Learning to Be Proactive: Self-Regulation of UAV Based Networks With UAV and User Dynamics

Ran Zhang, *Member, IEEE*, Miao Wang, *Member, IEEE*, Lin X. Cai, *Senior Member, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*— Multi-Unmanned Aerial Vehicle (UAV) control is one of the major research interests in UAV-based networks. Yet few existing works focus on how the network should optimally react when the UAV lineup and user distribution change. In this work, proactive self-regulation (PSR) of UAV-based networks is investigated when one or more UAVs are about to quit or join the network, with considering dynamic user distribution. We target at an optimal UAV trajectory control policy which proactively relocates the UAVs whenever the UAV lineup *is about to* change, rather than passively dispatches the UAVs *after* the change. Specifically, a deep reinforcement learning (DRL)-based self-regulation approach is developed to maximize the accumulated user satisfaction (US) score for a certain period within which at least one UAV will quit or join the network. To handle the changed dimension of the state-action space before and after the lineup changes, the state transition is deliberately designed. To accommodate continuous state and action space, an actor-critic based DRL, i.e., deep deterministic policy gradient (DDPG), is applied with better convergence stability. To effectively promote learning exploration around the timing of lineup change, an asynchronous parallel computing (APC) learning structure is proposed. Referred to as PSR-APC, the developed approach is then extended to the case of dynamic user distribution by incorporating time as one of the agent states. Finally, numerical results are presented to demonstrate the convergence and superiority of PSR-APC over a passive reaction method, and its capability in jointly handling the dynamics of both UAV lineup and user distribution.**

***Index Terms*— Unmanned aerial vehicle (UAV), machine learning, deep reinforcement learning (DRL), trajectory design, proactive self-regulation.**

I. Introduction

UNMANNED aerial vehicles (UAVs) have been demonstrating dazzling potentials in future wireless communications [2]. Compared to the terrestrial base stations (BSs), UAVs equipped with wireless transceivers stand out in providing highly on-demand services due to the flexible 3D mobility, better wireless connectivity due to higher chance of Line-of-Sight (LoS) links to ground users, and economical network deployment due to lower capital and operational cost [3]. As reported in [4], the UAV market is estimated at USD 19.3 billion in 2019 and is anticipated to reach USD 45.8 billion by 2025. In the booming market, UAVs have found practical applications in emergency rescue [5], network coverage enhancement and extension [6], crowd/traffic surveillance [7], cached content delivery [8], mobile edge computing [9], [10], etc.

While having a stunning future, UAV-based communications still face a number of research challenges in various aspects, ranging from resource allocation (RA), interference management, trajectory design, to energy management, computing offloading, etc., [11], [12]. Many related works have been conducted to tackle the challenges via different methodologies. Conventionally, the studied problem is mostly formulated into a (mixed integer) non-convex optimization problem. As such a problem is NP-hard to solve, the original problem is usually decoupled into sub-problems. Iterative algorithms are then exploited to achieve sub-optimal solutions with affordable computational complexity [9], [13]–[20]. While gaining satisfying performance in the existing works, the conventional methodology is a better fit for scenarios where the system parameters are considered to be fixed. When the system is dynamically changing due to the maneuverability of UAVs and user traffic dynamics and mobility, the decoupling-based iterative algorithms need to be re-executed every time the parameters are updated. As the network scale and heterogeneity of network entities are exponentially rising nowadays, it has been increasingly labored for the conventional approaches to handle the system dynamics.

Thanks to the recent advances in machine learning, reinforcement learning (RL) techniques are becoming promising solutions to UAV communication problems. Through actively interacting with the system environment and constantly learning from the experiences, RL is strongly capable of sequential decision making in time-varying environments with limited to even zero domain knowledge [21]. Its enhancement, deep RL (DRL), further settles the "dimension anxiety" caused by large or infinite state/action space, and significantly improves the convergence speed and stability [22]. With (D)RL, optimal policies can be derived to provide sequential-time control on UAVs (e.g., mobility, RA, user association, or task scheduling).

Manuscript received August 4, 2020; revised December 13, 2020; accepted February 8, 2021. Date of publication February 19, 2021; date of current version July 12, 2021. This work was supported in part by the National Science Foundation under Grant ECCS-1554576. This article was presented in part at the 2020 IEEE GLOBECOM. The associate editor coordinating the review of this article and approving it for publication was P. Li. *(Corresponding author: Ran Zhang.)*

Ran Zhang and Miao Wang are with the Department of Electrical and Computer Engineering, Miami University, Oxford, OH 45056 USA (e-mail: zhangr43@miamioh.edu; wangm64@miamioh.edu).

Lin X. Cai is with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL 60616 USA (e-mail: lincai@iit.edu).

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TWC.2021.3058533.

Digital Object Identifier 10.1109/TWC.2021.3058533

Any future changes in system parameters will be reflected in the experiences and accommodated via continued learning.

A. Related Works

Conventional optimization methods have been extensively applied to UAV communications. For instance, Zhang *et. al.* [23] consider UAVs as mobile sensors, and minimize the age of information by jointly optimizing the sensing/transmission time, UAV trajectory, and task scheduling. Zhang *et al.* [24] study UAVs as aerial users and minimize UAVs' mission completion time subject to the quality-of-connectivity constraints via trajectory design. When UAVs are used as mobile BSs, [6], [9], [13]–[16] consider single-UAV scenarios targeting different objectives. Specifically, Nasir *et al.* [6] study RA problem of a flying BS with adjustable altitude, and maximize the minimum user rate under non-orthogonal multiple access. Zeng *et al.* [13] and Wu and Zhang [14] further incorporate trajectory design of a single UAV, and minimize the mission completion time or maximize the minimum user throughput. Considering the energy-constrained nature of UAVs, Zeng *et al.* [15] investigate joint RA and trajectory design of a single UAV to minimize the UAV energy consumption. Li *et al.* [9] and Guo and Liu [16] study the computing offloading problem of a single UAV in mobile edge computing (MEC), and additionally consider task scheduling to optimize energy-relevant performances. When multiple UAVs are present, UAVs need to be coordinated in power control and user association/scheduling. Mozaffari *et al.* [17] study the joint UAV positioning, frequency planning, and user association problem to minimize the UAV-user latency. Wu *et al.* [18] take a step further by additionally considering UAV trajectory design, power control, and user scheduling. Mozaffari *et al.* [19] work on energy consumption as entry point and minimize the total propulsion energy of UAVs. Yang *et al.* [20] minimize UAV energy consumption in an MEC scenario and consider computation capacity as a new allocation factor.

Machine learning based approaches have been attracting increasing attention on UAV communications. In [25], Chen *et al.* study UAV-assisted mobile caching and employ Liquid State Machine Learning to predict users' content request distribution. RL-based approaches are more frequently exploited due to stronger adaptability to environment dynamics. For instance, Klaine *et al.* [26] proposed a distributed Q-learning (QL) approach to find the best UAV positions that maximize the total amount of covered users. Cui *et al.* [27] and Hu *et al.* [28] apply multi-agent QL to optimize RA and trajectory design, respectively. Liu *et al.* [29] develop a multi-agent QL framework to optimize the UAV trajectory and power control, considering ground user mobility predicted from an echo state network (ESN). To tackle the unafforded dimension problem when the state space is large, Deep QL (DQL) is exploited. By replacing the Q-matrix with a deep neural network [30], ESN [31], or deep belief network [32], RA, interference management, and trajectory design of UAVs can be optimized, respectively. To further accommodate large action space and expedite convergence, actor-critic (AC) based DRL is applied. Cheng *et al.* [33] propose an AC-based learning approach to optimize RA and task scheduling in UAV-assisted computing offloading. Khairy *et al.* [34] formulate the joint altitude control and channel access problem of a solar-powered UAV network into a constrained AC learning problem and apply Lagrangian primal-dual policy optimization to achieve convergence. Liu *et al.* [35] exploited the up-to-date AC variant, deep deterministic policy gradient (DDPG), to devise an optimal UAV control policy that jointly maximizes the energy efficiency, fairness and coverage performance of UAV networks.

B. Motivations and Contributions

The existing RL-based works on UAV communications mainly focus on control policy development given a fixed set of UAVs. Few works have investigated how the network should react when the UAV lineup is changed. On one hand, as the UAVs are battery powered, they are certain to drain out the battery at some point and have to quit the network. On the other hand, many UAV models are rechargeable from either the solar power [36] or the power grid. Thus new UAVs may join the network at any time. Therefore, it is quite practical and critical to investigate the optimal UAV regulation policy when the group of serving UAVs are changed. By regulation, only passive response after the UAV lineup change is not enough. Instead, the network is anticipated to identify or predict the upcoming change and proactively take early actions to minimize performance loss or maximize the performance improvement during the transition to the new optimal state. Such procedure is referred to as proactive self-regulation (PSR) in this paper. One major challenge of studying PSR with RL is that the dimensions of the state and action spaces will both change during the training process if the UAV lineup changes. Thus the state transition calls for deliberate design, and the exploration around the time of change needs to be skillfully promoted. In addition, most of the existing works only consider stationary user distributions. Though [29] and [37] involve user mobility in the RL framework, the UAV trajectories are limited to a mesh grid which may lead to considerable gaps from the optimal solution.

Driven by the above motivations, this article studies PSR of UAV-based networks given the dynamics of the UAV lineup and user distribution. We target at an optimal UAV control policy which proactively relocates the UAVs when i) at least a UAV is about to quit or join the network, or ii) user distribution is about to change, rather than passively adjust the serving UAVs after the change. To the best of our knowledge, this is the first work of optimal UAV trajectory control that jointly considers the dynamic UAV lineup and user distributions. Specifically, the contributions of the paper are summarized as follows.

