Energy-Efficient UAV-Aided Computation Offloading on THz Band: A MADRL Solution

Yuanjian Li*, A. S. Madhukumar*, Tan Zheng Hui Ernest†, Gan Zheng‡, Walid Saad§, A. Hamid Aghvami¶
*Nanyang Technological University, Singapore †Agency for Science, Technology and Research, Singapore
‡University of Warwick, U.K. §Virginia Tech, USA ¶King's College London, U.K.

***Abstract*—In this paper, the problem of energy-efficient unmanned aerial vehicle (UAV)-assisted computation offloading over the Terahertz (THz) spectrum is investigated. In the studied system, several UAVs are deployed as edge servers to aid task executions for multiple energy-limited computation-scarce terrestrial user equipments (UEs). Then, an expected energy efficiency maximization problem is formulated, aiming to jointly optimize UAVs' trajectories, UEs' local central processing unit (CPU) clock speeds, UAV-UE associations, time slot slicing, and UEs' offloading powers. To tackle the considered multi-dimensional optimization problem, the duo-staggered perturbed actor-critic with modular networks (DSPAC-MN) solution in a multi-agent deep reinforcement learning (MADRL) setup, is proposed and tailored, after mapping the original problem into a stochastic (Markov) game. Compared to representative benchmarks in simulations, e.g., multi-agent deep deterministic policy gradient (MADDPG) and multi-agent twin-delayed DDPG (MATD3), the proposed DSPAC-MN can achieve the optimal performance of average energy efficiency, while ensuring 100% safe flights.**

I. Introduction

MULTI-ACCESS edge computing (MEC), inheritor of the renowned mobile edge computing, is deemed as a promising building block for achieving real-time processing, improved user experience, enhanced privacy and security, and refined cost efficiency for the sixth generation (6G) wireless systems in general and the Internet of things (IoT) in particular. It is well understood that user equipments (UEs) within IoT are inherently limited in either on-board power supply or available computing resources, yet they demand prolonged operation durations and time-efficient task executions, highlighting the importance of sustainability in terms of, e.g., energy efficiency [1]. To achieve seamless and high-quality task execution performance within computation-intensive, delay-sensitive and energy-scarce IoT application scenarios, inter alia, industrial IoT (IIoT), environmental monitoring systems and remote sensor networks, MEC can provide the substantial backbone by shifting computational resources closer to UEs [2]. By offloading intensive computation tasks to MEC servers in proximity via various wireless communication technologies, the pressure of exceeding data traffic at UEs can be alleviated, while the corresponding data execution and/or energy efficiency can be further elevated. Moreover, edge servers can be deployed on unmanned aerial vehicles (UAVs) to take advantage of the drones' extra degree of freedom (DoF) of configurable mobility and line-of-sight (LoS)-dominant air-to-ground (A2G) wireless links [3], which can help extend high-performance and robust MEC services to complement terrestrial MEC systems that might be inaccessible or overwhelmed in cases of, e.g., infrastructure-sparse areas, temporary demand surges or natural disasters. Meanwhile, blessed by its promising potential to aid in achieving higher transmission rate, lower communications latency, more accurate localization and more precise sensing, Terahertz (THz) communication has been publicly acknowledged as a competitive candidate and core enabler to fulfil the proliferating demands of 6G networks. In particular, THz spectrum ranging from 100 Gigahertz (GHz) to 10 THz, between bands of millimetre wave (mmWave) and infrared (IR), can offer ultra-wide bandwidth to support unprecedented superfast transmission rate of Terabits per second (Tbps) [4]. Due to its high carrier frequency, the size of THz antenna elements and the corresponding inter-element spacing can be quite insignificant, which promises much greater beamforming and multiplexing gains. Therefore, THz technology is regarded as one of the key components for enabling fine-tuned MEC services, e.g., task offloading, with low latency and high data rate. To deal with constrained propagation distance, blockage vulnerability and limited coverage of THz-aided MEC systems, one promising solution is UAV-aided MEC on the THz band, which benefits from short-range LoS-prominent G2A links established by UAV's manageable mobility.

As of now, little research effort has been poured into the inter-domain energy-efficient UAV-aided MEC systems on the THz band, not to mention that artificial intelligence (AI)-native intelligent solutions adaptive to dynamically time-varying wireless environments are intrinsically challenging and severely lacking. To fill this research gap, this paper investigates UAV-aided MEC networks over the THz band, and proposes an AI-native algorithm to solve the formulated average energy efficiency maximization problem.

II. System Model and Problem Formulation

We consider an IoT system, e.g., Fig. 1(c), consisting of a set $\mathcal{G}=\{1,\cdots,g,\cdots,G\}\subset\mathbb{Z}^+$ of G terrestrial UEs located in a sub-area $\mathbb{k}$. The 2-dimensional (2D) horizontal coordinate of UE $g\in\mathcal{G}$ is defined as $\mathbf{q}_g=(x_g,y_g)\in\mathbb{R}^{1\times 2}$. Without loss of generality, the considered sub-region $\mathbb{k}$ is modelled as a square space, captured by $[\check{x},\hat{x}]\times[\check{y},\hat{y}]\times z_u$, where z_u represents the height of this square space, and $\check{\cdot}$ and $\hat{\cdot}$ indicate the lower and upper borders of $\mathbb{k}$, respectively. The

This research is supported by the National Research Foundation, Singapore, and Infocomm Media Development Authority under its Future Communications Research & Development Programme (FCP-NTU-RG-2022-014) and the National Research Foundation, Singapore, under its Competitive Research Programme (NRF-CRP23-2019-0005).

