Dynamic Air-Ground Collaboration for Multi-Access Edge Computing

Shaoyu Wang*, Yang Huang*, and Bruno Clerckx†
* College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics
† Department of Electrical and Electronic Engineering, Imperial College London
Email: yang.huang.ceie@nuaa.edu.cn, b.clerckx@imperial.ac.uk

ICC 2022 - IEEE International Conference on Communications | 978-1-5386-8347-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICC45855.2022.9839258

***Abstract*—Unmanned aerial vehicles (UAVs) are expected to improve the quality of services for the fifth-generation (5G) and beyond networks. Nevertheless, in the context of multi-access edge computing (MEC), a key pillar for meeting 5G key performance indicators, state-of-the-art design suffers from difficulty in matching time/spatial-varying communication/computation demands with distributed resources in highly dynamic air-ground integrated networks. To handle this issue, this paper proposes a joint online trajectory planning and offloading scheduling scheme based on reinforcement learning (RL), such that a UAV and base stations in a multi-cell network can dynamically and collaboratively offer edge computing services. We formulate the decision-making on trajectory planning/offloading scheduling as mutually embedded Markov decision processes, so as to avoid exponentially increasing joint state/action spaces and non-cooperative decision-making. In order to learn the policies for decision-making, novel RL algorithms are proposed based on deep Q-network and the kernel method, respectively. It is shown that the learned policy is able to make the joint trajectory planning and offloading scheduling adaptive to dynamic computation demands. Benefiting from this, the air-ground collaborative MEC can significantly outperform the terrestrial-only MEC in terms of the average backlog of newly produced computational task bits.**

***Index Terms*—Unmanned aerial vehicle, multi-access edge computing, computation offloading, reinforcement learning.**

I. Introduction

Unmanned aerial vehicles (UAVs) are expected to be promising solutions to improve the quality of services (QoS) for terrestrial user equipment (UE) during crowded events or emergency [1]. Although projects such as 5G!Drones have been launched to investigate and validate integrating UAVs with the fifth-generation (5G) and beyond networks, multi-access edge computing (MEC), which is a key pillar for meeting 5G key performance indicators (KPIs), suffers from the problem of matching time/spatial-varying communication/computation demands with distributed communication/computation resources in three-dimensional environments [2].

In the context of MEC in air-ground integrated networks, state-of-the-art studies on offloading policy optimization mainly focused on the scenario where terrestrial devices can decide to execute computational tasks locally or offload them to UAVs [3]–[5]. Indeed, not performing MEC in collaboration with terrestrial networks, these studies cannot unlock such UAVs' full potential in improving QoS for terrestrial networks. On the other hand, [6] studied a scenario of air-ground collaborative MEC, where UEs in a single-cell cellular network can offload task bits not only to a UAV-mounted MEC server but also to a server deployed at a base station (BS). However, [6] formulates the design problem as a deterministic optimization, where the size of produced tasks and full channel state information are known prior to the optimization. Such assumptions and deterministic optimizations may not work well in the presence of time-varying but unpredictable computation demands and channel propagation [2]. In contrast to the literature [3]–[6], considering highly dynamic environments, this paper investigates the air-ground collaborative MEC in a multi-cell network, where a UAV and each BS can provide UEs with edge computing services. Due to the highly dynamic environments, channel gains and statistical characteristics of the computational task production are unknown to the decision-making for offloading scheduling and online trajectory planning. Furthermore, due to the continuous task production in a whole timeslot and the non-negligible time of signaling and data preparation for transmission, decision-making in a timeslot has no knowledge of the number of task bits produced in the timeslot, and the offloading scheduling decisions for processing/offloading these task bits can only be executed in the next timeslot [7].

Learning policies for offloading scheduling and online trajectory planning turns out to be a sequential decision-making problem. Formulating the problem as Markov decision process (MDP) [8] with a single agent can suffer from the curse of dimensionality, where joint state/action spaces grow exponentially with the number of devices in the network [9]. However, by applying a non-cooperative multi-agent model where agents independently make decisions [10], the action of offloading task bits to the UAV may not benefit the overall backlog of task bits, as the agent with respect to (w.r.t.) UE offloading could not observe the UAV's state. Therefore, we formulate the MDP w.r.t. the UAV (referred to as MDP-UAV) and that w.r.t. each cell in a mutually embedded manner. By this means, decision-making on the trajectory planning at the agent w.r.t. the UAV (referred to as agent-UAV) and that on the offloading scheduling at each agent w.r.t. a cell can respectively utilize each other's states on the size of produced tasks in each cell and the UAV's position. In order to solve the formulated mutually embedded MDP problems and learn

This work was supported in part by the National Natural Science Foundation of China under Grant 61901216 and U2001210, the Natural Science Foundation of Jiangsu Province under Grant BK20190400, and Key R&D Plan of Jiangsu Province under Grant BE2021013-4. (Corresponding author: Yang Huang)

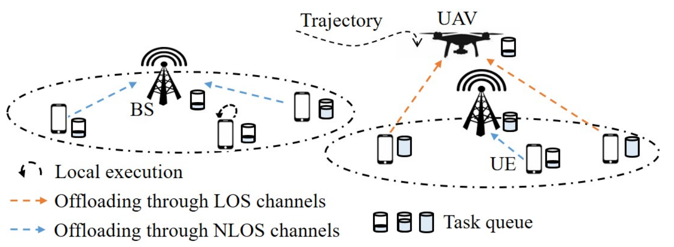
Trajectory
UAV
BS
UE
Local execution
Offloading through LOS channels
Offloading through NLOS channels
Task queue

Fig. 1. Terrestrial network devices collaborated with a UAV.

the policies, we propose novel reinforcement learning (RL) algorithms based on deep Q-network (DQN) [11] and the kernel method [8], respectively. Different from the former, the latter estimates action-values with linearly combined kernel functions, where the weight vectors are learned by exploiting the semi-gradient temporal difference (TD) method [8] and the features for decision-making are updated by performing approximate linear dependence (ALD) tests [12]. Simulation results reveal that following the learned policy, the joint trajectory planning and offloading scheduling can be adaptive to the time-varying computation demands. Benefiting from this, the dynamic air-ground collaboration can significantly outperform the terrestrial-only MEC, yielding lower average backlogs of newly produced task bits. Additionally, the average decision time of the kernel-based approach can be much less than that of the DQN-based approach.

Organization: Section II elaborates on the system model. Section III formulates the design problem. Section IV proposes the joint trajectory planning and offloading scheduling algorithms. Section V discusses the simulation results. Conclusions are drawn in Section VI. *Notations*: Matrices and vectors are respectively in bold capital and bold lower cases; $(\cdot)^T$, $\|\cdot\|$ and $|\cdot|$ represent the transpose, 2-norm, absolute value, respectively; $|\mathcal{A}|$ returns the cardinality of the set $\mathcal{A}$; $\text{cat}(\mathbf{a}, \mathbf{b})$ concatenates $\mathbf{b}$ vertically to the end of $\mathbf{a}$.