- A DRL-based approach for PSR of UAV-based network is proposed. The approach is designed to maximize the accumulated user satisfaction (US) scores within a certain period given that at least one UAV will quit or join the network. To ensure that the UAV battery status and altitude are accurately recorded, and that the UAVs can move without locational limitations in the target area, the state-of-the-art AC learning method, DDPG [38],

is selected among all the DRL frameworks to make both state and action spaces continuous.
- An asynchronous parallel computing (APC) structure is proposed to better promote learning exploration around the time of change, so that the chance of converging to a sub-optimal policy is minimized. The proposed PSR approach under APC is referred to as PSR-APC.
- The PSR-APC approach is further extended by considering dynamic user distribution. Time is integrated as one of the states to achieve a time-dependent control policy.
- Numerical results are presented to demonstrate the convergence and efficacy of the proposed PSR-APC approach. Example UAV trajectories are also provided to illustrate the proactive reaction of UAVs to the lineup and user distribution changes.

The reminder of the paper is organized as follows. Section II describes the system model and formulates the problem. Section III introduces preliminary knowledge on DRL and the adopted DDPG algorithm. Section IV elaborates the detailed design of the proposed APC-PSR approach. Section V extends the APC-PSR from fixed user locations to dynamic user distributions. Numerical results are presented in Section VI. Finally, Section VII concludes the paper.

II. System Model and Problem Formulation

In this section, the system model is first elaborated from three aspects: network environment, spectrum access, and energy-related considerations, based on which the problem formulation is presented.

A. Network Environment

A target area $\mathbf{A}$ is considered with a set $\mathbf{S}_{ur}$ of N_u ground users and a lineup $\mathbf{S}_{UAV}$ of N_{UAV} UAVs, as shown in Fig. 1. The target area is (but not limited to) an L-by-L square. Part of the N_u users are randomly distributed in proximity to several scattered hot spots while the rest are uniformly distributed all around $\mathbf{A}$. The UAVs fly horizontally within $\mathbf{A}$ at a fixed altitude H to serve the ground users with guaranteed QoS. We consider that the antennas of each UAV are strongly directional, so that most of the UAV transmit power is concentrated within an aperture angle of θ right below the UAV. Consequently, the ground coverage of a UAV is a disk area with radius $\mathbf{r} = H\tan(\frac{\theta}{2})$, as shown in Fig. 1. Under such settings, users will receive negligible interference from one UAV if they are outside its coverage area.

B. Spectrum Access

All the UAVs are connected to external networks (e.g., Internet) via their back-haul links (e.g., satellite links). The UAV back-haul links and the UAV-user links occupy disjoint spectrum portions, thus producing no mutual interference. Denote the path loss from UAV i to ground user u as PL_{iu}, which follows a commonly adopted model proposed by Al-Hourani *et al.* [39]:

$$PL_{iu} = 20\log_{10}(\frac{4\pi f_c d_{iu}}{c}) + \eta \quad \text{(dB)}, \tag{1}$$

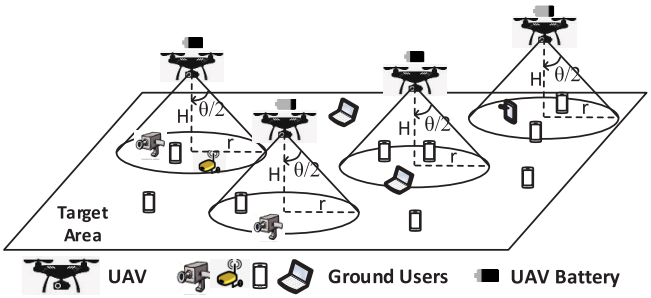


Fig. 1. UAV coverage range as a disk area.

where f_c denotes the center frequency of the spectrum assigned to user u, d_{iu} denotes the 3-D distance between UAV i and user u, c denotes the speed of the light, and η denotes extra loss taking different values for LoS and non-LoS links. Given Eq. (1), the siginal-to-interference-and-noise ratio (SINR) from UAV i to user u is calculated as

$$SINR_{iu} = \frac{P_t G_{iu}}{n_0 + \sum_{j\in\mathbf{S}_u^{UAV}\setminus\{i\}} P_t G_{ju}}, \quad \text{where } G_{iu} = 10^{-PL_{iu}/20}. \tag{2}$$

In Eq. (2), P_t and n_0 is the power spectrum density (psd) of UAV transmission and environment noise, respectively; $\mathbf{S}_u^{UAV}$ denotes the set of UAVs that have user u in the coverage.

Each user requires a minimum throughput of R_u. Therefore, user u can be served by UAV i only when $i \in \mathbf{S}_u^{UAV}$ and the following condition is satisfied:

$$W_{iu}\log_2(1 + SINR_{iu}) \geq R_u, \tag{3}$$

where W_{iu} denotes the bandwidth assigned to user u by UAV i. For user association, each user is associated to the UAV that can provide the highest SINR with enough available bandwidth, according to Eq. (3).

C. Energy-Related Considerations

Each UAV i is powered by battery with initial energy E_0^i. The timing is divided into time slots of duration T. With each time slot t, UAV i spends up to $T_1 < T$ to move a distance of $d_t^i \in [0, d_{max}]$ at a constant speed v in the direction of $\alpha_t^i \in [0, 2\pi)$, and hovers in the new position for the rest of time to interact with the environment. The power consumption of level flight is given as follows according to Eq. (7.10) in [40],

$$P_{level} = \frac{W}{\sqrt{2}\rho A}\frac{1}{\sqrt{v^2 + \sqrt{v^4 + 4V_h^4}}}, \tag{4}$$

where $V_h = \sqrt{\frac{W}{2\rho A}}$, W is the weight of UAV in Newton (N), ρ is the air density, and A is the total area of UAV rotor disks. From Eq. (4), it can be inferred that due to speed v, the power of level flight is interestingly less than that of hovering. Therefore, the energy consumption of UAV i in time slot t (denoted as EC_t^i) can be represented as

$$EC_t^i = E_{FLT}(v, d_t^i, T) + E_{TX} + E_{OP}(T). \tag{5}$$

The energy consumption of a UAV has three components: i) energy consumed in flying as a function of level speed v, flying distance d_t^i, and slot duration T, ii) energy consumed

by signal transmission on both UAV-user and UAV back-haul links, and iii) energy consumed due to operational cost which is assumed to be proportional to T.

Denote the residual battery energy of UAV i at the end of time slot t as E_t^i. When E_t^i is below a threshold E_{Thre}, UAV i will quit the network immediately for charging. Denote the altitude of UAV i at the end of time slot t as $H_t^i \in [H_{min}, H]$, where H_{min} is the altitude of the UAV charging point, and UAV stops elevating after it reaches the serving altitude H. H_t^i will be used in the case of UAV join-in.

D. Problem Formulation

The learning agent aims to find an optimal control policy for a group of UAVs that maximizes the accumulative US scores within a period of N_T time slots, during which the UAV lineup may be changed:

$$\begin{aligned}&\max_{x_t^i, y_t^i} \sum_{t=1}^{N_T} SC_t \\ &s.t. \quad 0 \le x_t^i \le L, \quad \forall i \in \{1, 2, \cdots, N_{UAV}\} \\ &\qquad 0 \le y_t^i \le L, \quad \forall i \in \{1, 2, \cdots, N_{UAV}\}, \\ &\text{where} \\ &SC_t := \left(\sum\nolimits_{u \in \mathbf{S}_{ur}} X_t^u\right)^{\beta}. \end{aligned} \tag{6}$$

In Eq.(6), (x_t^i, y_t^i) denotes the horizontal coordinate of UAV i. The US score in time slot t is denoted as SC_t and defined as a function of the total number of users that get served with satisfied QoS requirement. We define $X_t^u \in \{0, 1\}$ as an indicator taking value 1 when user u is successfully served and 0 when not. The value of X_t^u is jointly determined by UAV parameters (i.e., lineup dynamics, the positions, energy status, altitude), user distribution dynamics, and spectrum access policy. The exponent $\beta > 0$ is a factor indicating how much the agent appreciates the overall user satisfaction based on how many users are successfully served.

With the formulated optimization problem, when one UAV is about to run out of battery and prepares to quit, the agent is expected to proactively relocate the other UAVs before the quit to reduce service holes as much as possible, rather than start dispatching after the UAV quits. When one UAV plans to (re)join the lineup, the agent is expected to determine the optimal join-in position and exerts its discretion to relocate the existing UAVs before the join-in.

III. PRELIMINARIES

In a general RL context, an RL agent interacts with an environment by taking an action A_t based on the environment state (or observation) S_t at epoch t, and gets a reward r_{t+1} for taking A_t at S_t. The agent targets an optimal policy π that provides the best action A for different states S to maximize the future cumulative long-term reward R defined as

$$R = \sum_{t=0}^{\infty} \gamma^t r_{t+1}, \quad \gamma \in [0, 1]. \tag{7}$$

RL approaches are basically categorized into value-based and policy-based ones. Q-learning (QL) [41] is the most basic and representative value-based method. Instead of optimizing π directly, QL estimates the value of taking action A at state S, represented by the function $Q(S, A)$. The optimal policy π^* is then the collection of $A^* = \arg\max_A Q(S, A), \forall S$. The following iterative formula updates $Q(S, A)$ with guaranteed convergence,

$$\begin{aligned}Q_{t+1}(S_t, A_t) &= Q_t(S_t, A_t) \\ &+ \alpha \left[r_{t+1} + \gamma \max_A Q_t(S_{t+1}, A) - Q_t(S_t, A_t)\right], \end{aligned} \tag{8}$$

where α is the adjustable learning rate. However, one of the major drawbacks of QL is that it suffers from "curse of dimensionality". The agent needs to maintain a Q-matrix for every state-action pair, which is challenging or impossible when the state/action space is large or even infinite. This often happens in the area of communication and networking. To this end, deep QL (DQL) is proposed employing a deep neural network (DNN), referred to as deep Q network (DQN), to approximate the $Q(\cdot)$ function [42]. The input dimension of the DQN is equal to the cardinality of the state space, and the output dimension is equal to the number of possible actions. Compared to the Q-matrix, DQN has input dimension equal to the cardinality of the state space, thus solving the memory anxiety due to enormous state space. The DQN is trained by minimizing the loss function below [43]:

$$\mathcal{L}(\theta_Q) = \mathbb{E}[y_t - Q(S_t, A_t|\theta_Q)]^2, \tag{9}$$

where θ_Q denotes the trainable weights of DQN, y_t is the label value calculated as

$$y_t = \begin{cases} r_{t+1}, & \text{if } S_t \text{ is a terminal state;} \\ r_{t+1} + \gamma \max\limits_{A_{t+1}} Q(S_{t+1}, A_{t+1}|\theta_Q), & \text{otherwise.} \end{cases} \tag{10}$$

Although DQL solves the dimension anxiety in state space, the value-based methods may only apply to problems with low-dimensional discrete action space. This is because value-based methods need to conduct an exhaustive search over all possible actions to determine the best for a state, which is challenging for large or infinite action space. For a variety of problems in communication area such as power control and UAV mobility control (as considered in this paper), the action space is continuous. One viable option is to discretize the action space, but with prohibitively high training complexity and/or considerable loss in accuracy.

