Energy Minimization in UAV-Aided Networks: Actor-Critic Learning for Constrained Scheduling Optimization

Yaxiong Yuan, *Student Member, IEEE*, Lei Lei, *Member, IEEE*, Thang X. Vu, *Member, IEEE*, Symeon Chatzinotas, *Senior Member, IEEE*, Sumei Sun, *Fellow, IEEE*, and Björn Ottersten, *Fellow, IEEE*

***Abstract*—In unmanned aerial vehicle (UAV) applications, the UAV's limited energy supply and storage have triggered the development of intelligent energy-conserving scheduling solutions. In this paper, we investigate energy minimization for UAV-aided communication networks by jointly optimizing data-transmission scheduling and UAV hovering time. The formulated problem is combinatorial and non-convex with bilinear constraints. To tackle the problem, firstly, we provide an optimal algorithm (OPT) and a golden section search heuristic algorithm (GSS-HEU). Both solutions are served as offline performance benchmarks which might not be suitable for online operations. Towards this end, from a deep reinforcement learning (DRL) perspective, we propose an actor-critic-based deep stochastic online scheduling (AC-DSOS) algorithm and develop a set of approaches to confine the action space. Compared to conventional RL/DRL, the novelty of AC-DSOS lies in handling two major issues, i.e., exponentially-increased action space and infeasible actions. Numerical results show that AC-DSOS is able to provide feasible solutions, and save around 25-30% energy compared to two conventional deep AC-DRL algorithms. Compared to the developed GSS-HEU, AC-DSOS consumes around 10% higher energy but reduces the computational time from second-level to millisecond-level.**

***Index Terms*—UAV, deep reinforcement learning, user scheduling, hovering time allocation, energy optimization, actor-critic.**

I. Introduction

UNMANNED aerial vehicles (UAVs) have attracted much attention to high-speed data transmission in dynamic, distributed, or plug-and-play scenarios, e.g., disaster rescue, live concert, or sports events [1]. However, UAVs' limited endurance, energy supply, and storage become critical issues for their applications, which motivates the study of energy efficiency in UAV-aided communication networks. The UAV's energy consumption comes from two aspects, propulsion energy for flying and hovering, and communication energy for data transmission. The flying energy mainly depends on the UAV's velocity and trajectory [1]. The hovering energy is, in general, proportional to the hovering time. Compared to the propulsion energy, the communication energy consumption is not a negligible part, e.g., considerable communication energy can be consumed in the scenarios with high traffic requests from a large number of users. Thus joint energy optimization for both parts is necessary and has attracted considerable attention in the literature [2]–[9].

The authors in [2], [3] maximized the energy efficiency, referring to the ratio between transmitted data and propulsion energy. In [4], the authors introduced a complete UAV energy model and proposed a user-timeslot scheduling method to minimize the sum of the propulsion energy and communication energy. Based on the energy model in [4], the authors formulated an energy minimization problem with latency constraints by trajectory design in [5]. The above works in [2]–[5] adopted a time division multiple access (TDMA) mode, where the UAV serves one user per timeslot. Besides TDMA, space division multiple access (SDMA) enables simultaneous data transmission to multiple users, such that the hovering time and hovering energy can be reduced. In [6], the authors designed an SDMA-based beamforming scheme to minimize the total transmit power for multi-antenna UAVs. In [7], an energy efficiency maximization problem was investigated in an SDMA-based multi-antenna UAV network via optimizing the flying velocity and power allocation. However, serving multiple users simultaneously may lead to strong inter-user interference and may require more communication energy to fulfill users' demands. In [8], [9], two non-convex combinatorial optimization problems based on SDMA were studied. The authors in [8] proposed an alternative optimization algorithm based on the block coordinate descent method to optimize frequency, transmit power, and UAVs trajectory. In [9], the non-convex problem becomes convex by fixing a sensing-time variable. Then, a single-variable search method was adopted to optimize.

Deterministic optimization algorithms, e.g., [2]–[9], might not be suitable for fast decision making in a dynamic

Manuscript received June 22, 2020; revised January 20, 2021; accepted April 6, 2021. Date of publication April 27, 2021; date of current version June 9, 2021. This work was supported in part by the ERC project AGNOSTIC under Grant 742648, in part by the FNR CORE projects ROSETTA under Grant C17/IS/11632107, ProCAST under Grant C17/IS/11691338, in part by the 5G-Sky under Grant C19/IS/13713801, in part by the FNR bilateral project LARGOS (12173206), and in part by by the Jiangsu Natural Science Foundation under Grant SBK2018040630. A part of this paper was presented at IEEE EuCNC, Jun. 2020 [21]. The review of this article was coordinated by Prof. Rose Qingyang Hu. *(Corresponding author: Lei Lei.)*

Yaxiong Yuan, Lei Lei, Thang X. Vu, Symeon Chatzinotas, and Björn Ottersten are with the Interdisciplinary Centre for Security, Reliability and Trust, Luxembourg University, L-1855 Kirchberg, Luxembourg (e-mail: yaxiong.yuan@uni.lu; lei.lei@uni.lu; thang.vu@uni.lu; symeon.chatzinotas@uni.lu; bjorn.ottersten@uni.lu).

Sumei Sun is with the Institute for Infocomm Research, Agency for Science, Technology, and Research, Singapore 138632, Singapore (e-mail: sunsm@i2r.a-star.edu.sg).

Digital Object Identifier 10.1109/TVT.2021.3075860

wireless environment. To address this issue, deep learning-based solutions have been investigated in the literature. The authors in [10] relied on a deep neural network (DNN) to efficiently predict the resource allocation scheme for mobile edge networks. In [11], a deep learning-based auction algorithm was proposed to determine a dynamic battery charging scheduling for UAV-aided systems. Supervised learning, such as DNN, requires large amounts of training data, which is a non-trivial task in an offline manner [12]. Another category is reinforcement learning (RL). In [13], Q-learning was applied to solve an unconstrained UAVs trajectory design problem to avoid collisions. However, updating the Q-table results in unaffordable computing time/resources. To improve efficiency, deep reinforcement learning (DRL) was developed with the following advantages. Firstly, DRL provides timely solutions, adapted to environment variations. Secondly, DRL integrates DNN to make decisions and improve solution quality. Thirdly, DNN requires an offline data generating and training phase, whereas DRL is less needed for prior knowledge and is able to train by exploring unknown environments and exploiting received feedbacks in an online manner. In [14], the authors applied a deep Q network (DQN) to design an energy-efficient flying trajectory scheme for UAV-aided networks. In general, DQN is used to deal with a relatively small and discrete action space, where the action space refers to the set of all possible decisions [15]. The authors in [16] designed a different deep Q-learning architecture with a high dimensional action space, but it needs to evaluate all of the actions before making a decision, which is time-consuming.

Deep actor-critic is an emerging DRL method with fast convergent properties and the capability to deal with a large action space [17]. In [18], an actor-critic-based DRL (AC-DRL) algorithm was proposed to reduce the UAV's energy consumption and enhance the UAV's coverage of ground users via optimizing UAV's flying direction and distance. In [19], the authors employed deep actor-critic to design a learning algorithm for UAV-aided systems, considering energy efficiency and users' fairness. Note that the AC-DRL in [18], [19] was developed for unconstrained problems. However, most of the problems in UAV systems are constrained and with discrete variables. The conventional AC-DRL algorithms have limitations on tackling constrained combinatorial optimization problems, which may result in slow convergent, infeasible, and degraded solutions. The authors in [20], [21] developed AC-DRL algorithms for a combinatorial optimization problem in a UAV-aided system, where the performance is limited when the problem scale grows. In [22], [23], two AC-DRL algorithms based on deep deterministic policy gradient (DDPG) were developed to optimize UAV trajectory and resource allocation. The adopted reward function can satisfy simple constraints but might not be applicable for complicated combinatorial problems.

In this study, we minimize the UAV's communication and propulsion energy in a downlink UAV-aided communication system. The improvement of solution development lies in three aspects. Firstly, compared to offline optimization approaches, we provide online learning and timely energy-saving solutions based on DRL. Secondly, unlike the conventional DRL or AC-DRL methods, the proposed solution is suited to tackle constrained combinatorial optimization. Thirdly, compared to our previous work [21], we augment the algorithms by developing new theoretical results and tailored approaches to address two challenging issues in guaranteeing feasibility and controlling exponentially-increased action space. The major contributions are summarized as follows:

- We formulate an energy minimization problem for an SDMA-enabled UAV communication system, where user-timeslot allocation and UAV's hovering time assignment are the coupled optimization tasks. The formulated problem is combinatorial and non-convex with bilinear constraints.
- We provide a relax-and-approximate method to approach the optimum. That is, the bilinear terms are addressed by McCormick envelopes relaxation, then the remaining integer linear programming problem is solved by the branch-and-bound (B&B) algorithm.
- We characterize the interplay among communication energy, hovering time, and hovering energy. Based on the derived analytical results, we develop a golden section search-based heuristic (GSS-HEU) algorithm for benchmarking general instances with lower complexity than the optimal solution.
- We propose an actor-critic-based deep stochastic online scheduling (AC-DSOS) algorithm for UAV energy savings, where the original problem is transformed into a Markov decision process (MDP). AC-DSOS is computationally light and solves the problem in an online manner, against offline high-complexity optimal/sub-optimal algorithms. Unlike conventional DRL, we develop a set of tailored approaches in AC-DSOS, e.g., stochastic policy quantification, action space reduction, and feasibility-guaranteed reward function design, to overcome DRL's limitations in addressing combinatorial optimization problems with multiple constraints and large action space.
- Simulations demonstrate that the proposed AC-DSOS enables a feasible, fast-converging, and dynamically adaptive solution. The designed approaches are effective in reducing action space and guaranteeing feasibility. AC-DSOS achieves promising energy-saving performance compared with two recent AC-DRL methods and three heuristic algorithms.

The rest of the paper is organized as follows. Section II provides the system model and Section III formulates the considered optimization problem. In Section IV, we analyze the relationship between the energy consumption and hovering time, and propose a heuristic algorithm. In Section V, we reformulate the problem as an MDP and develop an AC-DSOS algorithm. Numerical results are presented and analyzed in Section VI. Finally, we draw the conclusions in Section VII.

The codes for generating the results are online available at the link: https:// github.com/ ArthuretYuan .

II. System Model and Problem Formulation

A. System Model

We consider a downlink UAV-aided communication system. A UAV serves as an aerial base station (BS) to deliver data to ground users, e.g., for the scenarios if terrestrial BSs are

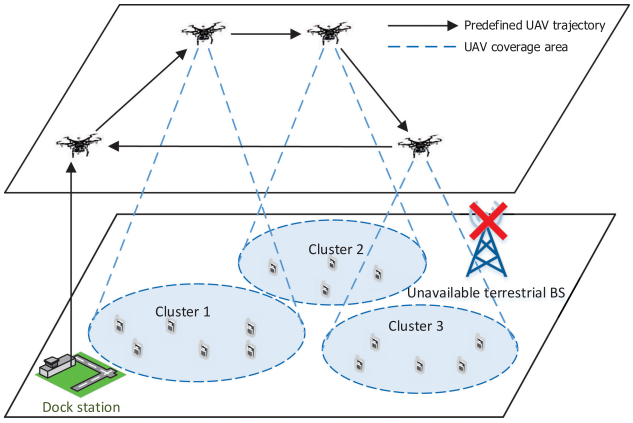

Fig. 1. An illustrative UAV-aided network.

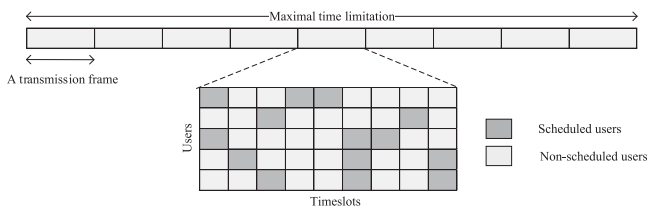

Fig. 2. An illustration of the frame-timeslot structure.

unavailable or overloaded by high traffic demand from numerous users. We assume that the UAV is equipped with L antennas and each ground user has a single antenna [7]. The UAV is fully loaded with data and energy at a dock station before the task starts. The service area is divided into N clusters considering the UAV's limited coverage area. This setup can be used in many practical scenarios such as emergency rescue and temporary communication [24], [25]. We denote $\mathcal{N} = \{1, \ldots, n, \ldots, N\}$ as the set of clusters and $\mathcal{N}^+ = \mathcal{N} \cup \{N+1\}$ as the extended set, where the $(N+1)$-th cluster denotes the dock station. The UAV flies through all the clusters successively according to a pre-optimized trajectory, and transmits data to the users when hovering at a given point, e.g., above the cluster's center. Let K_n and $\mathcal{K}_n$ denote the number and set of the users in the n-th cluster. The demands of user $k \in \mathcal{K}_n$ are denoted by $q_{k,n}$ (in bits). When all the demands in a cluster are satisfied, the UAV leaves the current cluster and visits the next one. After serving all the clusters, the UAV flies back to the dock station. The process of the UAV from leaving to returning the dock station is defined as a round or a task. Fig. 1 illustrates an example of the considered system.