II. System Model

A. Multi-cell Network & Offloading

In the studied multi-cell network, the UAV flies at a constant altitude H and serves as a flying edge computing node, as shown in Fig. 1. The terrestrial network consists of J cells, where each BS j (for $j \in \{1, \ldots, J\}$) is exclusively associated with M_j stationary UEs for which the indices are collected in a set $\mathcal{M}_j$ and $\mathcal{M}_j \bigcap \mathcal{M}_{j'} = \emptyset$. The BS j and all the UE $m \in \mathcal{M}_j$ constitute cell j. Moreover, $\mathcal{M} \triangleq \cup_{j=1}^{J} \mathcal{M}_j$.

Due to the terrestrial non-line-of-sight (NLOS) small-scale fading, this paper assumes a block-fading channel model. That is, terrestrial channel gains remain constant within a timeslot but vary across timeslots. The small-scale fading in timeslot t between a certain BS j and a UE $m \in \mathcal{M}_j$ can be designated as $h_{0,j,m,t}$, and the corresponding channel power gain $|h_{j,m,t}|^2 = |h_{0,j,m,t}|^2 d_{j,m}^{-\beta}$, where β and $d_{j,m}$ stand for the pathloss exponent and the distance between the BS and the UE, respectively. In timeslot t, given a certain stationary UE m's position $\mathbf{q}_m$ (which is a row vector) and a certain position $\mathbf{q}_{\text{UAV},t} = [x_t, y_t, H]$ of the UAV, the distance between the UAV and the UE can be obtained as $d_{0,m,t} = |\mathbf{q}_{\text{UAV},t} - \mathbf{q}_m|$. Therefore, due to the LoS channel between the UAV and a certain UE, the channel power gain between the UAV and UE m can be obtained as $|h_{0,m,t}|^2 = |h_0|^2 d_{0,m,t}^{-2}$, where $|h_0|^2$ represents the channel power gain at a reference distance of 1m. In each timeslot t, the UAV flies towards a direction $a_{0,t} \in \mathcal{A}_0$ over a constant distance of ν_0, where $\mathcal{A}_0$ represents a set collecting flight directions.

In each timeslot t, for each UE, the offloading scheduling options, which include executing computational tasks locally at a UE and offloading tasks to the UAV or the associated BS j, are mutually exclusive. The duration of offloading or/and executing tasks can be designated as τ. Assuming adequate number of frequency-domain channels, the UEs' offloading transmissions do not interfere with each other, and the computation results can be returned to the UEs via dedicated frequency-domain channels. Therefore, the achievable rate at BS j or the UAV in timeslot t can be obtained as

$$R_{X,m,t} = B \log_2 \left(1 + \frac{P_m |h_{X,m,t}|^2}{\sigma_n^2}\right), \tag{1}$$

where B, P_m and σ_n^2 respectively represent the uplink channel bandwidth, transmit power at UE m and the average noise power. Note that the value of the subscript X in (1) depends on the MEC server being deployed at the UAV or BS j. In the former case, $X = 0$; otherwise, $X = j \in \{1, \ldots, J\}$.

B. Computational Task Production, Buffering and Execution

We consider that each UE continuously produces computational tasks over timeslots, and the statistical characteristics of task production is unknown to the network. The number of task bits produced by UE $m \in \mathcal{M}_j$ during timeslot $t-1$ can be designated as $L_{j,m,t-1}$. Due to the overhead of signaling and data preparation [7], these $L_{j,m,t-1}$ task bits can only be processed locally at the UE or offloaded in timeslot t, following the decision made in timeslot $t-1$.[1] Nevertheless, due to the continuous data production within the entire timeslot, decision-making on offloading scheduling in timeslot $t-1$ is unable to gain knowledge of $L_{j,m,t-1}$. The CPU cycle frequency of each UE and that of an MEC server deployed at a BS/UAV can be designated as f_{local} and f_{MEC}, respectively. The processing density which means the number of CPU cycles required to process a task bit can be designated as c [4]. Moreover, each device of the UEs, the BSs and the UAV is equipped with a task queue to buffer unprocessed task bits, following the first-in-first-out (FIFO) rule. In order to evaluate the backlog of the task bits, we define a variable $L_{\text{BL},j,m,t-1}$ which is equal to the number of unprocessed task bits observed at the end of timeslot t but produced at UE $m \in \mathcal{M}_j$ in timeslot $t-1$. Note that, $L_{\text{BL},j,m,t-1}$ can reflect the timeliness of task processing. That

[1] In practice, an offloading scheduling decision is generated in timeslot $t-1$ by the BS and then distributed to the UEs. Each UE can obtain information of the number $L_{j,m,t-1}$ of task bits and execute the decision in timslot t.

is, a smaller $L_{\mathrm{BL},j,m,t-1}$ means that there are less bits to be processed in the future timeslots. Afterwards, the $L_{\mathrm{BL},j,m,t-1}$ observed in timeslot t is analyzed in the local execution (at the UE) and the offloading scenarios respectively.

In the presence of UE m executing computational tasks locally in timeslot t, prior to analyzing $L_{\mathrm{BL},j,m,t-1}$, we designate the number of unprocessed task bits, which are observed at the end of timeslot $t-1$ but produced before timeslot $t-1$, as $D_{j,m,t-1}$. Note that when processing the buffered task bits in timeslot t, these $D_{j,m,t-1}$ bits take priority due to the FIFO rule. Then, the time $\Delta t_{j,m,t}$ for processing all the buffered bits can be obtained as $\Delta t_{j,m,t} = c\cdot(L_{j,m,t-1}+D_{j,m,t-1})/f_{\mathrm{local}}$. If $\Delta t_{j,m,t} < \tau$, $D_{j,m,t} = 0$ and $L_{\mathrm{BL},j,m,t-1} = 0$. On the contrary, for $\Delta t_{j,m,t} \geq \tau$, $D_{j,m,t} = D_{j,m,t-1} + L_{j,m,t-1} - \tau f_{\mathrm{local}}/c$. In this case, if $D_{j,m,t-1} < \tau f_{\mathrm{local}}/c$, $L_{\mathrm{BL},j,m,t-1} = D_{j,m,t}$; otherwise, none of the $L_{j,m,t-1}$ bits can be processed, and $L_{\mathrm{BL},j,m,t-1} = L_{j,m,t-1}$.