Thanks to the policy-based methods, the dimension anxiety in action space can be well solved. Instead of estimating the $Q(\cdot)$ values to determine the optimal policy, the methods directly parameterize and optimize the policy $\pi(\theta\mu)$. The $Q(\cdot)$ values may still be used to optimize the policy parameters θ_μ, but not to select an action. Among all the policy-based methods, actor-critic (AC) method outstands in reducing the high variance of policy gradients. As shown in Fig. 2, a basic AC agent consists of a critic and an actor, both being DNNs. The critic takes in the collected experiences, and updates the $Q(\cdot)$ function via updating θ_Q. The actor integrates the updated $Q(\cdot)$ function, updates π via updating θ_μ, and determines the new action A' to be performed in the environment.

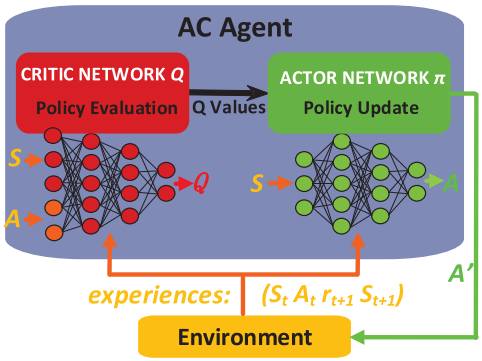


Fig. 2. The diagram for AC method.

Among all the variants of AC algorithms, DDPG is considered among the best to solve the problem of convergence instability [38].

Specifically, target networks are used for both the critic network ($Q(S, A|\theta_Q)$) and actor network ($\mu(S|\theta_\mu)$). The target networks, denoted as $Q'(S, A|\theta_{Q'})$ and $\mu'(S|\theta_{\mu'})$, have the same hyper-parameter setting and initialization as $Q(S, A|\theta_Q)$ and $\mu(S|\theta_\mu)$, respectively, but are updated slowly in each epoch as follows:

$$\theta_{Q'} = \tau\theta_Q + (1-\tau)\theta_{Q'}, \quad \theta_{\mu'} = \tau\theta_\mu + (1-\tau)\theta_{\mu'} \quad (11)$$

where $\tau \ll 1$. The target networks are used to update the label value y_t in Eq. (9). Accordingly, Eq. (10) is re-written as

$$y_t = \begin{cases} r^{t+1}, & \text{if } S_t \text{ is terminal state;} \\ r_{t+1} + \gamma Q'(S_{t+1}, \mu'(S_{t+1}|\theta_{\mu'})|\theta_{Q'}), & \text{otherwise.} \end{cases} \quad (12)$$

The slow update of the target networks prevents y_t from being affected too much by a bad deviation in θ_Q or θ_μ, thus significantly stabilizing the convergence.

With the updated Q values, the actor network is updated using the following sampled policy gradient:

$$\nabla_{\theta_\mu} J \approx \mathbb{E}[G_a G_\mu],$$
$$\text{where } G_a = \nabla_{\mu(S|\theta_\mu)} Q(S, A|\theta_Q), \quad G_\mu = \nabla_{\theta_\mu}\mu(S|\theta_\mu). \quad (13)$$

IV. Design of Proactive Self-Regulation Approaches

In this section, the design of the proposed PSR-APC approach is described in detail. The DDPG agent is implemented in a centralized server, which maintains communication with all the UAVs via their backhaul links. During training, the agent keeps updating both critic and actor networks based on the interaction experiences between UAVs and the environment. During execution, the well-trained actor network sequentially collects the UAV information (states) as inputs and outputs movement instructions (actions) for each UAV in each time slot. These movements collectively result in an optimal set of UAV trajectories to maximize the accumulative US score.

In addition, it is considered that after one UAV quits or joins in, all the serving UAVs will reach the new optimal positions before another UAV lineup change happens. Hence the case of multiple lineup changes can be broken into multiple cases of single lineup change. In such a situation, multiple trained agents for each different lineup change will be employed sequentially to make decisions for all the involved UAVs in the considered period.

In the following, the design is detailed in six aspects: states, actions, reward function, state transitions, tune-ups, and parallel computing. Two cases are considered: the case of UAV quit and the case of UAV join-in.

A. State Definition

1) Case of UAV Quit: The timing of UAV quit and the consequent involved UAV movements closely hinge on the battery energy of the UAVs. Therefore, the system states are designed as two groups: UAV positions and residual battery energy of each UAV.

- The UAV positions determine directly the number of users that get served in each epoch, thus affecting the learning objectives in a straightforward way. Since the UAVs fly at the same height when serving the ground users, only the 2-D coordinates (x_t^i, y_t^i), $\forall i \in \mathbf{S}_{UAV}$ need to be considered at epoch $t \leq N_T$. We limit the movements of UAVs within the target area $\mathbf{A}$, i.e., $x_t^i,\ y_t^i \in [0, L]$.
- The residual battery energy of UAVs $\{E_t^i\}$ is a key conditional factor. By "conditional", it should have little impact on the UAV movement when the battery energy of all UAVs is adequate. But when any E_t^i is close to E_{Thre} (i.e., any UAV is running out of battery and about to quit), this factor is expected to have significant impact on the movement of UAVs. Via learning from the experiences, the DDPG agent will learn the best timing or the critical period for $\{E_t^i\}$ to exert its significance. Moreover, for sustainable UAV operations, E_t^i is bounded within $[E_{Thre}, E_0^i]$.

Collectively, the formal state vector of the designed learning approach is defined as $S_t = [x_t^1, \cdots, x_t^{N_{UAV}}, y_t^1, \cdots, y_t^{N_{UAV}}, E_t^1, \cdots, E_t^{N_{UAV}}]$, with cardinality of $3N_{UAV}$.

2) Case of UAV Join-in: We consider that a UAV starts serving only when it reaches the serving altitude H. Similar to the residual battery energy of UAVs in the case of UAV quit, UAV altitude is the key conditional factor in this case that significantly affects the timing of proactive UAV movement. The existing UAVs will bide their time until when the joining UAV is about to reach the serving altitude. In addition, the position of the joining UAV needs to be carefully managed after it is fully charged and before it reaches the serving altitude. This is because where to join the UAV network is crucial to maximize the accumulative US score.

Therefore, the formal state vector of the designed learning approach is defined as $S_t = [x_t^1, \cdots, x_t^{N_{UAV}}, y_t^1, \cdots, y_t^{N_{UAV}}, H_t^1, \cdots, H_t^{N_{UAV}}]$, with cardinality of $3N_{UAV}$. From the experiences, the DDPG agent will learn the best period for $\{H_t^i\}$ to impose their significant influence.

B. Action Definition

The action set of the APC-PSR approach is the same for both cases. As a centralized DDPG agent controls the

movements of all the UAVs, the action A_t in epoch t consists of collective actions from all the UAVs. The action A_t^i of UAV i encompasses two factors: moving direction $\alpha_t^i \in [0, 2\pi)$ and moving distance $d_t^i \in [0, d_{max}]$. That is to say, each UAV could keep hovering at its current position or move in any direction for a maximum distance d_{max}. To this end, the formal action vector of the proposed APC-PSR approach is defined as $A_t = [\alpha_t^1, \cdots, \alpha_t^{N_{UAV}}, d_t^1, \cdots, d_t^{N_{UAV}}]$, with cardinality of $2N_{UAV}$.

C. Reward Function Design

The reward function is also the same for both cases of UAV quit and UAV join-in. Denote the reward at epoch t as r_t. To maximize the accumulative US score, r_t is designed as a function of the epoch US score SC_t, given as follows:

$$r_t = \left(\frac{\sum_{u \in \mathbf{S}_{ur}} X_t^u}{N_u}\right)^{\beta} = \frac{SC_t}{(N_u)^{\beta}}. \tag{14}$$

In (14), the epoch US score SC_t is divided by $(N_u)^{\beta}$. The reason is that empirically speaking, it is preferable to keep the absolute value of the epoch reward within 1 for better convergence performance. In addition, when $\beta > 1$, the reward difference between different ($\sum_{u \in \mathbf{S}_{ur}} X_t^u$) values is magnified. As a result, the agent is promoted to take proactive actions in advance when the UAV lineup is about to change, i.e., one UAV is about to quit or join in. However, β cannot be too large as $\beta \geq 3$ has been shown ending up with lower converged values in practical simulations. Moreover, under this design, maximizing the accumulated reward is equivalent to maximizing the accumulated US scores within N_T epochs.