The data stored in the UAV typically have a certain life span [26]. Thus, we consider the transmitted data are delay-sensitive, and all data delivery must be completed within T_{max} (in frames), where the time domain is divided by frames in set $\mathcal{T} = \{1, \ldots, t, \ldots, T_{max}\}$. One frame consists of I timeslots, and the duration of a timeslot is Φ. With SDMA, the UAV can simultaneously transmit data to more than one user in each timeslot. The frame-timeslot structure is shown in Fig. 2, where the shaded blocks indicate that the users are scheduled. We define the scheduled users at a timeslot as a user group. The union of the possible groups in cluster n is denoted by $\mathcal{G}_n = \{1, \ldots, g, \ldots, G_n\}$. The maximum number of candidate groups in cluster n is $G_n = 2^{K_n} - 1$ [27], which increases exponentially with K_n. The number and set of the users of group g in cluster n are denoted by $K_{g,n}$ and $\mathcal{K}_{g,n}$, respectively.

The channel vector from the UAV antennas to ground user $k \in \mathcal{K}_n$ is denoted as $\mathbf{h}_{k,n} \in \mathbb{C}^{1\times L}$, which can be expressed by $\boldsymbol{\alpha}_{k,n} 10^{-\xi_{k,n}/10}$, where $\boldsymbol{\alpha}_{k,n} \in \mathbb{C}^{1\times L}$ is the multipath Rician fading vector and $\xi_{k,n}$ is the free-space propagation loss between the UAV and ground user $k \in \mathcal{K}_n$. The model comprises a deterministic LoS component and a random multi-path component. The former predicts the propagation loss of a signal encounters air-to-ground scenarios, and the latter captures the effects of reflection, scattering, and diffraction by the ground obstacles. The model is suited and widely adopted for UAV applications in urban/suburban scenarios, e.g., [7], [13], [20], [24], [28], [29]. We collect all the channel vectors of the users in $\mathcal{K}_{g,n}$ to form a matrix $\mathbf{H}_{g,n} \in \mathbb{C}^{K_{g,n}\times L}$. Within a user group, we apply a linear minimum mean square error (MMSE) precoding scheme due to its high efficiency and low computational complexity in mitigating intra-group interference. The precoding vector for user $k \in \mathcal{K}_{g,n}$ is calculated by:

$$\mathbf{w}_{k,g,n} = \sqrt{p_{k,g,n}} \frac{\tilde{\mathbf{h}}_{k,g,n}}{\|\tilde{\mathbf{h}}_{k,g,n}\|}, \tag{1}$$

where $p_{k,g,n}$ is the transmit power for user k in group g, $\tilde{\mathbf{h}}_{k,g,n}$ is the k-th column in $\mathbf{H}_{g,n}^H(\sigma^2\mathbf{I} + \mathbf{H}_{g,n}\mathbf{H}_{g,n}^H)^{-1}$, and σ^2 is the noise power. Note that transmit power $p_{k,g,n}$ is fixed as parameters in this work by following practical UAV applications, e.g., constant transmit power can be selected from 0.1 W to 10 W [30]. The signal-to-interference-plus-noise ratio (SINR) for the user $k \in \mathcal{K}_{g,n}$ is given by:

$$\Gamma_{k,g,n} = \frac{\beta_{g,n}^{(kk)} p_{k,g,n}}{\sum_{j\in\mathcal{K}_{g,n}\setminus\{k\}} \beta_{g,n}^{(kj)} p_{j,g,n} + \sigma^2}, \quad k \in \mathcal{K}_{g,n},\ g \in \mathcal{G}_n, \tag{2}$$

where $\beta_{g,n}^{(kk)} = |\mathbf{h}_{k,n}\tilde{\mathbf{h}}_{k,g,n}|^2$ and $\beta_{g,n}^{(kj)} = |\mathbf{h}_{k,n}\tilde{\mathbf{h}}_{j,g,n}|^2$ are the effective channel gains.

We assume the channel state information (CSI) updates over frames, and the channel states keep static within a transmission frame. Based on the adopted path-loss and Rician-fading model, we further model the time-varying channel as the first state Markov channel (FSMC) to capture the time-correlation characteristics and allow mathematically tractable analysis. Given the Rician probability density function and the corresponding auto-correlation function, we can discretize the channel into several intervals and derive the transition probabilities. Thus, knowing the initial channel state (sampling by Rician distribution), the following channel states can be forecasted by the transition probabilities [31]. By FSMC modeling, the channel quality in the near future can be forecast based on the knowledge of previous channel conditions. Moreover, FSMC is efficient in quick simulations and system performance evaluations [32], [33]. Since CSI varies over frames, we use $\Gamma_{k,g,n,t}$, $\beta_{g,n,t}^{(kk)}$ and $\beta_{g,n,t}^{(kj)}$ to track SINR and channel coefficients on the t-th frame. We quantify each coefficient $\beta_{g,n,t}^{(kk)}$ and $\beta_{g,n,t}^{(kj)}$ to multiple Markov states and obtain a transition probability such that the

variations of $\beta_{g,n,t}^{(kk)}$ and $\beta_{g,n,t}^{(kj)}$ follow a Markov process between frames [34]. If group $g \in \mathcal{G}_n$ is scheduled at timeslot i on frame t, the amount of data transmitted to user $k \in \mathcal{K}_{g,n}$ and the consumed communication energy of group $g \in \mathcal{G}_n$ can be expressed by:

$$d_{k,g,n,t} = \Phi B \log_2 \left(1 + \Gamma_{k,g,n,t}\right), \quad k \in \mathcal{K}_{g,n},\ g \in \mathcal{G}_n,\ t \in \mathcal{T}, \tag{3}$$

and

$$e_{g,n,t} = \Phi \sum_{k \in \mathcal{K}_{g,n}} p_{k,g,n}, \quad g \in \mathcal{G}_n,\ t \in \mathcal{T}, \tag{4}$$

where B is the system bandwidth. Note that within a frame, we assume a user's channel condition is identical across all the timeslots, thus index i is omitted in $d_{k,g,n,t}$ and $e_{g,n,t}$.

B. UAV's Energy Model

We employ a UAV energy model proposed in [4]. The flying power is formulated as a function $f(U)$ of flying velocity U:

$$f(U) = P_0 \left(1 + \frac{3\,U^2}{U_{tip}^2}\right) + P_1 \left(\sqrt{1 + \frac{U^4}{4U_{ind}^4}} - \frac{U^2}{2U_{ind}^2}\right)^{\frac{1}{2}} + \frac{1}{2}\rho_1 \rho_2 U^3, \tag{5}$$

where

- P_0: the blade profile power in hovering status;
- P_1: the induced power in hovering status;
- U_{tip}: the tip speed of the rotor blade;
- U_{ind}: the mean rotor induced velocity;
- ρ_1: the parameter related to the fuselage drag ratio, rotor solidity, and the rotor disc area;
- ρ_2: the air density.

When UAV approaches the hovering point of each cluster, it will fly around the point with a certain velocity $U = U_{hov}$, which is more energy-efficient than $U = 0$ [5]. Thus, the hovering power P_H is $f(U = U_{hov})$. The flying energy with constant velocity U and traveling distance S is expressed as:

$$f(U) \cdot S/U = SP_0 \left(\frac{1}{U} + \frac{3\,U}{U_{tip}^2}\right) + SP_1 \left(\sqrt{\frac{1}{U^4} + \frac{1}{4U_{ind}^4}} - \frac{1}{2U_{ind}^2}\right)^{\frac{1}{2}} + \frac{S}{2}\rho_1 \rho_2 U^2. \tag{6}$$

Hovering energy and communication energy need to be jointly optimized since they are coupled by hovering time, whereas the optimization of flying energy is independent. By applying graph-based numerical methods [35], the minimum flying energy E_F^* along with the optimal flying speed U_F^* can be obtained by:

$$E_F^* = f(U_F^*) \cdot S/U_F^*, \tag{7}$$

where $U_F^* = \text{argmin}_{U \geq 0} \frac{f(U)}{U}$.

The main notations are summarized in Table I.

TABLE I
SUMMARY OF SYMBOLS AND NOTATIONS

Notation	Description
$N, \mathcal{N}$	number and set of clusters
L	number of antennas in UAV
$K_n, \mathcal{K}_n$	number and set of users in cluster n
$G_n, \mathcal{G}_n$	number and set of groups in cluster n
$K_{g,n}, \mathcal{K}_{g,n}$	number and set of users in group g of cluster n
$q_{k,n}$	demands of user k in cluster n
$T_{max}, \mathcal{T}$	maximum number and set of frames in each round
$I, \mathcal{I}$	number and set of timeslots in each frame
Φ	duration of each timeslot (in seconds)
$\Gamma_{k,g,n,t}$	SINR of user $k \in \mathcal{K}_{g,n}$ on frame t
$\beta_{g,n,t}^{(kj)}$	channel coefficient from user j's precoding vector to user k ($k, j \in \mathcal{K}_{g,n}$) on frame t
$d_{k,g,n,t}$	transmitted data of user $k \in \mathcal{K}_{g,n}$ per timeslot on frame t
$e_{g,n,t}$	communication energy of group $g \in \mathcal{G}_n$ per timeslot on frame t
U_F^*	UAV's flying velocity that minimizes flying energy with a predetermined flying path
E_F^*	minimal flying energy with a predetermined flying path

III. PROBLEM FORMULATION

We denote binary variables $\lambda_{i,g,n,t} \in \{0, 1\}$ as the scheduling indicator, where $\lambda_{i,g,n,t} = 1$ indicates that user group $g \in \mathcal{G}_n$ is assigned to timeslot i on frame t and $\lambda_{i,g,n,t} = 0$ otherwise. Another class of binary variables $\nu_{n,t} \in \{0, 1\}$ indicates that the UAV is hovering above cluster n on frame t ($\nu_{n,t} = 1$), and $\nu_{n,t} = 0$ otherwise. The UAV energy consumption consists of flying energy E_F, hovering energy E_H, and communication energy E_C. Since the minimal flying energy E_F^* can be independently obtained by Eq. (7) without loss of optimality, the objective focuses on joint optimization of E_C and E_H, which are expressed by:

$$E_C = \sum_{t=1}^{T_{max}} \sum_{n=1}^{N} \sum_{g=1}^{G_n} \sum_{i=1}^{I} \nu_{n,t} \lambda_{i,g,n,t} e_{g,n,t}, \tag{8}$$

$$E_H = \sum_{t=1}^{T_{max}} \sum_{n=1}^{N} \Phi I P_H \nu_{n,t}. \tag{9}$$

Note that the UAV is battery-limited in practice. We focus on the instances that the minimum consumed energy in (10a) is within the UAV's battery storage, otherwise, the task is infeasible. The optimization problem is formulated as:

$$\mathcal{P}_1: \min_{\substack{\lambda_{i,g,n,t}, \\ \nu_{n,t}}} \quad E_C + E_H \tag{10a}$$

s.t.

$$\sum_{t=1}^{T_{max}} \sum_{g=1}^{G_n} \sum_{i=1}^{I} \nu_{n,t} \lambda_{i,g,n,t} d_{k,g,n,t} \geq q_{k,n},\ \forall k \in \mathcal{K}_n,\ n \in \mathcal{N}, \tag{10b}$$

$$\nu_{n,t} \leq \nu_{n,t+1} + \nu_{n+1,t+1},\ \forall n \in \mathcal{N},\ t \in \mathcal{T}, \tag{10c}$$

$$\sum_{g=1}^{G_n} \sum_{i=1}^{I} \lambda_{i,g,n,t} = I \cdot \nu_{n,t},\ \forall n \in \mathcal{N}^+,\ t \in \mathcal{T}, \tag{10d}$$

$$\sum_{g=1}^{G_n} \lambda_{i,g,n,t} \leq 1, \ \ \forall i \in \mathcal{I},\ n \in \mathcal{N}^+,\ t \in \mathcal{T}, \tag{10e}$$

$$\sum_{n=1}^{N+1} \nu_{n,t} = 1,\ \forall t \in \mathcal{T}, \tag{10f}$$

$$\lambda_{i,g,n,t} \in \{0,1\},\ \forall i \in \mathcal{I},\ g \in \mathcal{G}_n,\ n \in \mathcal{N}^+,\ t \in \mathcal{T}, \tag{10g}$$

$$\nu_{n,t} \in \{0,1\},\ \forall n \in \mathcal{N}^+,\ t \in \mathcal{T}. \tag{10h}$$

Constraints (10b) guarantee that all the users' requests have to be satisfied within T_{max}. Constraints (10c) define that the UAV follows a successive and forward manner in visiting clusters. For example, if the UAV is hovering above cluster n on frame t, in the next frame $t+1$, the UAV either chooses to stay at the current cluster n or move to the next cluster $n+1$. The option of flying back to previously visited clusters, e.g., $n-1$, is thus excluded. Note that the UAV takes off from the first cluster, i.e., $\nu_{1,1} = 1$. Constraints (10d) represent that all the timeslots on frame t are assigned to a user group when $\nu_{n,t} = 1$, otherwise, no users are scheduled in any timeslot. Constraints (10e) and (10f) indicate that no more than one group can be scheduled at a timeslot and only one cluster can be served within a frame. Constraints (10g) and (10h) confine variables $\lambda_{i,g,n,t}$ and $\nu_{n,t}$ to binary.