In the presence of offloading in timeslot t, the task of $L_{j,m,t-1}$ bits can be transmitted from UE m to an MEC server at the UAV or the BS associated with the UE, where the transmission time $t_{\mathrm{trans}} = L_{j,m,t-1}/R_{X,m,t}$. If $t_{\mathrm{trans}} \geq \tau$, the MEC server omits the received bits, i.e. parts of the $L_{j,m,t-1}$ bits, while the $L_{j,m,t-1}$ bits are still reserved at the UE. In the meanwhile, the MEC server can process tasks which are buffered in its task queue during previous timeslots. The number of the unprocessed task bits, which are observed at the end of timeslot $t-1$ but buffered before timeslot $t-1$, can be designated as $D_{X,t-1}$, where the subscript $X = 0$ or $j \in \{1,\ldots,J\}$ similarly to the definition in (1). The time for processing the previously buffered task bits can be obtained as $t_{\mathrm{pre}} = D_{X,t-1}\cdot c/f_{\mathrm{MEC}}$. According to the relation between t_{trans} and t_{pre}, $L_{\mathrm{BL},j,m,t-1}$ is analyzed in two scenarios. In the scenario where $t_{\mathrm{trans}} > t_{\mathrm{pre}}$, all the $D_{X,t-1}$ bits can be processed before the end of offloading transmission. Hence, the total time $\Delta t_{\mathrm{MEC},m,t}$ for transmitting the $L_{j,m,t-1}$ bits and processing the offloaded bits at the MEC server can be obtained as $\Delta t_{\mathrm{MEC},m,t} = t_{\mathrm{trans}} + L_{j,m,t-1}\cdot c/f_{\mathrm{MEC}}$. If $\Delta t_{\mathrm{MEC},m,t} < \tau$, $D_{X,t} = L_{\mathrm{BL},j,m,t-1} = 0$; otherwise, $D_{X,t} = L_{\mathrm{BL},j,m,t-1} = L_{j,m,t-1} - f_{\mathrm{MEC}}(\tau - t_{\mathrm{trans}})/c$. However, in the scenario where $t_{\mathrm{trans}} \leq t_{\mathrm{pre}}$, the offloading completes before the end of processing the $D_{X,t-1}$ bits. Hence, the total time for offloading, waiting for processing the $D_{X,t-1}$ bits and processing the offloaded bits can be obtained as $\Delta t'_{\mathrm{MEC},m,t} = t_{\mathrm{pre}} + L_{j,m,t-1}\cdot c/f_{\mathrm{MEC}}$. If $\Delta t'_{\mathrm{MEC},m,t} < \tau$, $D_{X,t} = L_{\mathrm{BL},j,m,t-1} = 0$. On the contrary, for $\Delta t'_{\mathrm{MEC},m,t} \geq \tau$, $D_{X,t} = L_{j,m,t-1} + D_{X,t-1} - \tau f_{\mathrm{MEC}}/c$. In this case, if $D_{X,t-1} < \tau f_{\mathrm{MEC}}/c$, $L_{\mathrm{BL},j,m,t-1} = D_{X,t}$; otherwise, $L_{\mathrm{BL},j,m,t-1} = L_{j,m,t}$.

III. Mutually Embedded MDPs and Problem Formulation

Section II-B indicates that the design problem turns out to be a sequential decision-making. In order to address the curse of dimensionality [9], applying a non-cooperative multi-agent model [10] to this problem can be problematic. Specifically, assume that the agent w.r.t. cell j has no knowledge of the position of the UAV but makes decisions only based on the state of the backlog of task bits in the cell. If the agent visits a certain state several times and always chooses to offload task bits to the UAV, the immediate rewards indicating the backlog of task bits in the next timeslot can be quite diverse, due to the various positions of the UAV. The diverse rewards severely affect TD errors for estimating the action value of offloading task bits to the UAV, such that following the learned policy, a UE may not offload task bits to the UAV even if the UAV with adequate computation resource flies at an appropriate position.

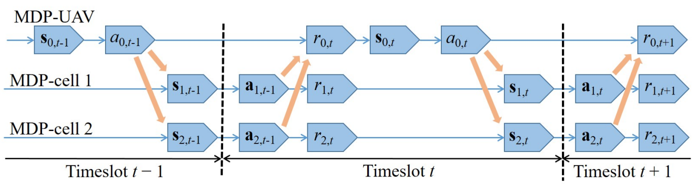

Fig. 2. An example of the mutually embedded MDPs.

Therefore, as depicted in Fig. 2, the formulated MDP-UAV and MDP w.r.t. cell j (referred to as MDP-cell j) are mutually embedded. That is, in timeslot $t-1$, a state $\mathbf{s}_{j,t-1}$ of the MDP-cell j contains the UAV's position and therefore can be partially affected by an executed action $a_{0,t}$ (i.e. the movement of UAV) of MDP-UAV. As the state $\mathbf{s}_{0,t}$ of the MDP-UAV in the next timeslot t contains information on the number of newly produced task bits in each cell j during timeslot $t-1$, the MDP-UAV is also partially affected by the MDP-cell j.

In the MDP-UAV, given the state space $\mathcal{S}_0$, the state $\mathbf{s}_{0,t} \in \mathcal{S}_0$ observed by agent-UAV in timeslot t can be defined as $\mathbf{s}_{0,t} \triangleq [\mathbf{q}_{\mathrm{UAV},t}, \mathbf{u}_{t-1}^T]^T$, where $\mathbf{u}_{t-1} = [U_{1,t-1},\ldots,U_{J,t-1}]^T$ and each entry $U_{j,t-1} = \sum_{m\in\mathcal{M}_j} L_{j,m,t-1}/M_j$, i.e. the average number of task bits produced in cell j during timeslot $t-1$. According to Section II-A, the direction selection action decided in timeslot t can be defined as $a_{0,t} \in \mathcal{A}_0$. The action $a_{0,t}$ is immediately executed in timeslot t, resulting in a new position, which remains constant until $a_{0,t+1}$ is taken in timeslot $t+1$ and therefore is designated as $\mathbf{q}_{\mathrm{UAV},t+1}$. Note that up to now, $\mathbf{s}_{0,t+1}$ has not been fully observed, as $L_{j,m,t}$ remains unknown until timeslot $t+1$. The immediate reward is defined as $r_{0,t+1} = -\sum_{j=1}^{J}\sum_{m\in\mathcal{M}_j} L_{\mathrm{BL},j,m,t}$, which can be affected by the task offloading/execution in timeslot $t+1$.

In the MDP-cell j, the state space and the action space can be designated as $\mathcal{S}_j$ and $\mathcal{A}_j$, respectively. The state observed by the agent w.r.t. cell j (which is then referred to as agent-cell j) in timeslot t can be designed as $\mathbf{s}_{j,t} = [\mathbf{q}_{\mathrm{UAV},t+1}, \mathbf{d}_{j,t}^T, \mathbf{u}_{t-1}^T]^T$, where $\mathbf{q}_{\mathrm{UAV},t+1}$ results from $a_{0,t}$; $\mathbf{d}_{j,t} = [D_{j,m_1,t},\ldots,D_{j,m_{M_j},t}, D_{j,t}, D_{0,t}]^T \in \mathbb{R}^{M_j+2}$, and the subscript m_i denotes the index of a UE, satisfying $\{m_i\}_{i=1}^{M_j} = \mathcal{M}_j$. Note that $\mathbf{u}_{t-1}$ is contained in $\mathbf{s}_{j,t}$ to avoid a greedy policy that always schedules the UAV to assist a particular cell. The offloading scheduling action w.r.t. UE m (for $m \in \mathcal{M}_j$) can be written as $\alpha_{j,m,t} \in \{\alpha_{\mathrm{L}}, \alpha_{\mathrm{D}}, \alpha_{\mathrm{B}}\}$, where α_{L}, α_{D} and α_{B} respectively mean executing the $L_{j,m,t-1}$ bits locally at the UE, offloading the $L_{j,m,t-1}$ bits to the UAV and offloading the $L_{j,m,t-1}$ bits to BS j. Then, the offloading scheduling action of MDP-cell j decided in timeslot t can be obtained as $\mathbf{a}_{j,t} = [\alpha_{j,m_1,t}, \ldots, \alpha_{j,m_{M_j},t}]^T \in \mathcal{A}_j$, where the subscripts m_i satisfy $\{m_i\}_{i=1}^{M_j} = \mathcal{M}_j$. Note that $\mathbf{a}_{j,t}$ is executed in timeslot $t+1$. The immediate reward in timeslot $t+1$ can be obtained as $r_{j,t+1} = -\sum_{j=1}^{J}\sum_{m\in\mathcal{M}_j} L_{\text{BL},j,m,t} - \sum_{m\in\mathcal{M}_j} D_{j,m,t+1} - D_{j,t+1} - D_{0,t+1}$.