An alternative design of reward function is to impose negative rewards to punish any UAVs for moving out of the boundaries, as adopted in [35]. The reward function will be something like:

$$r_t = \begin{cases} \left(\sum_{u \in \mathbf{S}_{ur}} X_t^u / N_u\right)^{\beta}, & \text{if inside boundaries} \\ P, & \text{otherwise} \end{cases} \tag{15}$$

where P can be a negative constant or a negative variable proportional to the number of UAVs moving out of the boundaries. When one UAV does move out of boundaries during training, the current movement will be cancelled. A punishment will be issued for taking the current action A_t at the current state S_t. Consequently, all the episode will contain N_T epochs. Although this design is reasonable, it may make convergence more difficult. Empirically speaking, in a good reward design with both possibly positive and negative rewards, the negative rewards need to "combat" the positive ones closely during the training for better convergence performance and speed. However, the percentages and values of the positive and negative rewards keep changing during the training. Therefore, it may be more challenging and computationally complex to arrive at satisfying convergence which is more sensitive to the relative values between the positive and negative rewards and thus takes more iterations to tune up.

D. State Transition

The terminal state is largely coupled with the design of the epoch reward. In either case, one episode ends on one of the two conditions: i) when any UAV moves outside the boundaries of the target area, i.e., $x_t^i < 0$, $y_t^i < 0$, $x_t^i > L$, or $y_t^i > L$; ii) when N_T epochs are completed. When the terminal state is reached, the agent ends the current episode and starts a new one.

As the dynamics of the UAV lineup is considered, the number of UAVs in the network may change due to the use-up or refreshing of the battery energy. This means that the dimension of the *actual* state-action space to explore during training will be different after one UAV quits or joins the network.

1) Case of UAV Quit: Consider that UAV i quits the network at epoch t_q. Then x_t^i, y_t^i, and E_t^i will stay unaltered for any $t > t_q$. Whatever actions α_t^i and d_t^i ($t > t_q$) are suggested by the actor network, the positions and battery energy of UAV i will never be updated. In other words, the actual dimension of the explorable state space is reduced from $3N_{UAV}$ to $3(N_{UAV} - 1)$. In addition, UAV i will never be involved in the reward calculation after t_q.

2) Case of UAV Join-in: Suppose UAV i completes charging at epoch t_c and is ready to take place to join the network. When $t < t_c$, x_t^i, y_t^i, and H_t^i will stay unchanged. Suppose UAV i elevates to the service altitude at epoch t_s. When $t_c \leq t < t_s$, UAV i is excluded from reward calculation, though its horizontal positions (x_t^i, y_t^i) could change towards the optimal position to maximize the reward when formally joining the network. A constant elevation distance h per epoch is considered.

E. Agent and Training Tune-Ups

1) Agent Tune-Ups: Both the critic and actor networks are DNNs. We would like to keep both networks just large enough to accurately learn the nonlinear mappings between inputs and outputs while preventing overfitting. Both DNNs adopt 2 fully connected hidden layers with 400 and 300 hidden nodes, respectively. To bound the actions as designed in Subsection IV-B, *tanh* and *scaling* layers are exploited in the actor network. ReLU function and L_2 regularization are implemented for activation and overfitting suppression, respectively, in both networks. The learning rates for updating both θ_Q and θ_μ is 10^{-4}. Although a larger learning rate may promote convergence, it more likely leads to convergence instability or sub-optimum. The mini-batch size for DNN training is 512 as a compromise between computational efforts and variance reduction of the gradients of the loss functions. Input normalization is adopted for faster convergence.

2) RL Training Tune-Ups: During RL training, both the target networks $Q'(S, A|\theta_{Q'})$ and $\mu'(S|\theta_{\mu'})$ are updated slowly with rate $\tau = 0.001$. The discount factor γ is set to 0.9. A higher γ forces the agent to care more about the future rewards, thus making convergence more difficult. In addition, DDPG adopts an exploration algorithm where the output of the actor network is superposed with a random noise of zero mean and decaying variance over epochs. In our implementation, the initial variance is 0.6 and decays at a rate of 0.9995. Experience replay is adopted with sufficient buffer to contain

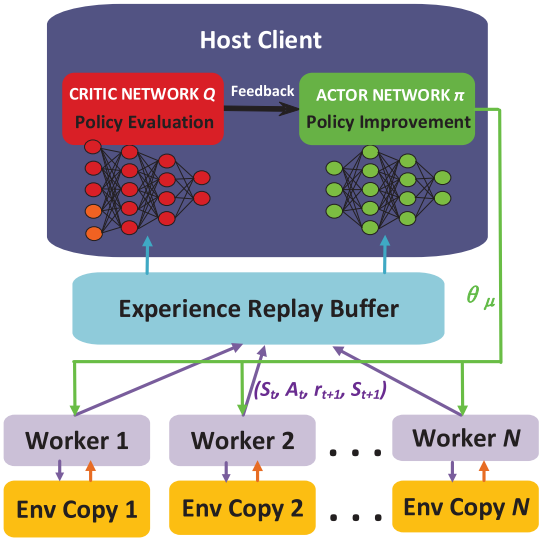


Fig. 3. The diagram for asynchronous parallel computing (APC) of DDPG algorithm.

all the experiences. Insufficient buffer may make the agent lose valuable experiences at early stage if one does not know clearly which experiences to drop, subsequently causing notable convergence instability or even divergence.

F. Parallel Computing

One of the major challenges in RL (especially DDPG) training is how to fully explore the state-action space to reach the optimal converged value. Failure in doing this will very likely cause convergence to a sub-optimal (sometimes even bad) result, which is often the case in our early simulations. After analysis, the reason of incomplete exploration is mainly two-folds. First, although experience replay in DDPG executes random experience sampling from the entire buffer to train the DNNs, the sampled experiences in one mini-batch inevitably have certain degree of correlation. This correlation among training examples of DNN will considerably harm the learning accuracy. Second, the dimension of the explorable state space changes every time when the UAV lineup changes. This dimension change during training often leads to no movement after the lineup change or no proactive movement when the lineup is about to change.

Magnifying the random noise added to the output of the actor network (e.g., increase the initial variance or decay rate) does not really help in our case. Inspired by the asynchronous advantage actor critic (A3C) [44] algorithm, we propose to use asynchronous parallel computing (APC) to promote exploration. The working diagram of APC is shown in Fig. 3.

The diagram is composed of a host client and multiple parallel workers. The host client maintains global critic and actor networks and periodically updates both networks from the collected experiences. The parallel workers are mutually independent, each interacting with a different copy of environment. In A3C, each worker has its own set of network parameters and updates the global networks with the *gradients* of policy loss. Different from A3C, the exploited APC is geared to DDPG algorithm where the parallel workers share the same set of policy parameters from the host client and upload their own *experiences* to the host client for global network updating. In our implementation, each worker uploads the experiences the moment it completes the current episode and immediately receives updated policy parameters from the host, thus being asynchronous.

Algorithm 1 PSR-APC Approach: Host Client Side (UAV Quit/Join-in)

/***Host Client***/
Randomly initialize critic network $Q(S, A|\theta_Q)$ and actor network $\mu(S|\theta_\mu)$;
Initialize the target networks $Q'(S, A|\theta_{Q'})$ and $\mu'(S|\theta_{\mu'})$ with the same weights: $\theta_{Q'} := \theta_Q, \theta_{\mu'} := \theta_\mu$;
while Not all workers complete all episodes **do**
  if Receive experience set **Ep** from worker k **then**
    Store **Ep** into experience replay buffer B;
    Send θ_μ to worker k;
  end if
  Sample a mini-batch of experiences from B;
  Update θ_Q according to Eq. (9)(12);
  Update θ_μ according to Eq. (13);
  Update $\theta_{Q'}$ and $\theta_{\mu'}$ according to (11);
end while

Having independent workers interacting with different copies of environment makes the experiences of each worker independent from the others, i.e., better exploration is achieved. In this way, the overall experiences available for training becomes more diverse, thus reducing the correlation among training examples in each mini-batch. But note that APC does not increase the convergence speed in our case as the computational cost of training is much higher than that of simulating the environment. The proposed APC-PSR approach is summarized in Algorithms 1 and 2 for both cases of UAV quit and join-in.

V. Proactive Self-Regulation With Dynamic User Distribution

The above section bases proactive self-regulation of UAV networks on *fixed* user distribution within the considered time frame. It can be more practical if *dynamic* user distribution is considered. Dynamic user distribution signifies that the optimal UAV positions may change from time to time in order to maximize the accumulated US score within N_T time slots, which is more challenging compared to the fixed case. With time-varying user distribution, the optimal self-regulation of UAVs in this work is more towards the optimal UAV trajectory design, with additional considerations on proactive response to UAV lineup change.

The key to the above problem is how to determine the optimal trajectory for each UAV according to the dynamically changing user distributions. There have already been some yet not many existing works [8], [29] that consider time-varying user distributions under the RL framework. In these works, time-varying user mobility patterns are first predicted using either echo state networks (ESNs) or Long-Short Term Memory (LSTM), based on which (multi-agent) QL is employed to achieve the optimal UAV trajectories. Nevertheless, the above

Algorithm 2 PSR-APC Approach: Parallel Worker Side (UAV Quit/Join-in)

/***Parallel Worker***/
for episode:$= 1, \cdots, N$ **do**
  Obtain the initial state S_1, IsTerminal:= False;
  for epoch t: $= 1, \cdots, N_T$ **do**
    $A_t = \mu(S_t|\theta_\mu) + \mathcal{N}$, where $\mathcal{N}$ is stochastic noise with zero mean and decaying variance over t;
    Execute A_t and observe next state S_{t+1};
    for UAV i: $= 1, \cdots, N_{UAV}$ **do**
      /*Case of *UAV Quit**/
      if $E_t^i <= E_{Thre}$ **then**
        $S_{t+1}^i := S_t^i$, where $S_t^i = \{x_t^i, y_t^i, E_t^i\}$;
        Exclude UAV i when calculating r_{t+1};
      else
        Obtain S_{t+1}^i according to S_t^i and A_t;
      end if
      /*Case of *UAV Join-In**/
      if $H_t^i < H$ **then**
        Exclude UAV i when calculating r_{t+1};
        $H_{t+1}^i = \min\{H_t^i + h, H\}$;
      end if
      Obtain $\{x_{t+1}^i, y^{i_{t+1}}\}$ according to S_t^i and A_t;
      /*Shared codes begin*/
      if UAV i goes out of boundaries **then**
        Cancel the movement of UAV i;
        IsTerminal: = True;
      end if
    end for
    Calculate r_{t+1};
    if IsTerminal == True **then**
      Break;
    end if
  end for
  Send experiences in this episode to host client;
  Obtain updated θ_μ from host client;
end for

works first derive the optimal UAV positions in each time slot by conducting RL given a slot-specific user distribution, and then string the optimal positions in different slots together into the trajectories. This requires the RL agent(s) to be trained once every time slot, instead of once for the entire time horizon. Consequently, high training complexity may be incurred if a large number of time slots are involved. Different from the above works, we incorporate the time slot t as one dimension of the state space, so that the obtained policy after training considers the dynamic distribution during the entire time horizon rather than one specific distribution in a particular time slot. In this way, the agent only needs to be trained once over the entire time horizon, and ends up with a time-aware optimal policy that may take different actions at different time slots even if the rest of the states are the same.