Note that $\mathcal{P}_1$ is a combinatorial optimization problem with a non-convex bilinear objective and constraints. The optimum can be approached by a well-established relax-and-approximate method. That is, the non-convex bilinear terms are relaxed and bounded by McCormick envelopes [36], where each variable ($\lambda_{i,g,n,t}$ and $\nu_{n,t}$) is bounded by an upper and a lower bound. The relaxation problem becomes an integer linear programming (ILP) problem which can be optimally solved by B&B. Overall, the optimum of $\mathcal{P}_1$ can be approached by ultimately tightening the bounds, e.g., increase the number of breakpoints in the envelopes, but this results in exponentially increasing complexity which is unaffordable in practice [37]. Thus, we adopt the above relax-and-approximate method to provide an optimal solution for benchmarking small-medium cases. For general cases, we propose a heuristic algorithm in the next section.

IV. Heuristic Approach

We decompose the joint optimization to two sub-problems, i.e., user-timeslot and hovering time allocation, corresponding to optimization of $\lambda_{i,g,t,n}$ and $\nu_{n,t}$, respectively. We then solve one sub-problem when the other is fixed.

A. User-Timeslot Scheduling

The bilinear items are resolved with the fixed $\nu_{n,t}$. The number of frames at each cluster is determined by:

$$t_n = \sum_{t=1}^{T_{max}} \nu_{n,t},\ \forall n \in \mathcal{N}, \tag{11}$$

and $\Phi I t_n$ is the hovering duration. The user-timeslot scheduling can be carried out independently in each cluster, and the resulting problem for the n-th cluster is formulated in $\mathcal{P}_2(n)$ with a given t_n. We denote $E_{H,n}$ and $E_{C,n}$ as the hovering and communication energy for the n-th cluster:

$$E_{H,n} = \Phi I P_H t_n, \tag{12}$$

$$E_{C,n} = \sum_{t=\tau_n+1}^{\tau_n+t_n} \sum_{g=1}^{G_n} \sum_{i=1}^{I} \lambda_{i,g,n,t} e_{g,n,t}, \tag{13}$$

where τ_n refers to the number of elapsed frames before the UAV arriving cluster n, which can be calculated by:

$$\tau_n = \sum_{t=1}^{T_{max}} \sum_{n'=1}^{n-1} \nu_{n',t}. \tag{14}$$

The sub-problem $\mathcal{P}_2(n)$ is formulated as:

$$\mathcal{P}_2(n): \min_{\lambda_{i,g,n,t}} \ E_{C,n} + E_{H,n} \tag{15a}$$

s.t.

$$\sum_{t=\tau_n+1}^{\tau_n+t_n} \sum_{g=1}^{G_n} \sum_{i=1}^{I} \lambda_{i,g,n,t} d_{k,g,n,t} \geq q_{k,n},\ \forall k \in \mathcal{K}_n, \tag{15b}$$

$$\sum_{g=1}^{G_n} \sum_{i=1}^{I} \lambda_{i,g,n,t} = I,\ \forall t \in \{\tau_n+1, \ldots, \tau_n+t_n\}, \tag{15c}$$

$$\sum_{g=1}^{G_n} \lambda_{i,g,n,t} \leq 1, \ \ \forall i \in \mathcal{I},\ t \in \mathcal{T}, \tag{15d}$$

$$\lambda_{i,g,n,t} \in \{0,1\},\ \forall i \in \mathcal{I},\ g \in \mathcal{G}_n,\ t \in \mathcal{T}. \tag{15e}$$

$\mathcal{P}_2(n)$ is a multi-choice multi-dimensional knapsack problem (MMKP), which can be solved by a guided local search (GLS)-based heuristic algorithm with high-quality sub-optimal solutions and pseudo-polynomial-time complexity [38].

B. Hovering Time Allocation

To optimize hovering time efficiently, we first investigate the connection between the objective energy and t_n. From Eq. (12) and Eq. (13), $E_{H,n}$ increases linearly with t_n while $E_{C,n}$ is determined by both t_n and $\lambda_{i,g,n,t}$. Next, we show the relationship between the optimum $E_{C,n}$ and t_n. For cluster n, we denote $E^*_{C,n}(t_n)$ as the communication energy with the optimal scheduling decision $\lambda^*_{i,g,n,t}$ at a given hovering time t_n.

Lemma 1: $E^*_{C,n}(t_n)$ is a non-increasing function of t_n,

$$E^*_{C,n}(\hat{t}) \geq E^*_{C,n}(\hat{t}+\Delta t),\ \hat{t} > 0, \Delta t > 0. \tag{16}$$

Proof: We denote the optimal user scheduling for $\mathcal{P}_2(n)|_{t_n=\hat{t}}$ as $\lambda^*_{i,g,n,t}$. If t_n increases from $\hat{t}$ to $\hat{t}+\Delta t$, $\lambda^*_{i,g,n,t}$ is still feasible for $\mathcal{P}_2(n)|_{t_n=\hat{t}+\Delta t}$ such that

$$\begin{aligned} E^*_{C,n}(\hat{t}) &= \sum_{t=\tau_n+1}^{\tau_n+\hat{t}} \sum_{g=1}^{G_n} \sum_{i=1}^{I} \lambda^*_{i,g,n,t} e_{g,n,t} \\ &= E'_{C,n}(\hat{t}+\Delta t) = \sum_{t=\tau_n+1}^{\tau_n+\hat{t}+\Delta t} \sum_{g=1}^{G_n} \sum_{i=1}^{I} \lambda^*_{i,g,n,t} e_{g,n,t}. \end{aligned} \tag{17}$$

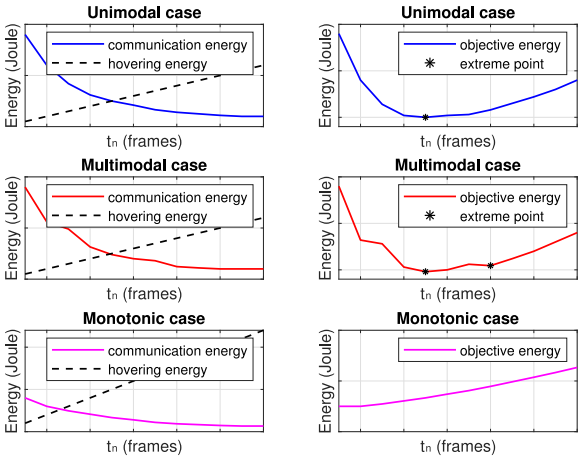

Fig. 3. Energy curves for three possible cases.

$\lambda^*_{i,g,n,t}$ might not be necessarily optimal for $t_n = \hat{t} + \Delta t$. There exists an optimal scheduling resulting in lower communication energy, i.e.,

$$E^*_{C,n}(\hat{t} + \Delta t) \le E'_{C,n}(\hat{t} + \Delta t) = E^*_{C,n}(\hat{t}). \tag{18}$$

Thus the conclusion. ■

From Lemma 1, we can observe that $E^*_{C,n}(t_n)$ is an non-increasing function of t_n, i.e., $\frac{dE^*_{C,n}(t_n)}{dt_n} \le 0$. For $E_{H,n}(t_n)$, we can derive that $\frac{dE_{H,n}(t_n)}{dt_n} = \Phi I P_H$ based on Eq. (12). Thus, the extreme point of $E^*_{C,n}(t_n) + E_{H,n}(t_n)$ can be obtained at $t_n = t^\dagger$ when

$$\frac{dE^*_{C,n}(t_n)}{dt_n}|_{t_n = t^\dagger} = -\Phi I P_H. \tag{19}$$

Since the existence and the number of extreme points are undetermined. There are three possible cases, i.e., unimodal, multimodal, and monotonic, for $E^*_{C,n}(t_n) + E_{H,n}(t_n)$, as illustrated in Fig. 3. In case 1, the curve is a unimodal function with only one extreme point. In case 2, the fluctuation of $\frac{dE^*_{C,n}(t_n)}{dt_n}$ leads to multiple extreme points such that the curve is a multimodal function. In case 3, Eq. (19) cannot hold, e.g., $\frac{dE^*_{C,n}(t_n)}{dt_n}$ is consistently lager than $-\Phi I P_H$, so the curve is monotonously increasing with no extreme point.

Observing the possible cases, we employ an efficient golden section search (GSS) to find the extreme points [39]. In GSS, we limit the hovering time $t_n \le \bar{t}_n$ to ensure that the total service duration does not exceed T_{max}, where $\bar{t}_n$ is a maximal time limitation for cluster n. Intuitively, the clusters with more demands need more transmission frames. We assume $\bar{t}_n$ is proportional to the users' demands:

$$\bar{t}_n = T_{max} \frac{\sum_{k=1}^{K_n} q_{k,n}}{\sum_{n=1}^{N} \sum_{k=1}^{K_n} q_{k,n}}. \tag{20}$$

C. Algorithm Summary

We summarize the proposed GSS-HEU in Alg. 1. We denote $\mathcal{B}_{n,t}$ as the set of channel states of cluster n on frame t, which is expressed as:

$$\mathcal{B}_{n,t} = \{\beta^{(kj)}_{1,n,t}, \ldots, \beta^{(kj)}_{G_n,n,t} | \ \forall k, j \in \mathcal{K}_{g,n}\}. \tag{21}$$

Algorithm 1: GSS-HEU Algorithm

Inputs:
Users' demands: $q_{1,1}, \ldots, q_{K_1,1}, \ldots, q_{1,N}, \ldots, q_{K_N,N}$;
Channel states: $\mathcal{B}_{1,1}, \ldots, \mathcal{B}_{1,T_{max}}, \ldots, \mathcal{B}_{N,1}, \ldots, \mathcal{B}_{N,T_{max}}$;
Search range's upper bound: $\bar{t}_1, \ldots, \bar{t}_N$.
Outputs:
Heuristic solution: $\lambda^*_{1,1,1,1}, \ldots, \lambda^*_{I,G_n,N,T_{max}}, t^*_1, \ldots, t^*_N$

1: **for** $n = 1; n \le N; n++$ **do**
2: $x_1 = 0; y_1 = \bar{t}_n$;
3: $u_1 = \lceil y_1 - 0.618(y_1 - x_1) \rceil$;
4: $v_1 = \lceil x_1 + 0.618(y_1 - x_1) \rceil$;
5: **for** $m = 1; |y_m - x_m| > 1; m++$ **do**
6: Solve $\mathcal{P}_2(n)|_{t_n = u_m}$ and $\mathcal{P}_2(n)|_{t_n = v_m}$;
7: Obtain the corresponding user scheduling schemes $\lambda_{i,g,n,t}|_{t_n = u_m}$ and $\lambda_{i,g,n,t}|_{t_n = v_m}$;
8: Obtain the objective energy $(E_{C,n} + E_{H,n})|_{t_n = u_m}$ and $(E_{C,n} + E_{H,n})|_{t_n = v_m}$;
9: **if** $(E_{C,n} + E_{H,n})|_{t_n = u_m} < (E_{C,n} + E_{H,n})|_{t_n = v_m}$ **then**
10: $x_{m+1} = x_m;\ y_{m+1} = v_m;\ v_{m+1} = u_m;$ $u_{m+1} = \lceil y_{m+1} - 0.618(y_{m+1} - x_{m+1}) \rceil$;
11: **else**
12: $x_{m+1} = u_m;\ y_{m+1} = y_m;\ u_{m+1} = v_m;$ $v_{m+1} = \lceil y_{m+1} - 0.618(y_{m+1} - x_{m+1}) \rceil$;
13: **end if**
14: **end for**
15: $t^*_n = v_m$; $\lambda^*_{i,g,n,t} = \lambda_{i,g,n,t}|_{t_n = v_m}$.
16: **end for**

In GSS-HEU, the initial search range of GSS $[x_1, y_1]$ is set as $[0, \bar{t}_n]$, which is partitioned into three sections by two points u_1 and v_1 with the golden ratio 0.618 in lines 2-4, where $\lceil . \rceil$ is an operation to round a value up to an integer. When a hovering time is searched in GSS, e.g., $t_n = u_m$ or $t_n = v_m$, the corresponding user-timeslot allocation is obtained by solving $\mathcal{P}_2(n)$ in line 6. In lines 9-13, we compare the objective energy and update the search range. The search process terminates at $|y_m - x_m| \le 1$. The selected hovering time t^*_n is v_m and the corresponding scheduling scheme $\lambda^*_{i,g,n,t}$ is $\lambda_{i,g,n,t}|_{t_n = v_m}$.