The optimizations w.r.t. the MDP-UAV and the MDP-cell j $\forall j$ can be respectively formulated as

$$\max_{\pi_0} \mathbb{E}\left\{\sum_{t=1}^{\infty} \gamma^{t-1} r_{0,t}\right\} \text{ and } \max_{\pi_j} \mathbb{E}\left\{\sum_{t=1}^{\infty} \gamma^{t-1} r_{j,t}\right\}, \tag{2}$$

where $\pi_0 : \mathcal{S}_0 \rightarrow \mathcal{A}_0$ and $\pi_j : \mathcal{S}_j \rightarrow \mathcal{A}_j$ represent deterministic policies w.r.t. the MDP-UAV and the MDP-cell j, and $\gamma \in (0,1)$ is a discount factor.

IV. Joint Trajectory Planning and Offloading Scheduling Algorithms

The modelling in Section II-B indicates that the dynamics of $P(\mathbf{s}_{0,t+1} = \mathbf{s}_0' | \mathbf{s}_{0,t} = \mathbf{s}_0, a_{0,t} = a_0)$ for $\mathbf{s}_0, \mathbf{s}_0' \in \mathcal{S}_0$ and $a_0 \in \mathcal{A}_0$ and $P(\mathbf{s}_{j,t+1} = \mathbf{s}_j' | \mathbf{s}_{j,t} = \mathbf{s}_j, \mathbf{a}_{j,t} = \mathbf{a}_j)$ for $\mathbf{s}_j, \mathbf{s}_j' \in \mathcal{S}_j$ and $\mathbf{a}_j \in \mathcal{A}_0$ are unknown to agent-UAV or agent-cell j. Therefore, solving problem (2) requires an RL approach. Nevertheless, a multi-objective RL [13] is inapplicable, as $\mathbf{s}_{j,t}$ of the MDP-cell j is affected by $a_{0,t}$ of the MDP-UAV.

A. DQN-Based Approach

This section proposes a DQN-based approach, where estimates of the action-values w.r.t. the agent-UAV and the agent-cell j, i.e. $Q(\mathbf{s}_0, a_0; \mathbf{w}_0)$ and $Q(\mathbf{s}_j, \mathbf{a}_j; \mathbf{w}_j)$, are approximated by deep neural networks for which the parameters $\mathbf{w}_0$ and $\mathbf{w}_j$ are iteratively learned.

In timeslot t, the agent-UAV exploits the ϵ-greedy strategy to select an action: randomly select $a_{0,t} \in \mathcal{A}_0$, with probability ϵ; otherwise, $a_{0,t} = \arg\max_{a_0} Q(\mathbf{s}_{0,t}, a_0; \mathbf{w}_0)$. The UAV then immediately flies towards a direction specified by $a_{0,t}$ and arrives at a new position $\mathbf{q}_{\text{UAV},t+1}$, while the agent-cell j observes $\mathbf{s}_{j,t} = [\mathbf{q}_{\text{UAV},t+1}, \mathbf{d}_{j,t}^T, \mathbf{u}_{t-1}^T]^T$. In terms of the MDP-cell j, although π_j converges over iterations, randomly exploring the offloading scheduling action space as in the ϵ-greedy strategy may always select strictly suboptimal offloading scheduling actions with a certain probability. This can cause dramatic fluctuations in the long-term average backlog of task bits. Therefore, in contrast to the conventional ϵ-greedy strategy, given a state, only the offloading scheduling actions that have not been visited before should be explored. To this end, $\mathbf{s}_{j,t}$ is quantized as $\tilde{\mathbf{s}}_{j,t} = [\tilde{\mathbf{q}}_{\text{UAV},t+1}, \tilde{\mathbf{d}}_{j,t}^T, \tilde{\mathbf{u}}_{t-1}^T]^T$, and therefore the corresponding state space can be designated as $\tilde{\mathcal{S}}_j$. For each MDP-cell j, a matrix $\mathbf{T}_j$ can be defined to record whether the state-action pairs are visited. If the pair of the mth quantized state and the nth action is visited, $[\mathbf{T}_j]_{m,n} = 1$; otherwise, it is equal to 0. Given thresholds μ_q, μ_d and μ_u, for all $\tilde{\mathbf{s}}_j = [\tilde{\mathbf{q}}_{\text{UAV}}, \tilde{\mathbf{d}}_j^T, \tilde{\mathbf{u}}^T]^T$ belonging to $\tilde{\mathcal{S}}_j$, if any one of $\|\tilde{\mathbf{q}}_{\text{UAV},t+1} - \tilde{\mathbf{q}}_{\text{UAV}}\| > \mu_q$, $\|\tilde{\mathbf{d}}_{j,t} - \tilde{\mathbf{d}}_j\| > \mu_d$ and $\|\tilde{\mathbf{u}}_{t-1} - \tilde{\mathbf{u}}\| > \mu_u$ is satisfied, $\tilde{\mathbf{s}}_{j,t}$ is recognized as a new quantized state i.e. $\tilde{\mathbf{s}}_{j,t} \notin \tilde{\mathcal{S}}_j$. In this case, $\tilde{\mathcal{S}}_j = \tilde{\mathcal{S}}_j \cup \tilde{\mathbf{s}}_{j,t}$, and $\mathbf{T}_j = \text{cat}(\mathbf{T}_j, \mathbf{0}_{1\times|\mathcal{A}_j|})$. Then, given $\tilde{\mathbf{s}}_{j,t}$ (which corresponds to the row index m of $[\mathbf{T}_j]_{m,n}$), with probability ϵ, $\mathbf{a}_{j,t}$ is randomly selected from the set of actions for which $[\mathbf{T}_j]_{m,n}$ is equal to 0; otherwise, $\mathbf{a}_{j,t} = \arg\max_{\mathbf{a}_j} Q(\mathbf{s}_{j,t}, \mathbf{a}_j; \mathbf{w}_j)$.