To investigate the capability of the proposed APC-PSR approach in learning the time-varying feature of the user distribution, a simplified dynamic model is exploited. Instead of imposing certain mobility models on individual users, we assign a different trace for each hot spot center. Following the trace, the center of each hot spot moves around the target area $\mathbf{A}$ along with time t. The portions of users distributed uniformly throughout $\mathbf{A}$ or around each hot spot can be either fixed or moderately fluctuated. An example of moving hot spots and corresponding user distribution is illustrated in Fig. 4. As shown in Fig. 4, there are 4 hot spots initially scattered at 4 corners of $\mathbf{A}$. During N_T time slots, the 4 hot spots first move towards the center of $\mathbf{A}$, i.e., ($L/2$, $L/2$), then stop when reaching a certain distance from the center, stay for a period, and finally move back to where they were initially. Such a disperse-gather-disperse procedure can be used to simulate some realistic scenarios such as how users commute between residence communities and a central business district in workdays.[1] Note that any time-varying model of user distribution should be applicable to the proposed approach as long as the user mobility is not much higher than the UAV mobility to ensure effectiveness.

As the user distribution is time-varying, even if the UAVs are in the same positions and energy/altitude status, they may take different actions at different time slots. Therefore, the design of APC-PSR approach needs to be revised to involve time as one of the states. To this end, the states of the proposed APC-PSR approach is redefined as: $S_t = [x_t^1, \cdots, x_t^{N_{UAV}}, y_t^1, \cdots, y_t^{N_{UAV}}, E_t^1, \cdots, E_t^{N_{UAV}}, t]$ (case of UAV quit), or $S_t = [x_t^1, \cdots, x_t^{N_{UAV}}, y_t^1, \cdots, y_t^{N_{UAV}}, H_t^1, \cdots, H_t^{N_{UAV}}, t]$ (case of UAV join-in), both with cardinality of $3N_{UAV} + 1$.

Note that despite the example distribution dynamics, the proposed approach is deemed to be applicable to any kind of distribution dynamics as long as the distribution can be considered invariant within one time slot.

VI. Numerical Results

Numerical results are presented in this section to demonstrate the performance of the proposed APC-PSR approach.

A. Simulation Setup

The target area is square with 10×10 square units. Each unit is 100 meters. The training is conducted using Reinforcement Learning Toolbox of Matlab 2020a on a Windows 10 server with Intel Core i7-7700 CPU @ 3.60GHz and 16GB RAM. The training has maximum 10000 episodes, each having up to 100 epochs. The trained agents are tested for a period of $N_T = 100$ epochs. Moreover, the communication-related power of UAVs is considered negligible compared to the flight/hovering power [45]. The main parameters are summarized in Table I below. Note that $unit \cdot s$ in the table indicates that the value is a product of power (1 power unit = 9.428W according to Eq. (4)) and time (unit is second).

In most of the simulated cases, the learning converges to a narrow interval instead of a fixed value. To handle such a situation, the accumulated US scores are demonstrated as average values over the latest 100 episodes.

[1] In such a scenario, the duration of one time slot needs to be scaled up to the order of minutes.

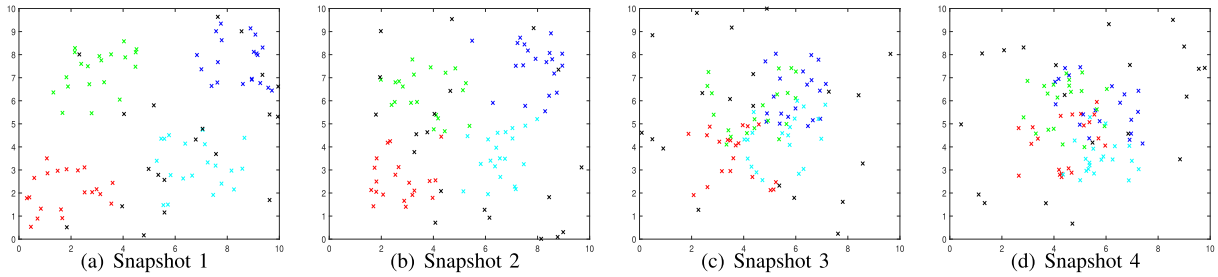


Fig. 4. Illustration of time-varying user distribution: Snapshots of different time slots. Users are represented by dots of different colors. Users of black are uniformly distributed throughout $\mathbf{A}$, and users of each other color belongs to one hot spot. From snapshots 1 to 4, hot spots move from being scattered to being overlapped.

TABLE I
SUMMARY OF MAIN PARAMETERS

Parameters	Values
Default number of users N_u	100
Default number of UAVs N_{UAV}	5
UAV level speed v	$40km/h$
UAV max. elevation speed	$14.4km/h$
UAV weight W, air density ρ	$4kg\times9.8m/s^2$, $1.225kg/m^3$
Total area of rotor disks A	$0.18m^2$
UAV height H, aperture angle θ	3 units, 60^o
Max. distance per epoch d_{max}	1 unit
Spectrum center frequency f_c	2GHz
Spectrum access technology	LTE with resource blocks (RBs)
Spectrum and RB bandwidth	4.5MHz and 180kHz
psd of transmission and noise	-49.5dBm, -174dBm
Required user throughput R_u	250kbps
LOS path loss parameter η	1dB
Time duration per epoch T	10s
Max. UAV moving (communication) time per epoch T_1 $(T-T_1)$	9s (1s)
Factor of US score β	2
Energy threshold to quit E_{Thre}	150 unit·s

Intermediate agents with good episode rewards are saved during training and compared during tests to determine the best trained agent.

B. Simulation Results

1) Case Without UAV or User Dynamics: The cases without any UAV lineup or user distribution change are first simulated to get a benchmark of optimal UAV positions for different N_{UAV}. The convergence performance of accumulated US scores is shown in Fig. 5. All UAVs are initially positioned evenly in a circle centered at (5,5). It can be observed that for all the simulated N_{UAV} values, the accumulated US scores eventually converge, with larger N_{UAV} taking more training episodes. The reason is straightforward: the more UAVs there are, the larger dimension of the state-action space has, thus requiring more time to fully explore and exploit. In addition, the converged accumulated US score is smaller for smaller N_{UAV}, which aligns with the fact that more UAVs can cover more users until the target area is saturated with UAVs. However, as N_{UAV} increases, the increment in the number of served users reduces according to Table II.

2) Case of UAV Quit: We then simulate the case of UAV quit where N_{UAV} UAVs start off in the beginning and 1 UAV quits within the considered period. The UAVs are initially positioned at the optimal locations obtained via Fig. 5. Although multiple UAVs may quit during the period, we consider that when one UAV quits, the remaining UAVs will reach the new optimal positions before another UAV quits. Hence the case of multi-UAV quit can be broken into multiple cases of single-UAV quit. The convergence performance is presented in Fig. 6. It can be seen that it takes more episodes for

TABLE II
MAXIMUM NUMBER OF SERVED USERS WITH N_{UAV}

N_{UAV}	3	4	5	6
Number of served users	56	71	80	88
Increment	-	15	9	8

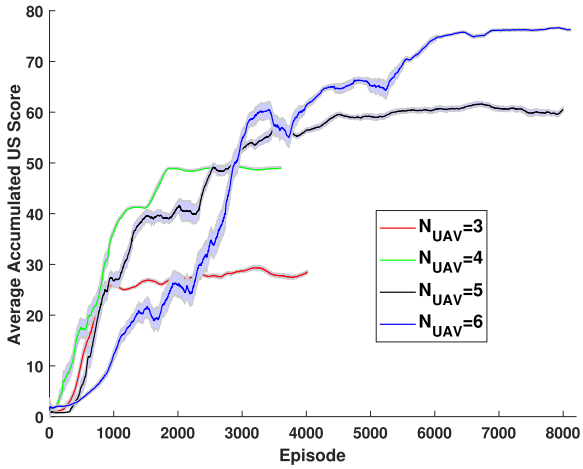


Fig. 5. Convergence with 95% credit interval for different N_{UAV} without UAV or user dynamics as a benchmark.

