted mobile edge computing with deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5723–5728, 2020.

[12] Y. Liu, Y. Li, Y. Niu, and D. Jin, "Joint optimization of path planning and resource allocation in mobile edge computing," *IEEE Trans. Mob. Comput.*, vol. 19, no. 9, pp. 2129–2144, 2020.

[13] R. Cong, Z. Zhao, G. Min, C. Feng, and Y. Jiang, "Edgego: A mobile resource-sharing framework for 6G edge computing in massive IoT systems," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14 521–14 529, 2022.