Similarly to the DQN approach [11], in order to minimize the expected mean-squared errors between the true action-values and the estimated action-values $Q(\mathbf{s}_0, a_0; \mathbf{w}_0)$ and $Q(\mathbf{s}_j, \mathbf{a}_j; \mathbf{w}_j)$, as well as stabilize the learning of $\mathbf{w}_0$ and $\mathbf{w}_j$, stochastic optimization and experience replay are exploited. The replay memories w.r.t. the MDP-UAV and the MDP-cell j are denoted as $\mathcal{G}_0$ and $\mathcal{G}_j$, respectively. Thus, in each iteration of the proposed algorithm, a minibatch of N transition samples is randomly taken from $\mathcal{G}_0$, as well as $\mathcal{G}_j$, for optimizing $\mathbf{w}_0$ and $\mathbf{w}_j$. The optimizations can be respectively formulated as

$$\mathbf{w}_0' = \arg\min_{\mathbf{w}_0} \frac{1}{N}\sum_{k=1}^{N} |y_{\text{UAV},k} - Q(\mathbf{s}_{0,k}, a_{0,k}; \mathbf{w}_0)|^2 \tag{3}$$

and

$$\mathbf{w}_j' = \arg\min_{\mathbf{w}_j} \frac{1}{N}\sum_{k=1}^{N} |y_{j,k} - Q(\mathbf{s}_{j,k}, \mathbf{a}_{j,k}; \mathbf{w}_j)|^2 , \tag{4}$$

where $y_{\text{UAV},k} = r_{0,k+1} + \gamma \max_{a_0} Q(\mathbf{s}_{0,k+1}, a_0; \mathbf{w}_0^-)$ and $y_{j,k} = r_{j,k+1} + \gamma \max_{\mathbf{a}_j} Q(\mathbf{s}_{j,k+1}, \mathbf{a}_j; \mathbf{w}_j^-)$ represent TD targets w.r.t. the estimates of $Q(\mathbf{s}_0, a_0; \mathbf{w}_0)$ and $Q(\mathbf{s}_j, \mathbf{a}_j; \mathbf{w}_j)$, respectively. Problems (3) and (4) are iteratively solved by exploiting Adam optimizer [11]. The proposed DQN-based algorithm is summarized in Algorithm 1.

B. Kernel-Based Approach

The steps in the proposed kernel-based approach are similar to those in the DQN-based approach. In the kernel-based approach, the decision-making processes of the agents, executing actions, obtaining rewards and observing states are identical to the pseudocodes from step 3 to step 14 of Algorithm 1, except the processes of adding transition samples to $\mathcal{G}_0$ and $\mathcal{G}_j$ in steps 7 and 14 due to the absence of experience reply.

Nevertheless, in contrast to the DQN-based approach, in the kernel-based approach $Q(\mathbf{s}_0, a_0; \mathbf{w}_0) = \mathbf{w}_0^T \mathbf{f}_{0,t}(\mathbf{s}_0, a_0)$ and $Q(\mathbf{s}_j, \mathbf{a}_j; \mathbf{w}_j) = \mathbf{w}_j^T \mathbf{f}_{j,t}(\mathbf{s}_j, \mathbf{a}_j)$, where $\mathbf{w}_0$ and $\mathbf{w}_j$ represent weight vectors; $\mathbf{f}_{0,t}$ and $\mathbf{f}_{j,t}$ are kernel vectors with $N_{0,t}$ and $N_{j,t}$ entries, respectively. This paper employs the Gaussian kernel i.e. $f(\mathbf{x}, \mathbf{x}') = \exp(-(\mathbf{x} - \mathbf{x}')^T \Sigma^{-1} (\mathbf{x} - \mathbf{x}')/2)$. Furthermore, by designating the feature space mapping as $\phi(\cdot)$, we have $f(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$, according to Mercer's theorem [12]. Each entry of $\mathbf{f}_{0,t}$ can be written as $[\mathbf{f}_{0,t}]_n = f(\mathbf{x}_0, \hat{\mathbf{x}}_{0,n}) = \exp(-\|\mathbf{s}_0 - \hat{\mathbf{s}}_{0,n}\|^2/2\sigma_{\text{s}}^2)\exp(-\|a_0 - \hat{a}_{0,n}\|^2/2\sigma_{\text{a}}^2)$ for $n = 1, \ldots, N_{0,t}$, where $\mathbf{x}_0 \triangleq [\mathbf{s}_0^T, a_0]^T$ and $\hat{\mathbf{x}}_{0,n} \triangleq [\hat{\mathbf{s}}_{0,n}^T, \hat{a}_{0,n}]^T$ represent the sample and the feature w.r.t. the decision-making at the agent-UAV. Similarly, each entry of $\mathbf{f}_{j,t}$ can be written as $[\mathbf{f}_{j,t}]_n = f(\mathbf{x}_j, \hat{\mathbf{x}}_{j,n}) = \exp(-\|\mathbf{s}_j - \hat{\mathbf{s}}_{j,n}\|^2/2\sigma_{\text{s}}^2)\exp(-\|\mathbf{a}_j - \hat{\mathbf{a}}_{j,n}\|^2/2\sigma_{\text{a}}^2)$ for $n = 1, \ldots, N_{j,t}$, where $\mathbf{x}_j \triangleq [\mathbf{s}_j^T, \mathbf{a}_j]^T$ and $\hat{\mathbf{x}}_{j,n} \triangleq [\hat{\mathbf{s}}_{j,n}^T, \hat{\mathbf{a}}_{j,n}]^T$. All the features of $\hat{\mathbf{x}}_{0,n}$ and $\hat{\mathbf{x}}_{j,n}$ are respectively collected in dictionaries $\mathcal{D}_{0,t} \triangleq \{\hat{\mathbf{x}}_{0,n}\}_{n=1}^{N_{0,t}}$ and $\mathcal{D}_{j,t} \triangleq \{\hat{\mathbf{x}}_{j,n}\}_{n=1}^{N_{j,t}} \forall j$.