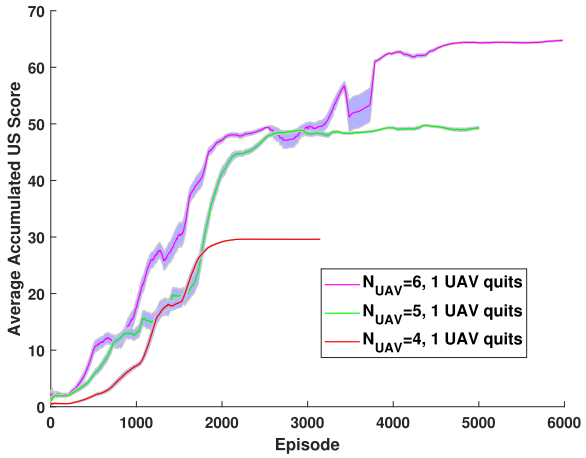
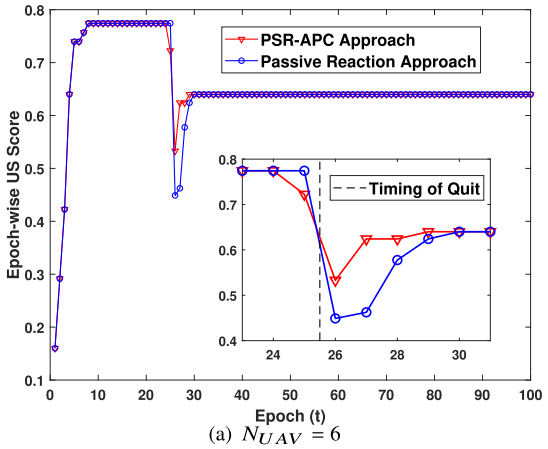
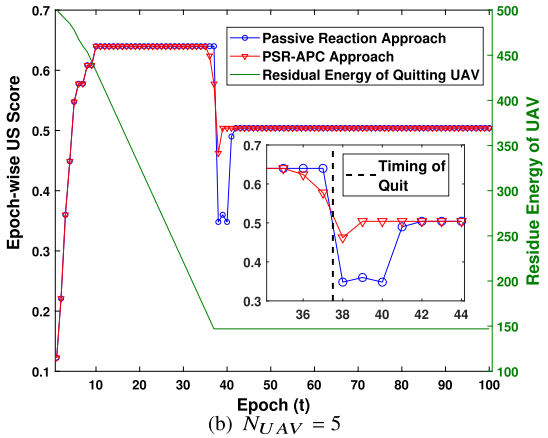


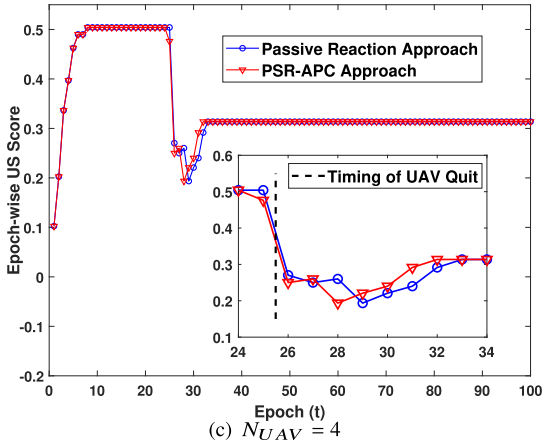
Fig. 6. Convergence performance with 95% credit interval for the case of UAV quit.



(a) $N_{UAV} = 6$



(b) $N_{UAV} = 5$



(c) $N_{UAV} = 4$

Fig. 7. Case of UAV quit: Epoch-wise reward comparison between the PSR-APC approach and the passive reaction approach.

larger N_{UAV} to converge. Then, the optimal epoch-wise US scores are shown in Fig. 7. As a comparison to the proposed PSR-APC approach, a passive reaction approach is also simulated, which only relocates the remaining UAVs passively after one UAV quits the network.

The epoch-wise US scores in Fig. 7 are obtained by combining the agents achieved in Fig. 5 and Fig. 6. The UAVs start from the circular positions, then to the optimal positions maximizing the epoch US score; a UAV then quits the network and the remaining UAVs are finally relocated to the new optimal positions. It can be observed that for all the simulated N_{UAV}, the epoch US score first increases to a maximum as

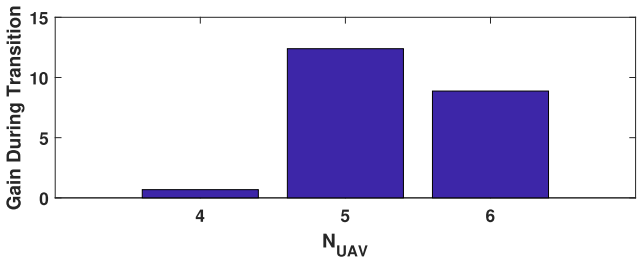
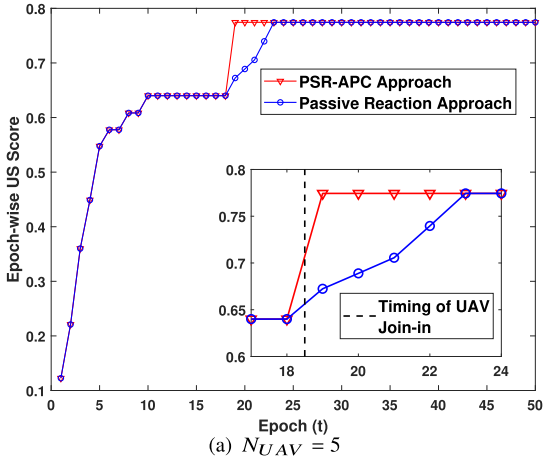


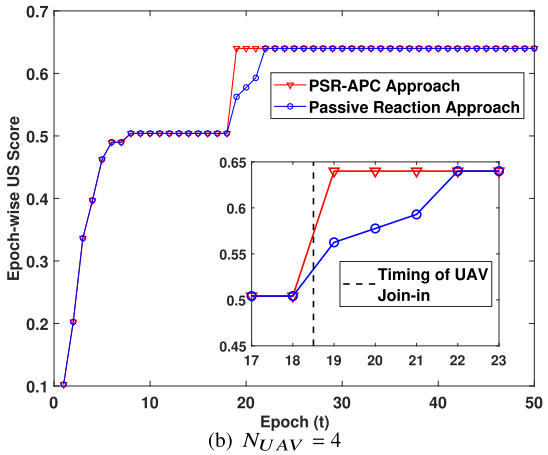
Fig. 8. Case of UAV quit: Gain (%) of PSR-APC over the passive reaction approach in accumulated US scores during transition to the new optimal UAV positions.

the UAVs are heading to the optimal positions. When a UAV quits the network, the epoch US scores drop dramatically due to the service holes caused by the quit. After a short period of self-regulation, the scores rise up to a new maximum smaller than the previous one when the remaining UAVs reach the new optimal positions. The proposed PSR-APC and the passive reaction approach differ around the timing of UAV quit. The passive reaction approach has no reaction before UAV quit and thus experiences dramatic drop in US scores. On the contrary, the PSR-APC approach monitors the UAV battery status and starts moving the UAVs one or two epochs before the UAV quit. Although the epoch US scores may drop early due to pre-movements, they will not drop that low when the UAV quit as those under the passive reaction approach. Besides, the transition process will be completed earlier. As a result, the accumulated US scores during the transition are higher than those under the passive reaction approach, as shown in Fig. 8. However, proactive movement is not always considerably beneficial since the gain depends on specific user distribution and user-to-UAV ratio. When $N_{UAV} = 4$, the gain is marginal. The reason is that before one UAV quits, the 4 UAVs are separately positioned over 4 hot spots far away from each other. When one UAV quits, at least one UAV needs to move a long way to the next optimal position, along which the epoch US score even drops lower. In such a situation, the gain of pre-movements are significantly diluted by the long transition period.

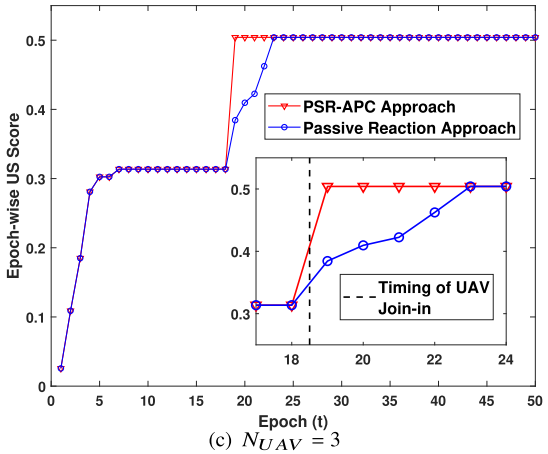
3) Case of UAV Join-in: The case of UAV join-in is then simulated with different N_{UAV}. The epoch-wise US scores under both the PSR-APC approach and the passive reaction approach are presented in Fig. 9. There are initially N_{UAV} UAVs that start off at the unit circular positions, and then reach the optimal positions. A joining UAV starts elevating from the ground in the center (5,5) at epoch 11, and reaches the serving altitude (i.e., formally join the network) at epoch 19. The passive reaction approach in this case relocates UAVs only after the joining UAV joins the network right above (5,5), while the PSR-APC approach starts tuning the horizontal positions of the joining UAV right after it starts off. This ensures that when the joining UAV reaches the serving altitude, all the UAVs are already near the new optimal positions. This is confirmed by the curves in Fig. 9. It can be observed that under the PSR-APC approach, all the UAVs are at the new optimal positions in the very first epoch after the new UAV joins in, while it takes the passive approach couple of epochs to dispatch the UAVs to the new optimal positions. In addition,



(a) $N_{UAV} = 5$



(b) $N_{UAV} = 4$



(c) $N_{UAV} = 3$

Fig. 9. Case of UAV join-in: Epoch-wise reward comparison between the PSR-APC approach and the passive reaction approach.

there is no pre-movement of the existing UAVs when the new UAV is about to join. This is because the new optimal positions of the existing UAVs are within 1 epoch reach to the previous optimal positions in our user distribution settings. The gain in accumulated US score during the transition period introduced by the PSR-APC approach is shown in Fig. 10, achieving at least 10% for all simulated N_{UAV}.