The complexity of GSS-HEU is $\mathcal{O}(\sum_{n=1}^{N} G_n^2 \times \max\{K_n, I\bar{t}_n\} \times \log(2\bar{t}_n))$, which is much lower than that of the optimal method. However, both the optimal and GSS-HEU approaches may have limitations in fast decision-making. The computational time for both algorithms grows exponentially with the number of users since $G_n = 2^{K_n} - 1$ [12]. In addition, both algorithms need the estimated and complete channel states for the whole task frames, i.e., from $t = 1$ to T_{max}. This may result in difficulties in channel estimation. Therefore, we reconsider $\mathcal{P}_1$ from the perspective of DRL to enable the UAV to make decisions intelligently, while the developed optimal and heuristic algorithms are used to benchmark the performance of learning-based solutions.

V. Actor-Critic-Based DRL Algorithm

A. Overview of Actor-Cirtic-Based DRL (AC-DRL)

In DRL, an agent learns to make decisions by exploring the unknown environments and exploiting the received feedbacks. At each learning step[1] t, the agent observes the current state $\boldsymbol{s}_t$ and takes an action $\boldsymbol{a}_t$ based on a policy. Then, a reward r_t will be fed back to the agent. The policy will be updated step by step according to the feedback. Actor-critic is an emerging reinforcement learning method that separates the agent into two parts, an actor and a critic. The actor is responsible for taking actions following a stochastic policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$, where $\pi(.|.)$ refers to a conditional probability density function. The critic is used to evaluate the decisions via a Q-value, which is given by:

$$Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t) = \mathbb{E}_{\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)}[R_t|\boldsymbol{s}_t, \boldsymbol{a}_t], \tag{22}$$

where $\mathbb{E}_{\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)}[.|.]$ is a conditional expectation under the policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$, and R_t is the cumulative discounted reward with a discount factor γ, which can be expressed as:

$$R_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}, \quad \gamma \in [0, 1]. \tag{23}$$

However, obtaining the explicit expressions of $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$ and $Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ is difficult. DRL uses DNNs as the parameterized approximators to provide estimations for $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$ and $Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$. We denote $\boldsymbol{\theta}_t$ and $\boldsymbol{\omega}_t$ as the parameter vectors for the actor and critic, and $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t; \boldsymbol{\theta}_t)$ and $Q^{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\omega}_t)$ as the corresponding parameterized functions[2]. The goal of the agent is to minimize the loss function of the actor $-J(\boldsymbol{\theta}_t)$:

$$-J(\boldsymbol{\theta}_t) = -\mathbb{E}[Q^{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\omega}_t)]. \tag{24}$$

Based on the fundamental results of the policy gradient theorem [15], the gradient of $J(\boldsymbol{\theta}_t)$ can be calculated by:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t) = \mathbb{E}[\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t|\boldsymbol{s}_t; \boldsymbol{\theta}_t) Q^{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\omega}_t)]. \tag{25}$$

The update rule of $\boldsymbol{\theta}_t$ can be derived based on gradient descent:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_a \cdot (-\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)), \tag{26}$$

where α_a is the learning rate of the actor. For the critic, the parameter vector $\boldsymbol{\omega}_t$ is updated based on temporal-difference (TD) learning [15]. In TD learning, the loss function of the critic $C_{_Q}(\boldsymbol{\omega}_t)$ is defined as the expectation of the square of TD error $\delta_{_Q}(\boldsymbol{\omega}_t)$, i.e., $\mathbb{E}[(\delta_{_Q}(\boldsymbol{\omega}_t))^2]$. The TD error $\delta_{_Q}(\boldsymbol{\omega}_t)$ refers to the difference between the TD target and estimated Q-value, which is given by:

$$\delta_{_Q}(\boldsymbol{\omega}_t) = r_t + \gamma Q^{\boldsymbol{\theta}}(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}; \boldsymbol{\omega}_t) - Q^{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\omega}_t), \tag{27}$$

where $r_t + \gamma Q^{\boldsymbol{\theta}}(\boldsymbol{s}_{t+1}, \boldsymbol{a}_{t+1}; \boldsymbol{\omega}_t)$ is the TD target. The objective of the critic is to minimize the loss function $C_{_Q}(\boldsymbol{\omega}_t)$ and the update rule of $\boldsymbol{\omega}_t$ can be derived by gradient descent:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \alpha_c \nabla_{\boldsymbol{\omega}} C_{_Q}(\boldsymbol{\omega}_t), \tag{28}$$

where α_c is the learning rate for the critic.

However, approximating $Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ brings about a large variance for the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)$, resulting in poor convergence [40]. To solve the problem, a V-value is introduced:

$$V^{\pi}(\boldsymbol{s}_t) = \mathbb{E}_{\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)}[R_t|\boldsymbol{s}_t]. \tag{29}$$

Approximating $V^{\pi}(\boldsymbol{s}_t)$ can reduce the variance. With the parameterized V-value $V^{\boldsymbol{\theta}}(\boldsymbol{s}_t; \boldsymbol{\omega}_t)$, the TD error and the loss function of the critic are expressed as:

$$\delta_{_V}(\boldsymbol{\omega}_t) = r_t + \gamma V^{\boldsymbol{\theta}}(\boldsymbol{s}_{t+1}; \boldsymbol{\omega}_t) - V^{\boldsymbol{\theta}}(\boldsymbol{s}_t; \boldsymbol{\omega}_t), \tag{30}$$

and

$$C_{_V}(\boldsymbol{\omega}_t) = \mathbb{E}[(\delta_{_V}(\boldsymbol{\omega}_t))^2]. \tag{31}$$

In addition, $\delta_{_V}(\boldsymbol{\omega}_t)$ provides an unbiased estimation of Q-value [40]. Thus, we can rewrite $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)$ in Eq. (25) as:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t) &= \mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \log(\pi(\boldsymbol{a}_t|\boldsymbol{s}_t; \boldsymbol{\theta}_t)) Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)\right] \\ &= \mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \log(\pi(\boldsymbol{a}_t|\boldsymbol{s}_t; \boldsymbol{\theta}_t)) \delta_{_V}(\boldsymbol{\omega}_t)\right]. \end{aligned} \tag{32}$$

B. Problem Reformulation

To apply AC-DRL, we reformulate $\mathcal{P}_1$ to an MDP problem, in which the UAV acts as an agent. We define the states, actions, and rewards as follows.

1) States: The system states $\boldsymbol{s}_t$ consist of the channel states for all the clusters on the current frame, i.e., $\mathcal{B}_{1,t}, \ldots, \mathcal{B}_{N,t}$, the undelivered demands, and the currently served cluster on frame t. The undelivered demands $b_{n,t}$ is the residual data to be delivered for cluster n on frame t:

$$b_{n,t+1} = b_{n,t} - d^{\pi}_{n,t}, \ \forall n \in \mathcal{N}, \ t \in \mathcal{T}, \tag{33}$$

$$b_{n,0} = \sum_{k=1}^{K_n} q_{k,n}, \ \forall n \in \mathcal{N}, \tag{34}$$

where $d^{\pi}_{n,t}$ is the delivered data for cluster n on frame t under the policy $\pi(\boldsymbol{s}_t|\boldsymbol{a}_t)$. We denote $o_t \in \mathcal{N}^+$ as an indicator to represent which cluster the UAV is serving on frame t. When the users requests in the current cluster are completed, the UAV will move to the next cluster on the next frame, otherwise, staying at the current cluster. For example, we assume that the UAV is hovering above cluster n on frame t, i.e., $o_t = n$. For the next frame, o_{t+1} is obtained by:

$$o_{t+1} = \begin{cases} n, & b_{n,t} > 0, \\ n+1, & b_{n,t} = 0. \end{cases} \tag{35}$$

When the UAV's duration exceeds T_{max}, the UAV will fly back to the dock station. By assembling the above three parts, the state $\boldsymbol{s}_t$ is defined as:

$$\boldsymbol{s}_t = [\mathcal{B}_{_{1,t}}, \ldots, \mathcal{B}_{_{N,t}}, b_{_{1,t}} \ldots, b_{_{N,t}}, o_t]. \tag{36}$$

Note that the elements of $\mathcal{B}_{n,t}$ are modeled as FSMC. In addition, based on Eq. (33) and Eq. (35), the next state of $b_{n,t}$ and o_t only depend on the current state and current policy. Therefore, the transition of the state $\boldsymbol{s}_t$ conforms to MDP [15].

[1] In this paper, a learning step is equivalent to a transmission frame.

[2] For simplicity, $Q^{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\omega}_t) = \mathbb{E}_{\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{s}_t; \boldsymbol{\theta}_t)}[R_t|\boldsymbol{s}_t, \boldsymbol{a}_t]$.

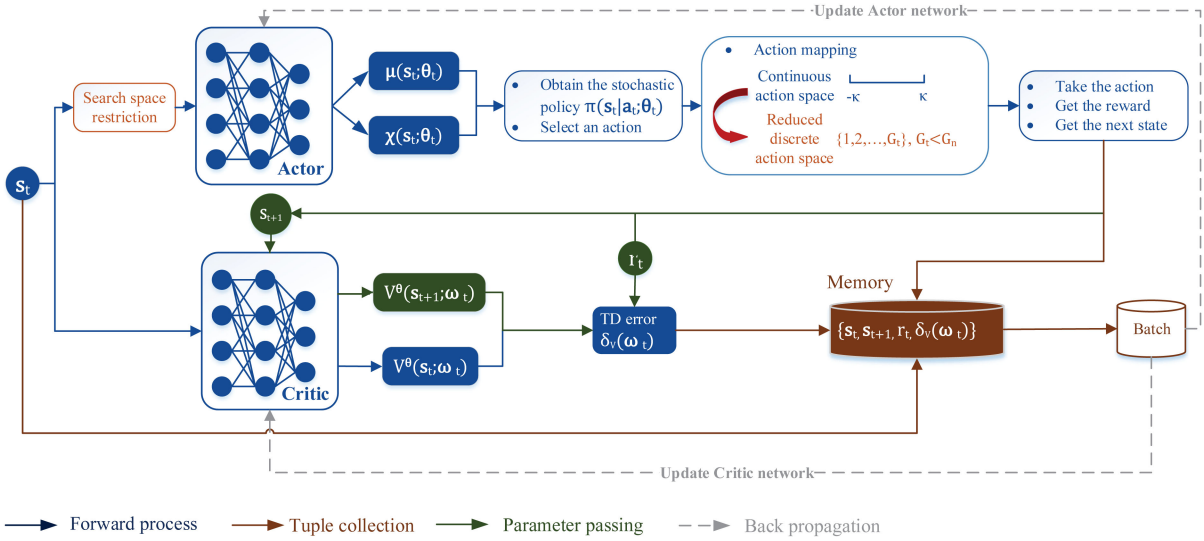

Fig. 4. The actor-critic framework of AC-DSOS.

2) Actions: The action of the UAV is the user-timeslot assignment on frame t, which is given by:

$$\boldsymbol{a}_t = [a_{1,t}, \ldots, a_{I,t}],$$
$$a_{i,t} \in \{1, \ldots, g, \ldots, G_n\},\ \forall i \in \mathcal{I},\ t \in \mathcal{T}, \tag{37}$$

where $a_{i,t} = g$ means the g-th group is selected at the i-th timeslot on the t-th frame. Note that the action space G_n can be huge since it increases exponentially with the number of users.