Algorithm 1 DQN-based Approach

```
1:  Initialize: For $j = 1\ldots, J$, initialize $\mathbf{s}_{j,0}, \tilde{\mathbf{s}}_{j,0}, \mathbf{a}_{j,0}, r_{j,1}, \mathbf{w}_j$; set
    $\tilde{\mathcal{S}}_j = \tilde{\mathbf{s}}_{j,0}$, $\mathbf{T}_j = \mathbf{0}_{1\times|\mathcal{A}_j|} \forall j$; set $t = 1$; initialize $\mathbf{s}_{0,t}$ and $\mathbf{w}_0$.
2:  repeat
3:      In timeslot $t$, the agent-UAV observes $\mathbf{s}_{0,t}$ and selects $a_{0,t}$
        by performing the $\epsilon$-greedy strategy:
        $$a_{0,t} = \begin{cases} \text{randomly select } a_0 \in \mathcal{A}_0, & \text{with probability } \epsilon \\ \arg\max_{a_0} Q(\mathbf{s}_{0,t}, a_0; \mathbf{w}_0), & \text{otherwise} \end{cases}$$
4:      The agent-UAV executes $a_{0,t}$, yielding $\mathbf{q}_{\mathrm{UAV},t+1}$;
5:      for $j = 1, \ldots, J$ do
6:          Each agent-cell $j$ observes $\mathbf{s}_{j,t}$ and obtains $\tilde{\mathbf{s}}_{j,t}$;
7:          Add the transition sample $(\mathbf{s}_{j,t-1}, \mathbf{a}_{j,t-1}, r_{j,t}, \mathbf{s}_{j,t})$ into $\mathcal{G}_j$;
8:          if $\tilde{\mathbf{s}}_{j,t} \notin \tilde{\mathcal{S}}_j$ then
9:              $\tilde{\mathcal{S}}_j = \tilde{\mathcal{S}}_j \cup \tilde{\mathbf{s}}_{j,t}$ and $\mathbf{T}_j = \mathrm{cat}(\mathbf{T}_j, \mathbf{0}_{1\times|\mathcal{A}_j|})$;
10:         end if
11:         Find the row index $m$ w.r.t. $\tilde{\mathbf{s}}_{j,t}$ in $\mathbf{T}_j$ and obtain $\mathcal{A}_{j,m} =$
            $\{\mathbf{a}_j | \text{all } \mathbf{a}_j \in \mathbf{A}_j \text{ for which } [\mathbf{T}_j]_{m,n} = 0\}$; select $\mathbf{a}_{j,t}$ via
            $$\text{Select } \mathbf{a}_{j,t} = \begin{cases} \text{randomly select } \mathbf{a}_{j,t} \in \mathcal{A}_{j,m}, & \text{with probability } \epsilon \\ \arg\max_{\mathbf{a}_j} Q(\mathbf{s}_{j,t}, \mathbf{a}_j; \mathbf{w}_j), & \text{otherwise} \end{cases}$$
12:         Set $[\mathbf{T}_j]_{m,n}$ corresponding to $(\tilde{\mathbf{s}}_{j,t}, \mathbf{a}_{j,t})$ as $[\mathbf{T}_j]_{m,n} = 1$;
13:     end for
14:     In timeslot $t+1$, for $j = 1\ldots, J$, each agent-cell $j$
        executes $\mathbf{a}_{j,t}$ and obtains $r_{j,t+1}$; The agent-UAV obtains the
        reward $r_{0,t+1}$, observes $\mathbf{s}_{0,t+1}$ and add the transition sample
        $(\mathbf{s}_{0,t}, a_{0,t}, r_{0,t+1}, \mathbf{s}_{0,t+1})$ to $\mathcal{G}_0$;
15:     The agent-UAV samples a random minibatch of $N$ transition
        samples $(\mathbf{s}_{0,k}, a_{0,k}, r_{0,k+1}, \mathbf{s}_{0,k+1})$ from $\mathcal{G}_0$ and exploits Adam
        optimizer to solve problem (3), yielding an updated $\mathbf{w}_0'$;
16:     Each agent-cell $j$ samples a random minibatch of $N$ transition
        samples $(\mathbf{s}_{j,k}, \mathbf{a}_{j,k}, r_{j,k+1}, \mathbf{s}_{j,k+1})$ from $\mathcal{G}_j$ and exploits the
        Adam algorithm to solve problem (4), yielding an updated $\mathbf{w}_j'$;
        reset $\mathbf{w}_0^- = \mathbf{w}_0'$ and $\mathbf{w}_j^- = \mathbf{w}_j'$ for every $T_c$ timeslots;
17:     $\mathbf{w}_0 \leftarrow \mathbf{w}_0'$ and $\mathbf{w}_j \leftarrow \mathbf{w}_j'$;
18:     $t \leftarrow t+1$;
19: until Stopping criteria
```

Once the agent-UAV observing the state $\mathbf{s}_{0,t+1}$ and obtaining the reward $r_{0,t+1}$, the agents learn and update the weight vectors $\mathbf{w}_0$ and $\mathbf{w}_j$, by exploiting the semi-gradient TD method [8]. Therefore, the computation of $\mathbf{w}_0'$ in step 15 of Algorithm 1 is replaced by $\mathbf{w}_0' = \mathbf{w}_0 + \alpha\big(r_{0,t+1} + \gamma \max_{a_0}\{\mathbf{w}_0^T \mathbf{f}_{0,t}(\mathbf{s}_{0,t+1}, a_0)\} - \mathbf{w}_0^T \mathbf{f}_{0,t}(\mathbf{s}_{0,t}, a_{0,t})\big)\mathbf{f}_{0,t}(\mathbf{s}_{0,t}, a_{0,t})$, where α stands for a positive step size. Although the agent-cell j has obtained $r_{j,t+1}$ up to now, $\mathbf{s}_{j,t+1}$ remains unknown as $a_{0,t+1}$ has not been decided or executed by the agent-UAV. Therefore, the most recent unbroken transition sample that can be utilized by the agent-cell j is $(\mathbf{s}_{j,t-1}, \mathbf{a}_{j,t-1}, r_{j,t}, \mathbf{s}_{j,t})$, and the update can be obtained as $\mathbf{w}_j' = \mathbf{w}_j + \alpha\big(r_{j,t} + \gamma \max_{\mathbf{a}_j}\{\mathbf{w}_j^T \mathbf{f}_{j,t}(\mathbf{s}_t, \mathbf{a}_j)\} - \mathbf{w}_j^T \mathbf{f}_{j,t}(\mathbf{s}_{j,t-1}, \mathbf{a}_{j,t-1})\big)\mathbf{f}_{j,t}(\mathbf{s}_{j,t-1}, \mathbf{a}_{j,t-1})$. We then substitute step 16 of Algorithm 1 with the above equation.

Besides the update of $\mathbf{w}_0$ and $\mathbf{w}_j$, adding more features to the dictionaries of $\mathcal{D}_{0,t}$ and $\mathcal{D}_{j,t}$ can further improve the accuracy of the estimates of action-values. Therefore, between steps 17 and 18, we insert a procedure where ALD tests [12] are performed to recognize new features. In order to check whether $\mathbf{x}_{0,t} \triangleq [\mathbf{s}_{0,t}^T, a_{0,t}]^T$ can be recognized as a new feature,

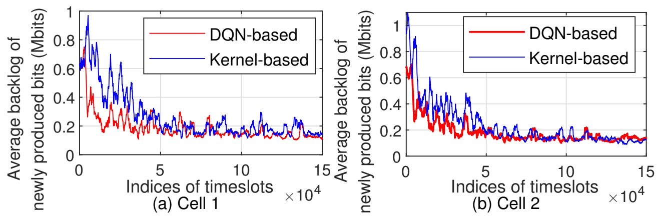

Fig. 3. Comparison of the average backlogs of newly produced task bits in each cell achieved by the DQN-based and the kernel-based approaches.

TABLE I
AVERAGE DECISION TIME OF THE PROPOSED APPROACHES.