4) Case of UAV and User Dynamics: At last, the case with dynamic user distribution is simulated. The disperse-gather-disperse procedure shown in Fig. 4 is employed. We divide the 100 epochs evenly into 10 segments. The centers of the

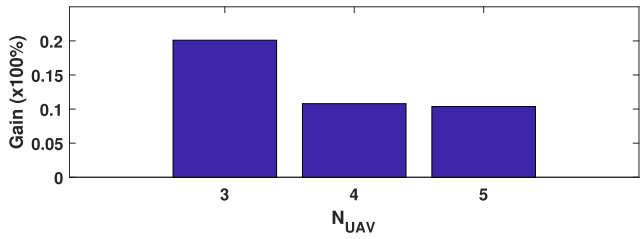


Fig. 10. Case of UAV join-in: Gain (%) of PSR-APC over the passive reaction approach in accumulated US scores during transition to the new optimal UAV positions.

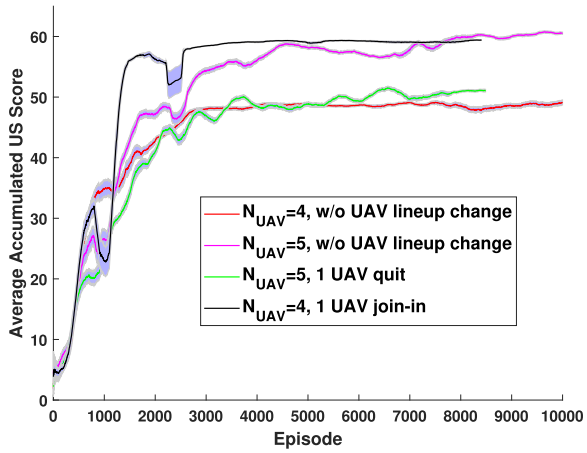
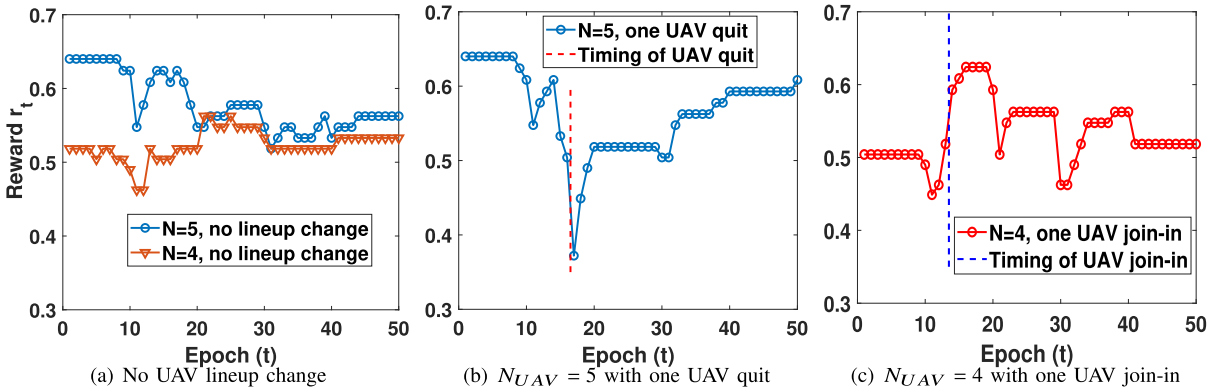


Fig. 11. Convergence with 95% credit interval with user dynamics.

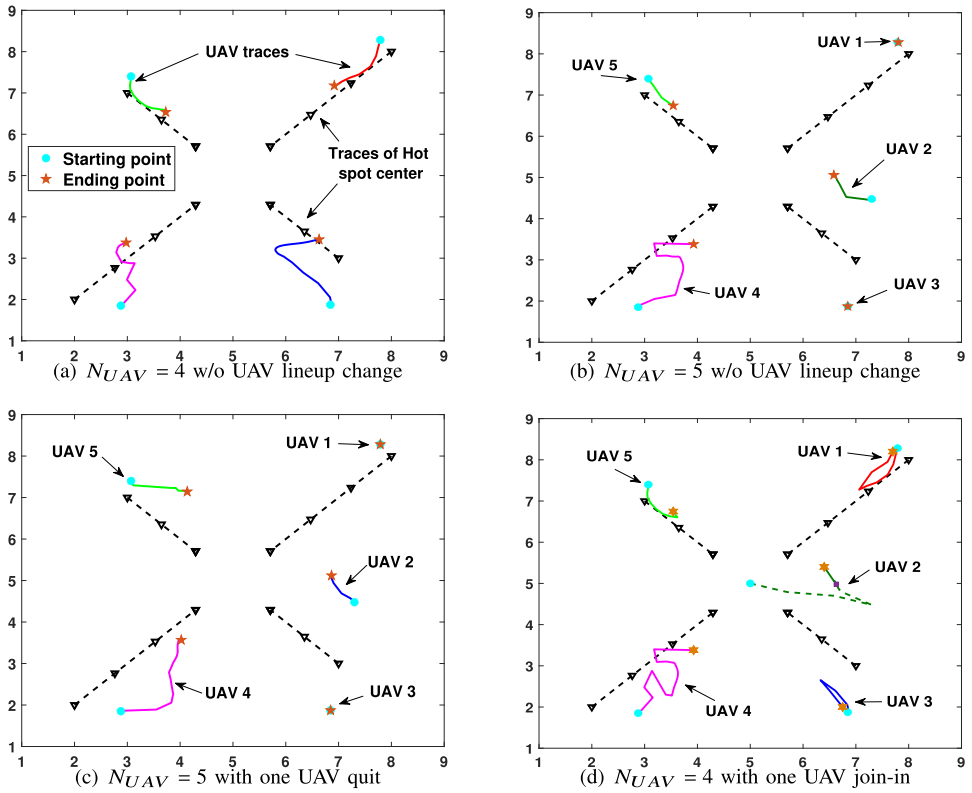
hot spots are updated (i.e., user distribution is updated) at the beginning of each segment. The order of update is snapshot $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 4 \rightarrow 4 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1$. The N_{UAV} UAVs start off initially from the optimal positions of snapshot 1, and move accordingly while the user distribution changes. The convergence of 4 situations is shown in Fig. 11: 4 UAVs and 5 UAVs with no UAV quit or join-in, 5 UAVs with 1 UAV quit, and 4 UAVs with 1 UAV join-in. In addition, the epoch-wise reward of each situation is presented in Fig. 12. It can be observed that the epoch-wise rewards are relatively steady within each time segment (where the hotspot centers remain still), but experience considerable changes when crossing the time segments. As shown in Subfigures 12(b)(c), our proposed approach can also handle the change in UAV lineup with time-varying user distribution, by getting the UAVs to the new optimal positions soon after the change.

The UAV trajectories are also demonstrated in Fig. 13. As the disperse-gather procedure is just the opposite mirror of the gather-disperse procedure, we only present the first 50 epochs. In all 4 subfigures, the black dashed lines represent the traces of the hotspot centers, moving from corners towards the center of the target region, and stops 1 unit away from the center. In Subfigure 13(a), as the hotspots move, the 4 UAVs proactively move from the initial positions (solid aqua circles) towards the region center to cover as many users as possible. But instead of exactly following the hotspot centers, the UAVs stop farther from the region center (solid brown pentagrams). This is because i) the ground coverage radius of each UAV is larger than 1 unit, and ii) coverage overlapping of UAVs will lead to significant intercell interference and further affect the



(a) No UAV lineup change (b) $N_{UAV} = 5$ with one UAV quit (c) $N_{UAV} = 4$ with one UAV join-in

Fig. 12. Epoch-wise reward of different situations with dynamic user distributions.



(a) $N_{UAV} = 4$ w/o UAV lineup change (b) $N_{UAV} = 5$ w/o UAV lineup change

(c) $N_{UAV} = 5$ with one UAV quit (d) $N_{UAV} = 4$ with one UAV join-in

Fig. 13. Optimal UAV trajectories with user distribution dynamics. The x and y axis represent the horizontal coordinates.

user QoS. Situations are different when there are 5 UAVs, as shown in Subfigure 13(b). Interestingly, while hotspots move, UAV 1 and UAV 3 almost stay still because of the existence of UAV 2. As UAV 2 moves towards the region center, it is able to cover most of the user flows from hotspots in the top and bottom right corners, so that UAV 1 and UAV 3 can stay put to cover more uniformly distributed users. Accordingly, the trace of UAV 4 leans a little towards the center to help cover the center users, and the trace of UAV 5 backs off a little to reduce overlapping.

Subfigure 13(c) shows the UAV traces when UAV 1 quits the network around epoch 16. It can be observed that after UAV 1 quits, the traces of UAV 2 and 5 turn more towards the hotspot in the top right corner to cover more users. UAV 4 goes deeper towards the region center to cover the users missed by UAV 2 and 5 while UAV 3 stays put. Subfigure 13(d) shows the UAV traces when UAV 2 starts off at the region center and formally joins the network at epoch 14. The dashed part of UAV 2 represents horizontal trace before it reaches the serving altitude. It can be observed that before UAV 2 joins the network, the 4 existing UAVs move very similarly to Subfigure 13(a). After UAV 2 joins, UAV 1 and 3 gradually back to the proximity of their original starting points, while UAV 4 and 5 start to move like subfigure 13(b) where there are 5 existing UAVs.

VII. Conclusion

In this paper, proactive self-regulation of the UAV-based networks has been studied considering dynamic UAV lineup and user distribution. A DRL-based learning approach, i.e, PSR-APC, has been developed to proactively control the UAV trajectories given that at least one UAV will quit or join the

network during a certain period. Dynamic user distribution has also been accommodated as practical enhancements. Simulation results have shown that compared to the passive reaction approach, PSR-APC can achieve up to 20% higher accumulated US scores during the transition process. In addition, when the user distribution is dynamically changing, the PSR-APC approach has been able to capture the dynamics and move UAVs accordingly.

REFERENCES

[1] R. Zhang, M. Wang, and L. X. Cai, "SREC: Proactive self-remedy of energy-constrained UAV-based networks via deep reinforcement learning," 2020, *arXiv:2009.08528*. [Online]. Available: http://arxiv.org/abs/2009.08528

[2] Q. Zhang, M. Jiang, Z. Feng, W. Li, W. Zhang, and M. Pan, "IoT enabled UAV: Network architecture and routing algorithm," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3727–3742, Apr. 2019.