3) Rewards: The reward functions are commonly related to the objective of the problem. Conventionally, the reward function of $\mathcal{P}_1$ can be designed by Eq. (38) and Eq. (39), referring to [41] and [42]:

$$r_t = 1/e_t^{\pi}, \tag{38}$$
$$r_t = -e_t^{\pi}, \tag{39}$$

where e_t^{π} is the energy consumed on frame t under the policy $\pi(\boldsymbol{s}_t|\boldsymbol{a}_t)$. Since both the above reward functions monotonically decrease with e_t^{π}, the UAV updates the policy towards reducing energy consumption.

C. The AC-DSOS Algorithm

Conventional AC-DRL algorithms may not be able to deal with constrained discrete problems. Firstly, the combinatorial component of $\mathcal{P}_1$ limits the conventional AC-DRL in addressing huge discrete action spaces [43]. Secondly, the increased action space reduces the exploration efficiency in the learning process and degrades overall energy-saving performance. Thirdly, the conventional AC-DRL algorithms cannot guarantee the solution's feasibility in general. This means that a high-reward action can fail to satisfy the constraints in $\mathcal{P}_1$. To overcome the above difficulties and limitations, we propose an AC-DSOS algorithm that is tailored for constrained problems with discrete action representation. The basic actor-critic framework is employed in order to take the advantages of the stochastic policy and TD learning, where the stochastic policy can be quantified to tackle the issue of huge discrete spaces and TD learning can improve the learning efficiency.

Algorithm 2 AC-DSOS Algorithm

Inputs: The current state $\boldsymbol{s}_t$.
Outputs: The current action $\boldsymbol{a}_t$.
Initialize $\boldsymbol{\theta}_1$ and $\boldsymbol{\omega}_1$.
for each learning episode **do**
  Observe the initial state $\boldsymbol{s}_1$.
  for $t = 1 : T_{max}$ **do**
    Remove the groups containing the demand-satisfied users.
    Predicted mean $\boldsymbol{\mu}(\boldsymbol{s}_t;\boldsymbol{\theta}_t)$ and variance $\boldsymbol{\chi}(\boldsymbol{s}_t;\boldsymbol{\theta}_t)$ by the DNN of the actor.
    Obtain action's distribution $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t;\boldsymbol{\theta}_t)$ based on Gaussian distribution.
    Randomly choose $\hat{\boldsymbol{a}}_t$ following $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t;\boldsymbol{\theta}_t)$.
    Map the elements $\hat{a}_{i,t}$ to $a_{i,t}$ by Eq. (40).
    Take the after-mapped action $\boldsymbol{a}_t$.
    Obtain reward r_t by Eq. (47).
    Collect the next state $\boldsymbol{s}_{t+1}$.
    Approximate the value functions $V^{\boldsymbol{\theta}}(\boldsymbol{s}_t;\boldsymbol{\omega}_t)$ and $V^{\boldsymbol{\theta}}(\boldsymbol{s}_{t+1};\boldsymbol{\omega}_t)$ by the DNN of the critic.
    Calculate TD error $\delta_v(\boldsymbol{\omega}_t)$ by Eq. (30).
    Form and store a new tuple $\{\boldsymbol{s}_t, \boldsymbol{s}_{t+1}, r_t, \delta_v(\boldsymbol{\omega}_t)\}$.
    Obtain $\boldsymbol{\theta}_{t+1}$ and $\boldsymbol{\omega}_{t+1}$ by gradient descent.
    $\boldsymbol{s}_t = \boldsymbol{s}_{t+1}$; $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t+1}$; $\boldsymbol{\omega}_t = \boldsymbol{\omega}_{t+1}$.
  end for
end for

We illustrate the actor-critic framework of AC-DSOS in Fig. 4, where two DNNs work as the actor and critic, respectively. The stochastic policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$ is usually modeled as Gaussian distribution with a mean $\boldsymbol{\mu}(\boldsymbol{s}_t)$ and a variance $\boldsymbol{\chi}(\boldsymbol{s}_t)$ [44]. Given the current state $\boldsymbol{s}_t$, the actor does not predict $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t;\boldsymbol{\theta}_t)$ directly but obtains approximations of the mean $\boldsymbol{\mu}(\boldsymbol{s}_t;\boldsymbol{\theta}_t)$ and

the variance $\boldsymbol{\chi}(\boldsymbol{s}_t;\boldsymbol{\theta}_t)$. An action $\boldsymbol{a}_t$ can be selected based on $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t;\boldsymbol{\theta}_t)$. Then, the agent receives a reward r_t after taking the action and collects the next state $\boldsymbol{s}_{t+1}$. For the critic, two V-values, $V^{\boldsymbol{\theta}}(\boldsymbol{s}_t;\boldsymbol{\omega}_t)$ and $V^{\boldsymbol{\theta}}(\boldsymbol{s}_{s+1};\boldsymbol{\omega}_t)$, are estimated by DNN with the inputs $\boldsymbol{s}_t$ and $\boldsymbol{s}_{t+1}$, respectively. The TD error $\delta_{_V}(\boldsymbol{\omega}_t)$ can be calculated by Eq. (30). A tuple $\{\boldsymbol{s}_t, \boldsymbol{s}_{t+1}, \delta_{_V}(\boldsymbol{\omega}_t), r_t\}$ is stored in a memory at each step t. By applying a memory replay mechanism, the data in the memory can be used for training the DNNs. In each training step, the actor and critic are updated by the gradient descent over a batch of training data. The whole training process consists of multiple episodes, each episode including T_{max} steps. Based on the above framework, the AC-DSOS algorithm is summarized in Alg. 2.

In AC-DSOS, two DNNs are employed to estimate the stochastic policy and V-value with the input $\boldsymbol{s}_t$. The complexity of AC-DSOS is dominated by the DNNs' forward-propagation and back-propagation process in lines 6, 13, and 16 in Algorithm 2. The size of the input is $N(K_{max}^2+1)+1$, where $K_{max}=\max\{K_1,\ldots,K_N\}$. We assume both DNNs have X hidden layers and the x-th layer has l_x nodes. In the forward propagation (line 6 and line 13 in Alg. 2), the complexity of the actor DNN and critic DNN are identical, i.e., $O((N(K_{max}^2+1)+1)l_1+\sum_{x=1}^{X-1} l_x l_{x+1}+l_X)$, referring to [45]. In the back propagation, (line 16 in Alg. 2), the stochastic gradient is obtained with the complexity $O(M((N(K_{max}^2+1)+1)l_1+\sum_{x=1}^{X-1} l_x l_{x+1}+l_X))$ to update the neural parameters, where M is the batch size. Overall, the complexity of AC-DSOS is $O(2T_{max}(M+1)((N(K_{max}^2+1)+1)l_1+\sum_{x=1}^{X-1} l_x l_{x+1}+l_X))$.

The novelties of the proposed AC-DSOS compared to the conventional AC-DRL are summarized as follows.

1) Action Mapping to Tackle the Issue of Huge Discrete Action Space: The conventional actor-critic is used for continuous action space. We denote $\hat{\boldsymbol{a}}_t=[\hat{a}_{1,t},\ldots,\hat{a}_{I,t}]$ as the original action selected by the stochastic policy, where the element $\hat{a}_{i,t}$ is fractional. However, as the decision variables are integers in $\mathcal{P}_1$, the action space is discrete. To deal with this issue, we adopt an action mapping method in AC-DSOS (line 9 in Alg. 2). Firstly, we confine $\hat{a}_{i,t}$ to a fixed range $[-\kappa,\kappa]$ to avoid its value being too large/small since the domain of Gaussian distribution is $[-\infty,\infty]$. Then, a uniform quantization method is used to map $\hat{a}_{i,t}$ to the discrete action space $\{1,\ldots,G_n\}$ by:

$$a_{i,t}=\lceil\frac{\kappa+\hat{a}_{i,t}}{2\kappa/G_n}\rceil, \tag{40}$$

where $2\kappa/G_n$ is the quantization interval. With the mapping operation, we can support a larger G_n by reducing the interval.

2) Action Space Restriction to Improve Solution Quality: Although AC-DSOS can tackle the issue of discrete action space by the above mapping operation, exploring in a huge space remains difficult. To improve the exploration efficiency and the quality of the solution, we design a method to restrict the action space in the learning process (line 5 in Alg. 2). Compared to the conventional DRL method, the difference mainly lies at the action selection. At the beginning of each frame, we first observe which users' demands have been satisfied. Then, we remove the corresponding candidate groups, i.e., the groups containing the successfully served users. In conventional DRL, the action space keeps fixed as $G_n=2^{K_n}-1$. This may result in two issues: Firstly, when the action space grows exponentially large, DRL/RL needs more time to search for the optimal action, thus decreases the exploration efficiency; Secondly, the probability of selecting undesirable low-reward actions will be increased. This is because the original actions $\hat{a}_{i,t}$ are selected by a stochastic policy such that a small difference in $\hat{a}_{i,t}$ could lead to different actions after mapping. For example, $\hat{a}_{i,t}=0.999$ and $\hat{a}_{i,t}=1.001$ map to $a_{i,t}=1$ and $a_{i,t}=2$, respectively. If $a_{i,t}=2$ is a low-reward action, a small error in $\hat{a}_{i,t}$ could cause a large loss in reward value. As illustrated in Fig. 4, the size of the action space in AC-DSOS, denoted by G_t ($G_t\le G_n$), is not fixed but gradually reduces over $1,\ldots,T_{max}$. Before taking an action, we remove the redundant actions with lower rewards from the action space, such that the action space can keep concise and with controllable size.

Lemma 2: At each learning step, $V^{\pi}(\boldsymbol{s}_t)\le V^{\pi'}(\boldsymbol{s}_t)$, where $V^{\pi}(\boldsymbol{s}_t)$ and $V^{\pi'}(\boldsymbol{s}_t)$ are the V-values under the policy with the fixed action space and the reduced action space, respectively.

Proof: We denote $\mathcal{A}$ and $\mathcal{A}'$ as the fixed action space and the reduced action space, respectively. Based on bellman equation [15], $V^{\pi}(\boldsymbol{s}_t)$ can be expressed as:

$$\begin{aligned}V^{\pi}(\boldsymbol{s}_t)&=\sum_{\boldsymbol{a}_t\in\mathcal{A}}\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)\left(r(\boldsymbol{s}_t,\boldsymbol{a}_t)+\gamma V^{\pi}(\boldsymbol{s}_{t+1})\right)\\&=\sum_{\boldsymbol{a}_t\in\mathcal{A}}\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)r(\boldsymbol{s}_t,\boldsymbol{a}_t)\\&\quad+\gamma\sum_{\boldsymbol{a}_{t+1}\in\mathcal{A}}\pi(\boldsymbol{a}_{t+1}|\boldsymbol{s}_{t+1})r(\boldsymbol{s}_{t+1},\boldsymbol{a}_{t+1})\\&\quad+\gamma^2\sum_{\boldsymbol{a}_{t+2}\in\mathcal{A}}\pi(\boldsymbol{a}_{t+2}|\boldsymbol{s}_{t+2})r(\boldsymbol{s}_{t+2},\boldsymbol{a}_{t+2})+\cdots\end{aligned} \tag{41}$$

$\mathcal{A}'$ excludes the redundant actions from $\mathcal{A}$, that is, the actions that bring the lowest rewards, thus,

$$r(\boldsymbol{s}_t,\boldsymbol{a}_t|\boldsymbol{a}_t\in\mathcal{A}')>r(\boldsymbol{s}_t,\boldsymbol{a}_t|\boldsymbol{a}_t\in\mathcal{A}\setminus\mathcal{A}'). \tag{42}$$

For the probability distribution of the two policies, the following equations hold.

$$\sum_{\boldsymbol{a}_t\in\mathcal{A}}\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)=\sum_{\boldsymbol{a}_t\in\mathcal{A}}\pi'(\boldsymbol{a}_t|\boldsymbol{s}_t)=1. \tag{43}$$

$$\pi'(\boldsymbol{a}_t|\boldsymbol{s}_t)=0,\ \boldsymbol{a}_t\in\mathcal{A}\setminus\mathcal{A}',' \tag{44}$$

$$\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)\ge 0,\ \boldsymbol{a}_t\in\mathcal{A}\setminus\mathcal{A}'. \tag{45}$$

Based on Eq. (42)-(45), we can derive:

$$\begin{aligned}&\sum_{\boldsymbol{a}_t\in\mathcal{A}}\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)r(\boldsymbol{s}_t,\boldsymbol{a}_t)\\&=\sum_{\boldsymbol{a}_t\in\mathcal{A}'}\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)r(\boldsymbol{s}_t,\boldsymbol{a}_t)+\sum_{\boldsymbol{a}_t\in\mathcal{A}\setminus\mathcal{A}'}\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)r(\boldsymbol{s}_t,\boldsymbol{a}_t)\\&<\sum_{\boldsymbol{a}_t\in\mathcal{A}'}\pi'(\boldsymbol{a}_t|\boldsymbol{s}_t)r(\boldsymbol{s}_t,\boldsymbol{a}_t)+\sum_{\boldsymbol{a}_t\in\mathcal{A}\setminus\mathcal{A}'}\pi'(\boldsymbol{a}_t|\boldsymbol{s}_t)r(\boldsymbol{s}_t,\boldsymbol{a}_t)\\&=\sum_{\boldsymbol{a}_t\in\mathcal{A}}\pi'(\boldsymbol{a}_t|\boldsymbol{s}_t)r(\boldsymbol{s}_t,\boldsymbol{a}_t).\end{aligned} \tag{46}$$

Substituting Eq. (46) into Eq. (41), the inequality $V^{\pi}(\boldsymbol{s}_t) < V^{\pi'}(\boldsymbol{s}_t)$ can be obtained. Thus the conclusion. ■

By recalling Eq. (29), the definition of V-value is the average accumulative discounted reward, $V^{\pi}(\boldsymbol{s}_t) = \mathbb{E}_{\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)}[R_t|\boldsymbol{s}_t]$. Based on Lemma 2, as $V^{\pi}(\boldsymbol{s}_t) \leq V^{\pi'}(\boldsymbol{s}_t)$, the policy with the reduced action space provides a higher average R_t than that of the fixed action space. In addition, the reduced action space helps the agent to avoid searching for low-reward actions, thereby reducing the computational time in exploration, which can be verified by simulation.