Algorithms	DQN-based	Kernel-based
Average decision time [s]	0.157	0.011

we solve the problem: $\delta_{0,t} = \min_{\lambda_n \forall n} \| \sum_{n=1}^{N_{0,t}} \lambda_n \phi(\hat{\mathbf{x}}_{0,n}) - \phi(\mathbf{x}_{0,t})\|^2$. The above problem can be solved by utilizing the relation $f(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$ and the method in [12]. Given a threshold μ, if $\delta_{0,t} \leq \mu$ which means that $\phi(\mathbf{x}_{0,t})$ is approximately linearly dependent of $\{\phi(\hat{\mathbf{x}}_{0,n})\}_{n=1}^{N_{0,t}}$, $\mathcal{D}_{0,t+1} = \mathcal{D}_{0,t}$; otherwise, $\mathcal{D}_{0,t+1} = \mathcal{D}_{0,t} \cup \mathbf{x}_{0,t}$. The update of $\mathcal{D}_{j,t}$ is similar to that of $\mathcal{D}_{0,t}$, and the detailed discussions are omitted due to the space constraint.

V. Simulation Results

In the simulations, the UAV's flight direction set $\mathcal{A}_0$ contains eight cardinal directions, and the UAV's horizontal movement is restricted to a $1\,\mathrm{km}\times 1\,\mathrm{km}$ square, with $H = 100\,\mathrm{m}$ and $\nu_0 = 50\,\mathrm{m}$. Rayleigh fading is used to characterize the small-scale fading. The pathloss at a reference distance of 1 m is set as 42 dB. For terrestrial NLOS channels, the pathloss exponent $\beta = 2.8$. For the terrestrial network, $P_m = 30\,\mathrm{dBm}$ $\forall m$, $\sigma_n^2 = -60\,\mathrm{dBm}$ and $B = 6\,\mathrm{MHz}$. Moreover, $J = 2$ and $|\mathcal{M}_1| = |\mathcal{M}_2| = 3$, unless otherwise stated. In terms of computational task execution, $c = 10^3$ cycles/bit, $f_{\mathrm{MEC}} = 1.6\times 10^9$ cycles/s, $f_{\mathrm{local}} = 4\times 10^8$ cycles/s and $\tau = 2\,\mathrm{s}$. Task production in each cell is periodic as illustrated in Fig. 4(a). In the DQN-based approach (with Adam), each agent applies a fully connected neural network with 3 hidden layers, where each layer consists of 64 neurons. In the kernel-based approach, $\sigma_{\mathrm{s}} = 2$, $\sigma_{\mathrm{a}} = 1$ and $\mu = 0.82$. For both approaches, $\gamma = 0.5$. The simulations are conducted by MATLAB R2016a on a single computer, with an Intel Core i7 processor at 3.6GHz, a RAM of 16GB and the Windows 10 operating system.

Fig. 3 compares the proposed DQN-based and kernel-based approaches, in terms of the average backlog, referred to as $\bar{L}_{\mathrm{BL},j,t}$, of newly produced task bits in each cell, with $J = 2$ and $|\mathcal{M}_1| = |\mathcal{M}_2| = 1$. The average backlog $\bar{L}_{\mathrm{BL},j,t}$ observed in timeslot t is computed by averaging the overall backlog in cell j over 2000 most recent timeslots, i.e. $\bar{L}_{\mathrm{BL},j,t} = \sum_{t'=t-1999}^{t} \sum_{m\in\mathcal{M}_j} L_{\mathrm{BL},j,m,t'-1}/2000$. Fig. 3 indicates that the average backlogs $\bar{L}_{\mathrm{BL},j,t}$ for $j \in \{1,2\}$ achieved by both the approaches can converge to similar values. In Table I, the average decision time is evaluated by averaging the overall running time of the proposed algorithms over the elapsed

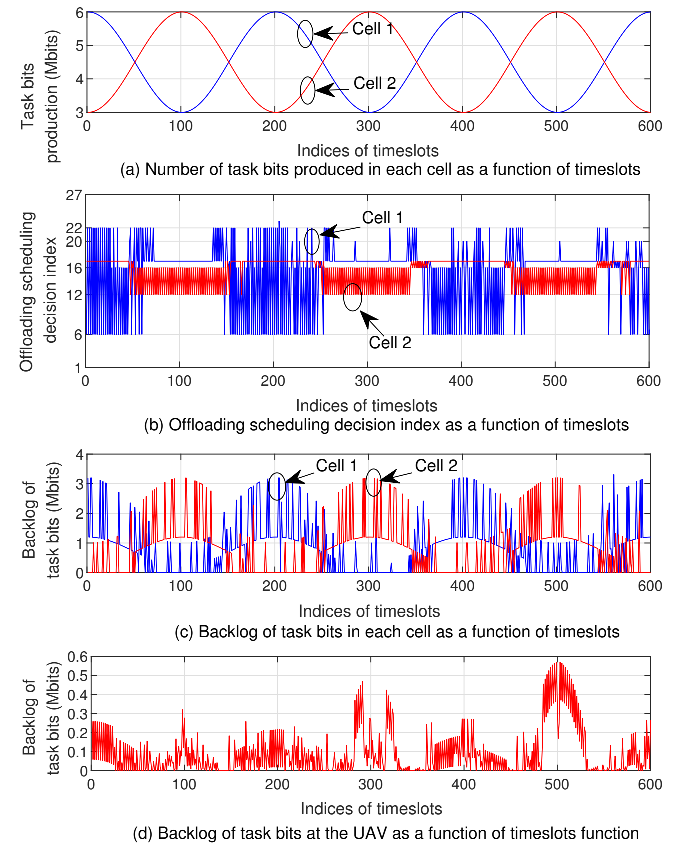

Fig. 4. The results are obtained through the kernel-based approach, which has already been running for 431,200 timeslots before the simulation. The number of task bits produced in cell j during timeslot t is equal to $\sum_{m\in\mathcal{M}_j} L_{j,m,t}$. The backlog in cell j and that at the UAV observed in timeslot t are equal to $D_{\text{cell},j} \triangleq \sum_{m\in\mathcal{M}_j} D_{j,m,t} + D_{j,t}$ and $D_{0,t}$, respectively.

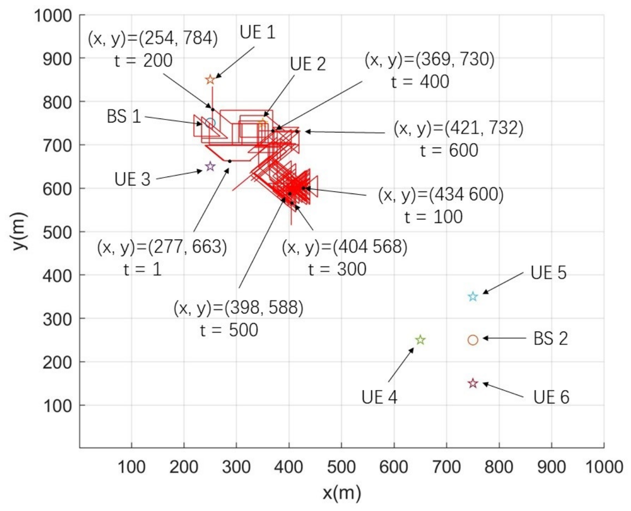

Fig. 5. The UAV's trajectory, where the notations (x, y) and t represent the horizontal position of the UAV and the index of a timeslot, respectively. Cell 1 consists of BS 1 and UEs 1 to 3; cell 2 consists of BS 2 and UEs 4 to 6.