[3] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 36–42, May 2016.

[4] (Oct. 2019). *Unmanned Aerial Vehicle (UAV) Market*. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/unmanned-aerial-vehicles-uav-market-662.html

[5] N. Zhao *et al.*, "UAV-assisted emergency networks in disasters," *IEEE Wireless Commun.*, vol. 26, no. 1, pp. 45–51, Feb. 2019.

[6] A. A. Nasir, H. D. Tuan, T. Q. Duong, and H. V. Poor, "UAV-enabled communication using NOMA," *IEEE Trans. Commun.*, vol. 67, no. 7, pp. 5126–5138, Jul. 2019.

[7] N. H. Motlagh, M. Bagaa, and T. Taleb, "UAV-based IoT platform: A crowd surveillance use case," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 128–134, Feb. 2017.

[8] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, "Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 5, pp. 1046–1061, May 2017.

[9] 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.

[10] D. Shi, H. Gao, L. Wang, M. Pan, Z. Han, and H. V. Poor, "Mean field game guided deep reinforcement learning for task placement in cooperative multiaccess edge computing," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9330–9340, Oct. 2020.

[11] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2334–2360, 3rd Quart., 2019.

[12] Q. Wu, L. Liu, and R. Zhang, "Fundamental trade-offs in communication and trajectory design for UAV-enabled wireless network," *IEEE Wireless Commun.*, vol. 26, no. 1, pp. 36–44, Feb. 2019.

[13] Y. Zeng, X. Xu, and R. Zhang, "Trajectory design for completion time minimization in UAV-enabled multicasting," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2233–2246, Apr. 2018.

[14] Q. Wu and R. Zhang, "Common throughput maximization in UAV-enabled OFDMA systems with delay consideration," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6614–6627, Dec. 2018.

[15] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2329–2345, Apr. 2019.

[16] H. Guo and J. Liu, "UAV-enhanced intelligent offloading for Internet of Things at the edge," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2737–2746, Apr. 2020.

[17] M. Mozaffari, A. T. Z. Kasgari, W. Saad, M. Bennis, and M. Debbah, "Beyond 5G with UAVs: Foundations of a 3D wireless cellular network," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 357–372, Jan. 2019.

[18] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-UAV enabled wireless networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 2109–2121, Mar. 2018.

[19] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Mobile unmanned aerial vehicles (UAVs) for energy-efficient Internet of Things communications," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7574–7589, Nov. 2017.

[20] 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.

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

[22] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, May 2019.

[23] S. Zhang, H. Zhang, Z. Han, H. V. Poor, and L. Song, "Age of information in a cellular Internet of UAVs: Sensing and communication trade-off design," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6578–6592, Oct. 2020.

[24] S. Zhang, Y. Zeng, and R. Zhang, "Cellular-enabled UAV communication: A connectivity-constrained trajectory optimization perspective," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 2580–2604, Mar. 2019.

[25] M. Chen, W. Saad, and C. Yin, "Liquid state machine learning for resource and cache management in LTE-U unmanned aerial vehicle (UAV) networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 3, pp. 1504–1517, Mar. 2019.

[26] P. V. Klaine, J. P. B. Nadas, R. D. Souza, and M. A. Imran, "Distributed drone base station positioning for emergency cellular networks using reinforcement learning," *Cognit. Comput.*, vol. 10, no. 5, pp. 790–804, Oct. 2018.

[27] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning-based resource allocation for UAV networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 2, pp. 729–743, Feb. 2020.

[28] J. Hu, H. Zhang, L. Song, Z. Han, and H. V. Poor, "Reinforcement learning for a cellular Internet of UAVs: Protocol design, trajectory control, and resource management," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 116–123, Feb. 2020.

[29] X. Liu, Y. Liu, Y. Chen, and L. Hanzo, "Trajectory design and power control for multi-UAV assisted wireless networks: A machine learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7957–7969, Aug. 2019.

[30] S. Singh, A. Kumbhar, I. Güvenç, and M. L. Sichitiu, "Distributed approaches for inter-cell interference coordination in UAV-based LTE-advanced HetNets," in *Proc. IEEE 88th Veh. Technol. Conf. (VTC-Fall)*, Aug. 2018, pp. 1–6.

[31] U. Challita, W. Saad, and C. Bettstetter, "Interference management for cellular-connected UAVs: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2125–2140, Apr. 2019.

[32] F. Tang, Y. Zhou, and N. Kato, "Deep reinforcement learning for dynamic uplink/downlink resource allocation in high mobility 5G HetNet," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 12, pp. 2773–2782, Dec. 2020.

[33] X. Cheng *et al.*, "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.

[34] S. Khairy, P. Balaprakash, L. X. Cai, and Y. Cheng, "Constrained deep reinforcement learning for energy sustainable multi-UAV based random access IoT networks with NOMA," 2020, *arXiv:2002.00073*. [Online]. Available: http://arxiv.org/abs/2002.00073

[35] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2059–2070, Sep. 2018.

[36] Y. Sun, D. Xu, D. W. K. Ng, L. Dai, and R. Schober, "Optimal 3D-trajectory design and resource allocation for solar-powered UAV communication systems," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4281–4298, Jun. 2019.

[37] Y. Huang, X. Mo, J. Xu, L. Qiu, and Y. Zeng, "Online maneuver design for UAV-enabled NOMA systems via reinforcement learning," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, May 2020, pp. 1–6.

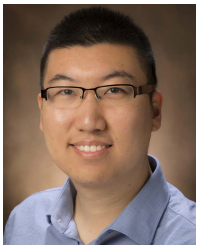
[38] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*. [Online]. Available: http://arxiv.org/abs/1509.02971

[39] A. Al-Hourani, S. Kandeepan, and A. Jamalipour, "Modeling air-to-ground path loss for low altitude platforms in urban environments," in *Proc. IEEE Global Commun. Conf.*, Dec. 2014, pp. 2898–2904.

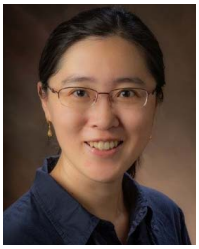
[40] J. M. Seddon and S. Newman, *Basic Helicopter Aerodynamics*, vol. 40. Hoboken, NJ, USA: Wiley, 2011.

[41] M. Han, S. Khairy, L. X. Cai, Y. Cheng, and R. Zhang, "Reinforcement learning for efficient and fair coexistence between LTE-LAA and Wi-Fi," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8764–8776, Aug. 2020.

[42] D. Shi *et al.*, "Deep Q-network-based route scheduling for TNC vehicles with passengers' location differential privacy," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7681–7692, Oct. 2019.
[43] T. Hester *et al.*, "Deep Q-learning from demonstrations," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.
[44] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[45] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3747–3760, Jun. 2017.



Ran Zhang (Member, IEEE) received the B.Sc. degree from Tsinghua University, China, in 2010, and the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2016. He then joined as a System Engineer with the Ottawa Research Center, Huawei Technologies, Markham, ON, Canada, in 2016. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Miami University, Oxford, OH, USA. His research interests include radio resource management for next-generation wireless communication networks, the Internet of Things (IoT), and to channel coding for 5G New Radio. His current research emphasizes on Unmanned Aerial Vehicle (UAV)-based communications, machine learning, and radio resource management for 5G/6G mobile communications. He received the Best Paper Award from the IEEE GLOBECOM 2014 and the CFR Faculty Research Award from Miami University in 2021.



Miao Wang (Member, IEEE) received the B.Sc. degree from the Beijing University of Posts and Telecommunications in 2007, and the M.Sc. degree from Beihang University, Beijing, China, in 2010, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2015. She is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Miami University, Oxford, OH, USA. She also serves as an Associate Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. Her current research interests include the Internet of Things, cyber physical self-driving system, electric vehicles in smart grid, space-air-ground communications, and machine learning.

Lin X. Cai (Senior Member, IEEE) received the M.A.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2005 and 2010, respectively. She is currently an Associate Professor with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA. Her research interests include green communication and networking, intelligent radio resource management, and wireless Internet of Things. She received the Post-Doctoral Fellowship Award from the Natural Sciences and Engineering Research Council of Canada (NSERC) in 2010, the Best Paper Award from the IEEE GLOBECOM 2011, the NSF Career Award in 2016, and the IIT Sigma Xi Research Award in the Junior Faculty Division in 2019. She is an Associate Editor of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, *IEEE Network Magazine*, and the Co-Chair of IEEE conferences.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University–New Brunswick, New Brunswick, NJ, USA, in 1990. He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research interests include network resource management, wireless network security, the Internet of Things, 5G and beyond, and vehicular ad-hoc and sensor networks. He is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Fellow, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society. He is a member of the IEEE Fellow Selection Committee. He received the R. A. Fessenden Award in 2019 from the IEEE, Canada, the Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, the James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, the Joseph LoCicero Award in 2015 and the Education Award in 2017 from the IEEE Communications Society, the Technical Recognition Award from the Wireless Communications Technical Committee in 2019 and the AHSN Technical Committee in 2013, the Excellent Graduate Supervision Award in 2006 from the University of Waterloo, and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario. He has served as the Technical Program Committee Chair/Co-Chair for the IEEE GLOBECOM 2016, the IEEE INFOCOM 2014, the IEEE VTC2010 Fall, and the IEEE GLOBECOM 2007, the Symposia Chair for the IEEE ICC 2010, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He was/is the Editor-in-Chief of the IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, *IET Communications*, and *Peer-to-Peer Networking and Applications*. He is the elected IEEE Communications Society Vice President for Technical and Educational Activities, the Vice President for Publications, a Member-at-Large on the Board of Governors, and the Chair of the Distinguished Lecturer Selection Committee.