3) Re-Designed Reward Function to Deal With Feasibility Issues: Without a carefully designed mechanism, the actions made in conventional AC-DRL may easily violate constraints, thus fail to guarantee the solution feasibility. In $\mathcal{P}_1$, the major difficulty comes from constraints (10b), whereas (10c)-(10h) can be satisfied by properly defined actions. Under the commonly-used reward designs, e.g., Eq. (38) or Eq. (39), constraint (10b) may not be satisfied since the criterion of the decision making is to minimize the objective energy without considering constraints. To solve the problem, we re-design the reward function by incorporating constraint (10b), which is given by:

$$r_t = \frac{\sum_{n=1}^{N} d_{n,t}^{\pi}}{(e_t^{\pi})^{\epsilon}}. \tag{47}$$

The rationale is that the proposed reward function is the ratio between the delivered data and the consumed energy on frame t, where ϵ is a control parameter. When ϵ is small, the reward enforces the UAV to deliver more data to meet users' demands. However, transmitting more data results in more energy consumption. To control energy growth, we can increase ϵ such that the agent will reduce the energy consumption to avoid the reward losses. Thus, by tuning an appropriate ϵ, the decisions made by AC-DSOS can achieve good energy-saving performance while satisfying users' demands.

For practical applications, AC-DSOS is designed to overcome the limitations brought by constrained combinatorial problems, e.g., guarantee feasibility and control exponentially-increased action space. Compared to conventional optimization and DRL approaches, AC-DSOS is expected to achieve a good trade-off between solution quality and complexity. From a theoretical perspective, AC-DSOS is of polynomial-time complexity, which provides a theoretical basis for its further real-time applications. The developed Lemma 2 proves that the reduced action space can lead to higher accumulative reward, which justifies the developed approaches and grantees the performance theoretically.

VI. Numerical Results

In this section, we present numerical results to evaluate the performance of the proposed AC-DSOS algorithm and compare it with other schemes:

- Previous AC-DRL scheme: Deep deterministic policy gradient (DDPG) [46];
- Previous AC-DRL scheme: Proximal policy optimization (PPO) [47];
- High-complexity heuristic scheme: the proposed GSS-HEU in Alg. 1;
- High-complexity heuristic scheme: the alternative optimization algorithm (ALT-HEU) [8];
- Low-complexity heuristic scheme: semi-orthogonal user scheduling-based heuristic algorithm (SUS-HEU) [48];
- Optimal scheme: optimal algorithm (OPT).

DDPG and PPO provide performance benchmarks from the AC-DRL perspective. Both of them are based on stochastic policy gradient with fixed action space. The structure of the DNNs, parameter settings, and reward function, i.e., Eq. (47), for AC-DSOS, DDPG and PPO keep the same in order to enable a fair comparison. The proposed GSS-HEU, ALT-HEU in [8], SUS-HEU in [48], and OPT are the benchmark schemes from an optimization perspective. We implement ATL-HEU by applying its core idea of the block coordinate descent method to alternatively optimize two blocks, i.e., hovering time and user scheduling. SUS-HEU adopts a simple user-grouping strategy with lower complexity than GSS-HEU and ALT-HEU.

In the simulation, we first evaluate the performance of energy consumption and computational time. After that, we justify the developed new reward function in guaranteeing solution feasibility by comparing several well-known reward functions. Furthermore, we evaluate the convergence performance of AC-DSOS with different learning rates.

TABLE II
Parameters in AC-DSOS

Parameters	Actor	Critic
Number of hidden layers	3	3
Number of nodes/layer	300	300
Activation function (hidden layers)	ReLU	ReLU
Activation function (output layer)	Sigmoid	None
Learning rate	0.003	0.002
Loss function	Eq. (24)	Eq. (31)
Optimizer	Adam	Adam
Batch size	64	64
Discount factor γ	0.9	
Memory size	10,000 tuples	
Number of learning episodes	400	
Value range $[-\kappa, \kappa]$ of $\hat{a}_{i,t}$	[-2, 2]	
Software platform	Python 3.6 with TensorFlow 0.12.1	

A. Parameter Settings

The UAV is equipped with $L = 10$ antennas serving $N = 3$ clusters. The ground users are randomly scattered in the service area. Each cluster contains up to $K = 9$ users. The users' demands $q_{k,n}$ are randomly selected from $\{1, 2, 3, 4, 5\}$ (Mbit). We assume the bandwidth $B = 10$ MHz, noise power $\sigma^2 = 0.1$ mW, hovering power $P_H = 10$ W, and transmit power $p_{k,g,n} = 3$ W, referring to [4]. Based on FSMC, we quantize $\beta_{g,n,t}^{(kk)}$ and $\beta_{g,n,t}^{(kj)}$ into 9 levels, $\{0, 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4\}$. The setting of the transfer probability matrix is similar in [49]. Two fully-connected DNNs are employed as the actor and the critic. The adopted parameters for implementing AC-DSOS are summarized in Table II.

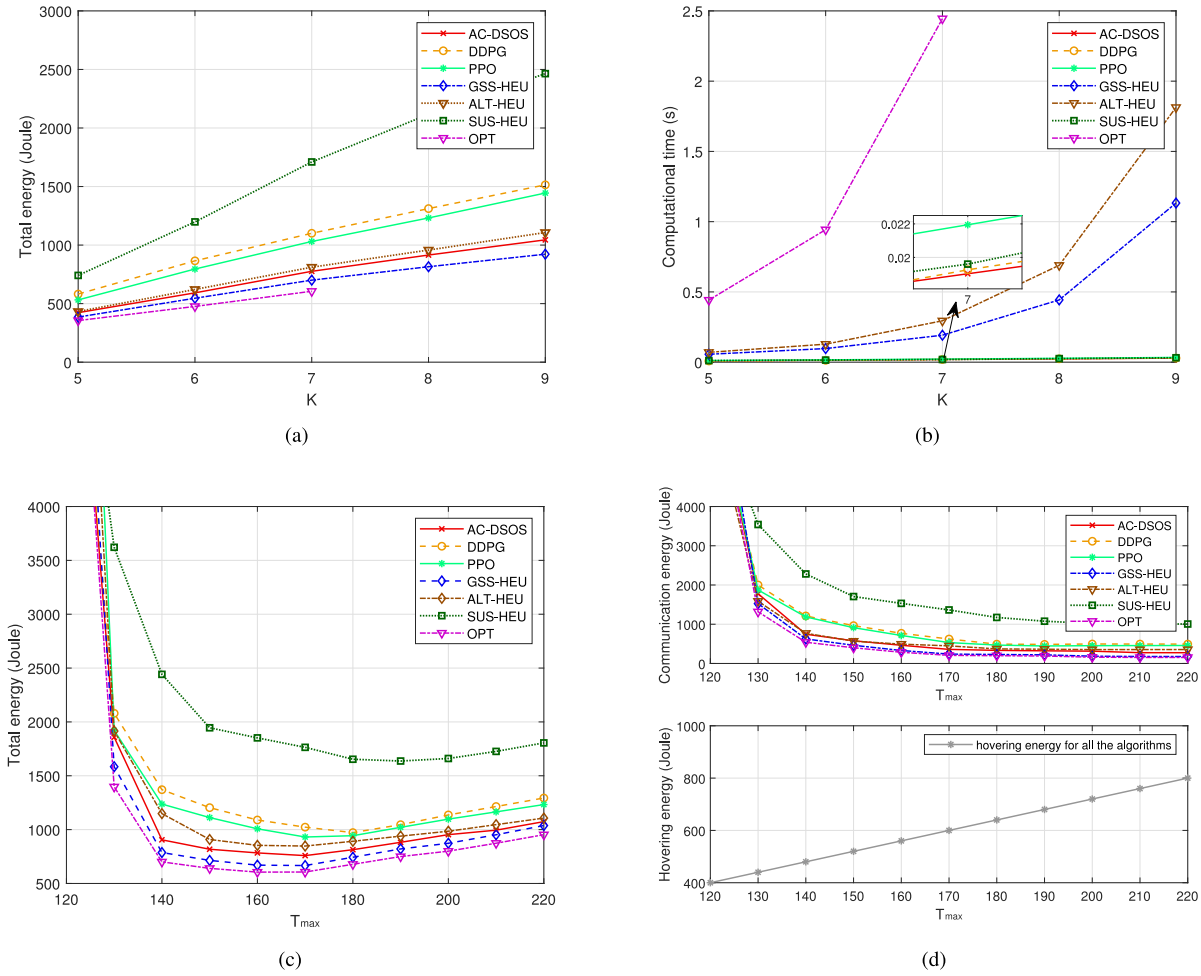

Fig. 5. (a) Total energy vs. K ($T_{max} = 160$). (b) Average computational time vs. K ($T_{max} = 160$). (c) Total energy vs. T_{max} ($K = 7$). (d) Communication and hovering energy vs. T_{max} ($K = 7$).

B. Results and Analysis

1) Trade-Off Performance Between Energy and Computational Time: Firstly, by comparing with six benchmarking algorithms in Fig. 5, the proposed AC-DSOS achieves a good trade-off between energy minimization and computational time. Note that for $K > 7$, the optimal energy results are absent due to the high complexity and the corresponding long computational time. From Fig. 5, AC-DSOS saves around 29.94% and 24.84% energy compared to DDPG and PPO on average. Overall, AC-DSOS provides a sub-optimal solution, with 19.17% gap to the optimum. GSS-HEU achieves near optimality, and consumes less 9.8% energy than AC-DSOS in average but with paying much higher complexity and time, e.g., see Fig. 5(b). ALT-HEU takes more computational time but the average energy-saving performance is 4.28% inferior to AC-DSOS as the algorithm is sensitive to the initial point. SUS-HEU consumes the highest energy since it schedules users based on channel conditions without considering energy consumption. It is also shown that the total objective energy follows a roughly linear increase in all the algorithms. The gaps between the optimal algorithm and other algorithms become larger as K increases. When K grows from 5 to 7, the gap to the optimum increases from 47.7% to 65.1% for SUS-HEU, and from 31% to 44.5% for DDPG. In AC-DSOS, since the delivery-completed users are deleted during the learning process, the size of the action space will continuously decrease. This improves the searching efficiency and quality, and reduces the growth rate of the gap as K increases, from 11.1% ($K = 5$) to 16.7% ($K = 7$).