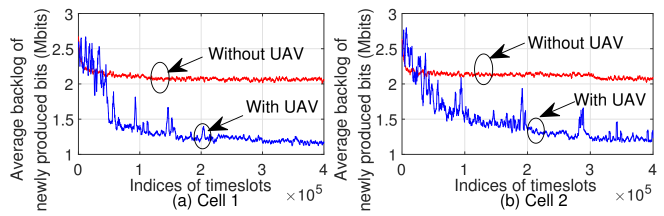

Fig. 6. Comparison of the average backlogs of newly produced task bits in each cell achieved by the proposed kernel-based approach and the MEC without the assistance of a UAV (i.e. terrestrial-only MEC).

timeslots. The kernel-based approach is shown to be much more computationally efficient than the DQN-based approach.

We then gain insights into the relation of the number of task bits produced during a timeslot, learned (or convergent) policies of π_0 and π_j $\forall j$ and the backlogs of task bits in the network, by analyzing Figs. 4 and 5. It can be seen from Fig. 4(a) and Fig. 5 that following the policy π_0, the UAV always flies towards and may hover over the cell where the value of $\sum_{m\in\mathcal{M}_j} L_{j,m,t}$ reaches a maximum of 6 Mbits. Remarkably, given the task production characteristics as shown in Fig. 4(a), the UAV does not fly round-trip from cell 1 to cell 2, as a decrease in $R_{0,m,t}$ can increase $\Delta t_{\text{MEC},m,t}$ and therefore degrade $\mathbb{E}\{\sum_{t=1}^{\infty}\gamma^{t-1} r_{0,t}\}$. For this reason, $D_{0,t}$ always suffers from a rapid growth in timeslots 100, 300 and 500, as depicted in Fig. 4(d). Moreover, Fig. 4(b) illustrates that when the value of $\sum_{m\in\mathcal{M}_1} L_{1,m,t}$ for cell 1 is greater than 4.5 Mbits, the offloading scheduling actions with indices of 6, 16, 20 and 22 (corresponding to an action of $\mathbf{a}_{1,t} = [\alpha_{\text{L}}, \alpha_{\text{D}}, \alpha_{\text{B}}]^T$ and its permutations) are alternatively executed in cell 1. Such decisions indicate that no more than one UE can simultaneously offload bits to BS 1; otherwise, $D_{1,t}$ can suffer from a much higher increase. This is due to the fact that the small-scale fading reduces $R_{1,m,t}$ and can cause a large $\Delta t_{\text{MEC},m,t}$ for offloading bits to the BS. Furthermore, the UE can alternate between executing tasks locally and offloading tasks to an MEC server, so as to reduce $D_{1,m,t}$. When $\sum_{m\in\mathcal{M}_2} L_{2,m,t} \geq 4.5$ Mbits, similar observations can be made in cell 2, while UE 6 does not offload bits to the UAV. On the contrary, in the presence of $\sum_{m\in\mathcal{M}_j} L_{j,m,t} < 4.5$ for a certain cell j, due to the small number of produced task bits, two UEs can simultaneously offload bits to the BS while the other UE offloads bits to the UAV, so as to maximize the objectives of (2). It can be drawn from Fig. 4 that the joint trajectory planning and offloading scheduling can be adaptive to the changes in the number of produced task bits, such that $D_{\text{cell},j}$ $\forall j$ can be always lower than 3.2 Mbits.

Fig. 6 compares the average backlogs $\bar{L}_{\text{BL},j,t}$ $\forall j$ achieved by MEC with and without the assistance of a UAV. The dynamic offloading scheduling algorithm for the terrestrial-only MEC is developed by omitting the steps regarding the UAV in the proposed kernel-based approach. Fig. 6 depicts that compared to the average backlog $\bar{L}_{\text{BL},j,t}$ achieved by the terrestrial-only MEC, there are more fluctuations in that achieved by the MEC with a UAV even if $\bar{L}_{\text{BL},j,t}$ tends to convergence, due to the additional state/action space introduced by the UAV. Despite this, the air-ground collaborative MEC can achieve significantly lower $\bar{L}_{\text{BL},j,t}$ $\forall j$ than the terrestrial-only MEC.

VI. Conclusions

We have investigated the joint online trajectory planning and offloading scheduling for MEC in an air-ground integrated multi-cell network, where a UAV and BSs can dynamically and collaboratively offer MEC services. The sequential decision-making process for trajectory planning and offloading scheduling are formulated as mutually embedded MDPs. Based on this, novel RL algorithms are respectively proposed based on DQN and the kernel method, where the latter is shown to be much more computationally efficient than the former. It is also shown that the learned policy successfully enables the joint trajectory planning and offloading scheduling to be adaptive to the time-varying computation demands. Benefiting from this, the air-ground collaborative MEC can significantly outperform the terrestrial-only MEC, in terms of the average backlog of newly produced bits.

References

[1] A. Liao, Z. Gao, D. Wang, H. Wang, H. Yin, D. W. K. Ng, and M.-S. Alouini, "Terahertz ultra-massive mimo-based aeronautical communications in space-air-ground integrated networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 6, pp. 1741–1767, 2021.

[2] Q. Wu, J. Xu, Y. Zeng, D. W. K. Ng, N. Al-Dhahir, R. Schober, and A. L. Swindlehurst, "A comprehensive overview on 5G-and-beyond networks with UAVs: From communications to sensing and intelligence," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 10, pp. 2912–2945, 2021.

[3] S. Zhu, L. Gui, D. Zhao, N. Cheng, Q. Zhang, and X. Lang, "Learning-based computation offloading approaches in UAVs-assisted edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 1, pp. 928–944, 2021.

[4] 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, 2019.

[5] 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. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 73–84, 2021.

[6] J. Xiong, H. Guo, J. Liu, N. Kato, and Y. Zhang, "Collaborative computation offloading at UAV-enhanced edge," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[7] E. Dahlman, S. Parkvall, and J. Skold, *4G, LTE-Advanced Pro and The Road to 5G, Third Edition*, 3rd ed. USA: Academic Press, Inc., 2016.

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

[9] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1226–1252, 2021.

[10] J. Heydari, V. Ganapathy, and M. Shah, "Dynamic task offloading in multi-agent mobile edge computing networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, and M. Bellemare et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[12] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2275–2285, 2004.

[13] Y. Huang, C. Hao, Y. Mao, and F. Zhou, "Dynamic resource configuration for low-power iot networks: A multi-objective reinforcement learning method," *IEEE Commun. Lett.*, vol. 25, no. 7, pp. 2285–2289, 2021.