Fig. 5(b) compares the computational time with respect to K. The computational time is accounted as the elapsed time of producing an optimized solution per frame. In GSS-HEU, ALT-HEU and OPT, the computational time grows exponentially with K, whereas the proposed AC-DSOS along with DDPG, PPO and SUS-HEU maintain at the millisecond magnitude and insensitive to K. In average, AC-DSOS saves 99.23%, 92.86%, and 89.98% computational time compared to OPT, GSS-HEU, and ALT-HEU, respectively. This is due to the fact that DRL can provide online decisions based on the current environment state instead of solving the optimization problem directly. PPO consumes 14.55% more computational time than AC-DSOS since calculating the gradient for a complex loss function consumes extra time. The computational time of AC-DSOS is slightly lower than DDPG and SUS-HEU. However, by recalling Fig. 5(a), AC-DSOS saves 24.84%, 29.94%, and 52.51% energy

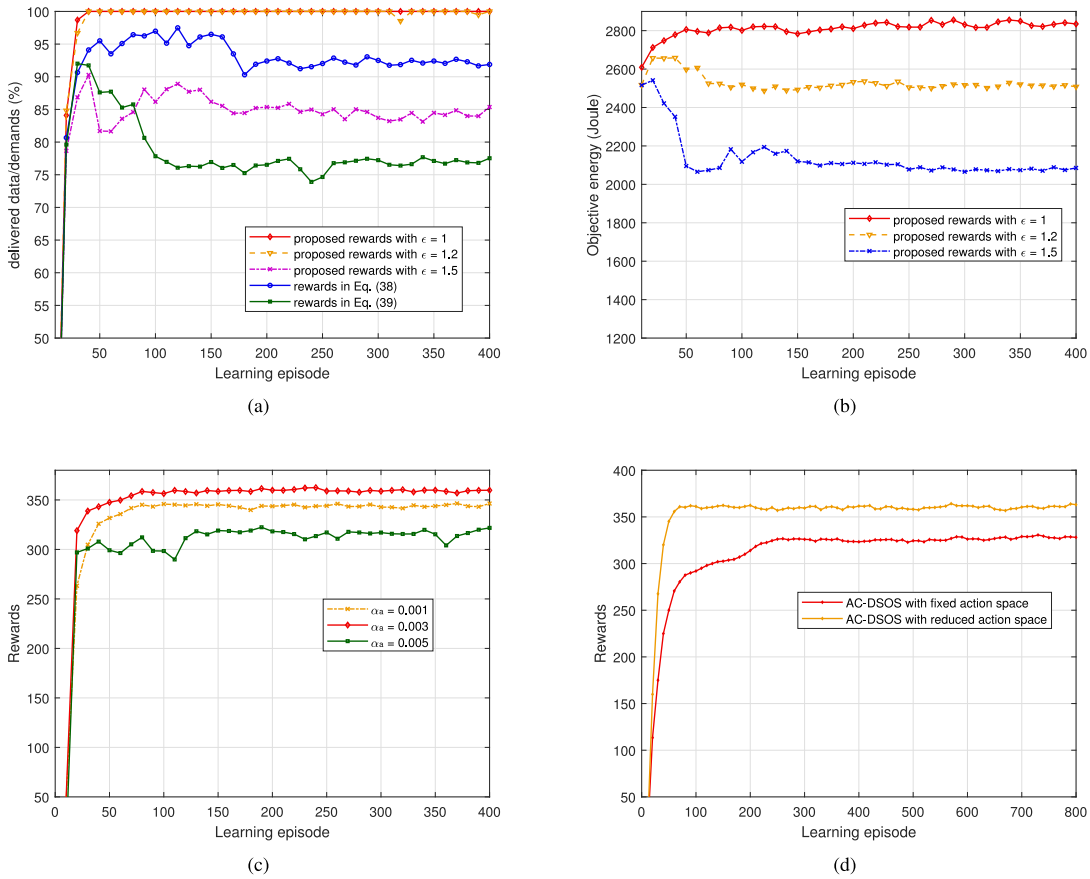

Fig. 6. (a) Feasibility vs. learning episode. (b) Energy vs. learning episode. (c) Rewards vs. learning episode. (d) Rewards in AC-DSOS with fixed and reduced action space.

compared with PPO, DDPG, and SUS-HEU, respectively. In addition, we can observe that the computational time of GSS-HEU and OPT exceeds 1 s when $K = 9$ and $K = 7$, respectively, which is impractical in the scenarios with strict delay requirements. For AC-DSOS, the result remains at the millisecond-level, even if the number of users increases from 5 to 9.

Fig. 5(c) demonstrates the total energy consumption with respect to T_{max}, and Fig. 5(d) illustrates the communication energy and hovering energy separately. From Fig. 5(c), AC-DSOS outperforms DDPG and PPO by saving 21.37% and 18.45% total energy on average. The average gap between GSS-HEU and the optimal solution is 8.91% smaller than that of AC-DSOS, but, from Fig. 5(b), GSS-HEU consumes nearly 126 times higher calculation time than AC-DSOS at $T_{max} = 160$. The energy-saving performance of SUS-HEU is worse than other algorithms and its gap to the optimum reaches 59.44%. Fig. 5(c) also shows that, as T_{max} increases, the objective energy rapidly decreases first then grows steadily. This can be explained via Fig. 5(d). The objective energy consists of the communication energy and hovering energy. From Fig. 5(d), the communication energy drops rapidly when $T_{max} < 140$, and becomes stable after $T_{max} > 180$. Whereas, the hovering energy increases linearly with T_{max} for all the algorithms.

2) Feasibility and Convergence Performance: Fig. 6(a) verifies the capability of the proposed reward function in dealing with feasibility issues, where a feasible solution is obtained only if the ratio of delivered demand over total demand in y-axis achieves 100%. From Fig. 6(a), the reward functions used in Eq. (38) and Eq. (39) fail to guarantee the feasibility of the solution. For the re-designed reward, we evaluate the performance by setting ϵ to 1, 1.2, and 1.5. A small ϵ means that transmitting more data can bring more rewards gain than saving energy. When ϵ drops below 1.2, the feasibility issue can be solved. Fig. 6(b) shows the objective energy with different ϵ. It can be found that a smaller ϵ leads to more energy consumption. Thus, an appropriate parameter ϵ lies at 1.2, enabling the after-learned solution to guarantee the demands while consuming less energy.

Fig. 6(c) demonstrates the convergence of AC-DSOS with different actor's learning rate α_a. The x-axis is the learning episode and the y-axis is the received accumulative reward R_m in the m-th episode. We define that the AC-DSOS algorithm converges if there exist $\bar{R}$ and a sufficiently large integer m_{con}, such that $|R_m - \bar{R}| < \varepsilon$ for all $m > m_{con}$, where ε is a positive tolerance. From 6, we can observe that when the learning rate $\alpha_a = 0.001$ and $\alpha_a = 0.003$, the curves converge around 80 episodes. As α_a increases to $\alpha_a = 0.005$, the curve fluctuates due to the large update step. Taking the actor as an example, the learning rate for the critic α_c has the same tendency. In conclusion, the learning rates of the actor and critic are sensitive to the convergence, and need to be properly selected, e.g., 0.003 for the actor.

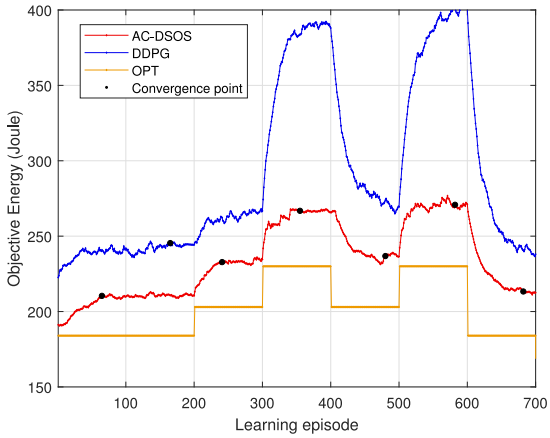

Fig. 7. Energy comparison in a scenario with dynamic user arrival and departure.

Fig. 6(d) compares the policy with reduced action space and fixed action space in convergence speed and reward evolution. AC-DSOS with reduced action space converges around 60 episodes, against 250 episodes in AC-DSOS with fixed action space, and achieves 8.33% higher reward value than the other in average. In addition, we can observe that, with the fixed-large action space, the agent is likely to get stuck in local points, which can result in more time in exploration to escape from the points, referring to the red curve's step-like effect at the 100-200 episodes. Overall, the policy with the reduced action space is effective in improving learning efficiency and reward quality.

3) Performance Comparison in Dynamic Environments: In Fig. 7, we evaluate the capability of AC-DSOS in adapting network dynamics. Unlike previous static scenarios, we consider a dynamic scenario, where users request diverse amounts of data, and their arrival/departure in each cluster are varying over time by following the Poisson distribution. Starting from the 200-th episode, we assume that the entry/leave event happens every 100 episodes. For example, at the 200-th episode, some new users join the clusters and request date services. As a consequence, more energy is consumed (see the optimal energy consumption in OPT). In DDPG and AC-DSOS, the agents need time to learn and train to adapt to the new users due to the lack of their prior/historical knowledge. Both algorithms, therefore, undergo an adjusting period to converge to adapt to the environment change.

From the results, AC-DSOS demonstrates two advantages compared to DDPG. Firstly, AC-DSOS converges faster than DDPG. AC-DSOS is able to converge within 60 episodes such that it is more timely and adaptive to handle the periodically-changed network. Such improved computational efficiency and convergence are benefited by the developed action-space-reduction and policy-quantification approaches. In contrast, DDPG leads to a worse case. That is, the algorithm has not been converged to react to the first environment change but the second has arrived. As a result, DDPG is not able to converge. Secondly, compared to the performance in static cases, e.g., Fig. 5(a), the average gap in energy consumption between AC-DSOS and OPT remains stable, within 20%, whereas the performance of DDPG fluctuates dramatically.

VII. Conclusion

In this paper, we have investigated an energy minimization problem for UAV-aided communication systems from the perspective of AC-DRL. The formulated problem is combinatorial and non-convex. We provided an optimal method and proposed a GSS-based heuristic algorithm to solve the problem and serve as benchmarks. To make the solutions adaptive to online operations, we propose an AC-DSOS algorithm. Different from previous AC-DRL methods, the proposed AC-DSOS is able to deal with the huge discrete action space and guarantee feasibility. Numerical results have shown that AC-DSOS provides a good trade-off between energy efficiency and computational efficiency. Furthermore, the re-designed reward function is effective to deal with the feasibility issue.

As a future extension, the proposed AC-DSOS can be further extended to adapt multi-UAV/multi-agent scenarios in two possible ways, i.e., distributed AC-DSOS and hybrid centralized-distributed AC-DSOS. For the former, a decentralized AC-DSOS is applied to each UAV/agent individually in a distributed multi-UAV system. Each agent learns its own value function network and strategy network without considering the mutual influence of other agents. For the latter, AC-DSOS is extended to enable centralized training and decentralized execution, where AC-DSOS is to use the global Q-value for each agent to update the local policy.

References

[1] 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. Surv. Tut.*, vol. 21, no. 3, pp. 2334–2360, Mar. 2019.

[2] S. Ahmed, M. Z. Chowdhury, and Y. M. Jang, "Energy-efficient UAV-to-User scheduling to maximize throughput in wireless networks," *IEEE Access*, vol. 8, pp. 21215–21225, Jan. 2020.

[3] J. Zhang, Y. Zeng, and R. Zhang, "Spectrum and energy efficiency maximization in UAV-Enabled mobile relaying," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.

[4] 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, Mar. 2019.

[5] D. H. Tran, T. X. Vu, S. Chatzinotas, S. ShahbazPanahi, and B. Ottersten, "Coarse trajectory design for energy minimization in UAV-enabled," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 9483–9496, Jun. 2020.

[6] S. Zhu, K. Yang, J. Ouyang, and Y. Du, "Cooperative beamforming for UAV-assisted cognitive relay networks with partial channel state information," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2018, pp. 158–162.

[7] Q. Song and F. Zheng, "Energy efficient multi-antenna UAV-enabled mobile relay," *China Commun.*, vol. 15, no. 5, pp. 41–50, May 2018.

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

[9] F. Zhou, N. C. Beaulieu, J. Cheng, Z. Chu, and Y. Wang, "Robust max-min fairness resource allocation in sensing-based wideband cognitive radio with SWIPT: Imperfect channel sensing," *IEEE Syst. J.*, vol. 12, no. 3, pp. 2361–2372, Sep. 2018.

[10] Z. Chang, L. Lei, Z. Zhou, S. Mao, and T. Ristaniemi, "Learn to cache: Machine learning for network edge caching in the big data era," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 28–35, Jun. 2018.

[11] M. Shin, J. Kim, and M. Levorato, "Auction-based charging scheduling with deep learning framework for multi-drone networks," *IEEE Trans. Veh. .Technol.*, vol. 68, no. 5, pp. 4235–4248, May 2019.

[12] L. Lei, L. You, G. Dai, T. X. Vu, D. Yuan, and S. Chatzinotas, "A deep learning approach for optimizing content delivering in cache-enabled HetNet," in *Proc. IEEE Int. Symp. Wirel. Commun. Syst.*, 2017, pp. 449–453.

[13] Y. Hsu and R. Gau, "Reinforcement learning-based collision avoidance and optimal trajectory planning in UAV communication networks," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2020.3003639.

[14] W. Liu, P. Si, E. Sun, M. Li, C. Fang, and Y. Zhang, "Green mobility management in UAV-Assisted IoT based on dueling DQN," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–6.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An introduction*. Cambridge, MA, USA: MIT Press, 2018.

[16] J. He *et al.*, "Deep reinforcement learning with a combinatorial action space for predicting popular reddit threads," in *Proc. Conf. Empirical Meth. Nat. Lang. Processing*, 2016, pp. 1838–1848.

[17] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1008–1014.

[18] 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, Aug. 2018.

[19] C. H. Liu, Z. Dai, Y. Zhao, J. Crowcroft, D. O. Wu, and K. Leung, "Distributed and energy-efficient mobile crowdsensing with charging stations by deep reinforcement learning," *IEEE Trans. Mobile Comput.*, vol. 20, no. 1, pp. 130–146, 1 Jan. 2021.

[20] Y. Cao, L. Zhang, and Y. Liang, "Deep reinforcement learning for channel and power allocation in UAV-enabled IoT systems," in *Proc. IEEE Glob. Commun. Conf.*, 2019, pp. 1–6.

[21] Y. Yuan, L. Lei, T. X. Vu, S. Chatzinotas, and B. Ottersten, "Actor-critic deep reinforcement learning for energy minimization in UAV-aided networks," in *Proc. IEEE Eur. Conf. Netw. Commun.*, 2020, pp. 348–352.

[22] R. Ding, F. Gao, and X. Shen, "3D UAV trajectory design and frequency band allocation for energy-efficient and fair communication: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 7796–7809, Dec. 2020.

[23] W. Shi, J. Li, H. Wu, C. Zhou, N. Cheng, and X. Shen, "Drone-cell trajectory planning and resource allocation for highly mobile networks: A hierarchical DRL approach," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2020.3020067.

[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] S. Yin, L. Li, and F. R. Yu, "Resource allocation and basestation placement in downlink cellular networks assisted by multiple wireless powered UAVs," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2171–2184, Feb. 2020.

[26] F. Iannello and O. Simeone, "On the optimal scheduling of independent, symmetric and time-sensitive tasks," *IEEE Trans. Autom. Control*, vol. 58, no. 9, pp. 2421–2425, Sep. 2013.

[27] Y. Yuan, T. X. Vu, L. Lei, S. Chatzinotas, and B. Ottersten, "Joint user grouping and power allocation for MISO systems: Learning to schedule," in *Proc. Eur. Signal Process. Conf.*, Nov. 2019, pp. 1–5.

[28] E. Yanmaz, R. Kuschnig, and C. Bettstetter, "Channel measurements over 802.11a-Based UAV-to-ground links," in *Proc. IEEE Glob. Commun. Conf. Workshops*, 2011, pp. 1280–1284.

[29] C. You and R. Zhang, "3D trajectory optimization in rician fading for UAV-Enabled data harvesting," *IEEE Trans. Wireless Commun.*, vol. 18, no. 6, pp. 3192–3207, Apr. 2019.

[30] C. Yan, L. Fu, J. Zhang, and J. Wang, "A comprehensive survey on UAV communication channel modeling," *IEEE Access*, vol. 4, pp. 107769–107792, Aug. 2019.

[31] Q. Pan, S. Liu, M. Xu, and C. Jia, "Finite-state Markov model for the aeronautical channel," in *Proc. IEEE Int. Conf. Wirel. Commun., Netw. Mobile Comput.*, Sep. 2009, pp. 1–4.

[32] J. Yang, P. Liu, and H. Mao, "Model and simulation of narrowband ground-to-air fading channel based on Markov process," in *Proc. Int. Conf. Netw. Comput. Inf. Secur.*, Jul. 2011, pp. 142–146.

[33] Y. Zhou, J. Li, L. Lamont, and C. A. Rabbath, "A Markov-based packet dropout model for UAV wireless communications," *J. Commun.*, vol. 7, no. 6, pp. 418–426, Jun. 2012.

[34] H. Wang and N. Moayeri, "Finite-state Markov channel—A useful model for radio communication channels," *IEEE Trans. Veh. Technol.*, vol. 44, no. 1, pp. 163–171, Feb. 1995.

[35] A. Filippone, *Flight Performance of Fixed and Rotary Wing Aircraft*. Amsterdam, Netherlands: Elsevier, 2006.

[36] G. P. McCormick, "Computability of global solutions to factorable nonconvex programs: Part I convex underestimating problems," *Math. Program.*, vol. 10, no. 1, pp. 147–175, Dec. 1976.

[37] L. Lei *et al.* "Learning-assisted optimization for energy-efficient scheduling in deadline-aware NOMA systems," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 3, pp. 615–627, Sep. 2019.

[38] M. Hifi, M. Michrafy, and A. Sbihi, "Heuristic algorithms for the multiple-choice multidimensional knapsack problem," *J. Oper. Res. Soc.*, vol. 55, no. 12, pp. 1323–1332, Dec. 2004.

[39] J. Guillot, D. R. Leal, C. R. Algarn, and I. Oliveros, "Search for global maximal in multimodal functions by applying numerical optimization algorithms: A comparison between golden section and simulated annealing,"*Computation*, vol. 7, no. 3, pp. 1–13, 2019.

[40] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. Int. Conf. Learning Representations*, 2016, pp. 1–14.

[41] L. Wang *et al.*, "RL-based user association and resource allocation for multi-UAV enabled MEC," in *IEEE Int. Wireless Commun. and Mobile Comput. Conf.*, 2019, pp. 741–746.

[42] Y. Lu, H. Lu, L. Cao, F. Wu, and D. Zhu, "Learning deterministic policy with target for power control in wireless networks," in *Proc. IEEE Glob. Commun. Conf.*, 2018, pp. 1–7.

[43] G. Dulac-Arnold *et al.*, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Mach. Learning*, vol. 110, pp. 1–7, Apr. 2021.

[44] Y. Wei, F. R. Yu, M. Song, and Z. Han, "User scheduling and resource allocation in HetNets with hybrid energy supply: An actor-critic reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 680–692, Nov. 2017.

[45] N. Yang, H. Zhang, K. Long, H. Hsieh, and J. Liu, "Deep neural network for resource management in NOMA networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 876–886, Jan. 2020.

[46] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learning*, Jan. 2014, pp. 387–395.

[47] B. Liu, Q. Cai, Z. Yang, and Z. Wang," Neural trust region/proximal policy optimization attains globally optimal policy," in *Adv. Neural Inf. Process. Syst.*, Jun. 2019.

[48] T. Yoo and A. Goldsmith, "On the optimality of multiantenna broadcast scheduling using zero-forcing beamforming," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 3, pp. 528–541, Mar. 2006.

[49] Y. He *et al.*, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11 pp. 10433–10445, Sep. 2017.

Yaxiong Yuan (Student Member, IEEE) received the M.S. degree from the Laboratory of Wireless Communication Systems and Networks, Beijing University of Posts and Telecommunications, Beijing, China. He is currently working toward the Ph.D. degree with Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg City, Luxembourg. His research interests include optimization theory, machine learning, wireless resource management, and wireless communication networks.

Lei Lei (Member, IEEE) received the B.Eng. and M.Eng. degrees from Northwestern Polytechnic University, Xi'an, China, in 2008 and 2011, respectively, and the Ph.D. degree from the Department of Science and Technology, Linköping University, Linköping, Sweden, in 2016. He is currently a Research Scientist with the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg City, Luxembourg. In November 2016, he joined SnT as a Research Associate. From June 2013 to December 2013, he was a Research Assistant with Institute for Infocomm Research, A*STAR, Singapore. His current research interests include resource allocation and optimization in 5G-satellite networks, energy-efficient communications, and deep learning in wireless communications. He was the recipient of the IEEE Sweden Vehicular Technology-Communications-Information Theory (VT-COM-IT) joint chapter Best Student Journal Paper Award in 2014. He was the co-recipient of the IEEE SigTelCom 2019 Best Paper Award.

Thang X. Vu (Member, IEEE) received the B.S. and M.Sc. degrees in electronics and telecommunications engineering, from the VNU University of Engineering and Technology, Hanoi, Vietnam, in 2007 and 2009, respectively, and the Ph.D. degree in electrical engineering from the University Paris-Sud, Paris, France, in 2014. From September 2010 to May 2014, he was with the Laboratory of Signals and Systems, a joint laboratory of CNRS, Centrale Supelec and University Paris-Sud XI, Paris, France. From July 2014 to January 2016, he was a Postdoctoral Researcher with the Information Systems Technology and Design Pillar, Singapore University of Technology and Design, Singapore. He is currently a Research Scientist with the Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg City, Luxembourg. His research interests include wireless communications, with particular interests of wireless edge caching, cloud radio access networks, machine learning for communications and cross-layer resources optimization. He was the recipient of the SigTelCom 2019 Best Paper Award. In 2010, he was also the recipient of the Allocation de Recherche fellowship to study Ph.D. in France.

Symeon Chatzinotas (Senior Member, IEEE) received the M.Eng. degree in telecommunications from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 2003, and the M.Sc. and Ph.D. degrees in electronic engineering from the University of Surrey, Surrey, U.K., in 2006 and 2009, respectively. He is currently a Full Professor or Chief Scientist I and the Co-Head of the SIGCOM Research Group, SnT, University of Luxembourg, Luxembourg City, Luxembourg. He was a Visiting Professor with the University of Parma, Parma, Italy and he was involved in numerous Research and Development projects for the National Center for Scientific Research Demokritos, the Center of Research and Technology Hellas and the Center of Communication Systems Research, University of Surrey, Guildford, U.K. He has coauthored more than 400 technical papers in refereed international journals, conferences and scientific books. He was the co-recipient of the 2014 IEEE Distinguished Contributions to Satellite Communications Award, the CROWNCOM 2015 Best Paper Award, and the 2018 EURASIP JWCN Best Paper Award. He is currently on the Editorial Board of the IEEE Open Journal of Vehicular Technology and the International Journal of Satellite Communications and Networking.

Sumei Sun (Fellow, IEEE) is currently the Principal Scientist and Head of the Communications and Networks Department, Institute for Infocomm Research, Singapore. She is also holding a joint appointment with the Singapore Institute of Technology, Singapore, and an adjunct appointment with the National University of Singapore, Singapore, as a Full Professor. Her current research interests include next-generation wireless communications, cognitive communications and networks, and industrial internet of things. She is the Editor-in-Chief of IEEE OPEN JOURNAL OF VEHICULAR TECHNOLOGY, Member of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS Steering Committee, and a Distinguished Speaker of the IEEE Vehicular Technology Society from 2018 to 2021. She is also the Director of IEEE Communications Society Asia Pacific Board and a Member-at-Large with the IEEE Communications Society.

Björn Ottersten (Fellow, IEEE) was born in Stockholm, Sweden, in 1961. He received the M.S. degree in electrical engineering and applied physics from Linköping University, Linköping, Sweden, in 1986, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 1990. He has held research positions with the Department of Electrical Engineering, Linköping University, the Information Systems Laboratory, Stanford University, the Katholieke Universiteit Leuven, Leuven, Belgium, and the University of Luxembourg, Luxembourg City, Luxembourg. From 1996 to 1997, he was the Director of Research with ArrayComm, Inc., a start-up in San Jose, CA, USA, based on his patented technology. In 1991, he was appointed as a Professor of signal processing with the Royal Institute of Technology (KTH), Stockholm, Sweden. From 1992 to 2004, he was the Head of the Department for Signals, Sensors, and Systems, KTH, and from 2004 to 2008, he was the Dean of the School of Electrical Engineering, KTH. He is currently the Director for the Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg. As Digital Champion of Luxembourg, he acts as an Adviser to the European Commission. He was the recipient of the IEEE Signal Processing Society Technical Achievement Award in 2011 and the European Research Council advanced research grant twice, in 2009–2013 and in 2017–2022. He has coauthored journal papers which received the IEEE Signal Processing Society Best Paper Award in 1993, 2001, 2006, and 2013, rspectively, and seven other IEEE conference papers Best Paper awards. He was the Editor-in-Chief of *EURASIP Signal Processing Journal*, an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING, and the Editorial Board of the *IEEE Signal Processing Magazine*. He is currently a Member of the Editorial Boards of *EURASIP Journal of Advances Signal Processing* and *Foundations and Trends of Signal Processing.* He is a Fellow of the EURASIP.