UEs periodically generate computation tasks, maintained by dynamic first-in-first-out (FIFO) data queues. Constrained by UE's limited energy supply and/or constrained local computation capacity, we deploy a set $\mathcal{U}=\{1,\cdots,u,\cdots,U\}\subset\mathbb{Z}^+$ of U drones as mobile edge servers to aid in data processing, by providing UEs with accessible computation offloading service. The UAVs' 2D horizontal coordinates are given by $\mathbf{q}_u=(x_u,y_u)\in\mathbb{R}^{1\times 2}$, while their flight altitudes are identically set to be z_u. We adopt a fine-grained computation offloading strategy, whereby parts of tasks can be either executed locally or offloaded to UAV servers. To achieve high-speed and low-latency task offloading performance, the UEs are supposed to operate in the THz band. We will exploit drones' configurable mobility, to help establish short-range LoS-dominant G2A THz links. Moreover, we divide the UAV's flight duration T_u into N even chunks. Then, the length of each time slot is weighed by $\delta_t=T_u/N$. The initial location and destination of each UAV are represented by $\mathbf{q}_u[0]=\mathbf{q}_u^I$ and $\mathbf{q}_u[N\delta_t]=\mathbf{q}_u^D$, respectively. In this regard, the trajectory of each UAV can be traced by $\mathbf{q}_u[n\delta_t],\forall n\in\{0,1,\cdots,N\}\subset\mathbb{N}$. For conciseness, $\mathbf{q}_u[n\delta_t]$ is abbreviated as $\mathbf{q}_u[n]$ thereafter. In our system model, each UAV's mobility is subject to

$$\mathbf{q}_u[n+1]=\mathbf{q}_u[n]+\delta_t\mathbf{v}_u[n],\forall n\in[0,N-1], \tag{1a}$$
$$\check{\mathbf{q}}\preceq\mathbf{q}_u[n]\preceq\hat{\mathbf{q}},\forall n\in[0,N], \tag{1b}$$
$$\mathbf{q}_u[0]=\mathbf{q}_u^I,\mathbf{q}_u[N]=\mathbf{q}_u^D, \tag{1c}$$
$$\|\mathbf{q}_u-\mathbf{q}_{u'\in\{\mathcal{U}\setminus u\}}\|\geq D,\forall n\in[1,N], \tag{1d}$$
$$\mathbf{v}^-\leq\|\mathbf{v}_u[n]\|\leq\mathbf{v}^*,\forall n\in[1,N], \tag{1e}$$

where $\check{\mathbf{q}}=(\check{x},\check{y})\in\mathbb{R}^{1\times 2}$, $\hat{\mathbf{q}}=(\hat{x},\hat{y})\in\mathbb{R}^{1\times 2}$, $\mathbf{v}_u\in\mathbb{R}^{1\times 2}$ indicates UAV's velocity covering both forward direction and propulsion speed, $\preceq$ captures element-wise inequality, and $\mathbf{v}^-$ and $\mathbf{v}^*$ represent the minima and maxima of UAV's propulsion speed, respectively. Note that (1d) handles potential collisions among UAVs by introducing the safety distance D.

A. Computation Model

1) Terrestrial Computing Model: In each time slot, the amount of locally computed data in bits is given by $d_g^l[n]=\delta_t f_g[n]/c_g,\forall g\in\mathcal{G},\forall n\in[1,N]$, where $f_g[n]$ in cycles per second (cycle/s or Hertz) captures the computation capability, i.e., CPU clock speed or CPU-cycle frequency, and c_g represents the number of CPU cycles needed for handling one bit of data, i.e., the computation intensity/workload. In practice, the local computing capability must be bounded, as follows

$$f_g[n]\in[0,f_g^*],\forall g\in\mathcal{G},\forall n\in[1,N], \tag{2}$$

where f_g^* is the maximum CPU clock speed of local computing.

2) Computation Offloading Model: We assume that UAVs can fly near UEs that require computation offloading, where short-range LoS-dominant G2A transmission links can be potentially established. To manage the uplink pairing between UEs and UAVs, we define a UAV-UE association indicator ψ_g^u, whereby $\psi_g^u[n]=1$ when UE g offloads its computation task to UAV u, and $\psi_g^u[n]=0$ otherwise. Then, we have

$$\psi_g^u[n]\in\{0,1\},\forall g\in\mathcal{G},\forall u\in\mathcal{U},\forall n\in[1,N]. \tag{3}$$

As per (3), over each time slot, a UAV is allowed to be associated with multiple UEs. Unfortunately, constrained by its limited onboard energy supply and/or computational capability, it might be impractical to consider that each UAV can provide computing service for multiple UEs simultaneously. To handle this issue, a time slot slicing (TS2)-based approach is adopted, where a τ_g^u portion of each segmented time slot will be granted to UE g that is paired with UAV u, subject to

$$\tau_g^u[n]\in[0,1],\forall g\in\mathcal{G},\forall u\in\mathcal{U},\forall n\in[1,N], \tag{4a}$$
$$\sum_{g'\in\{\arg\limits_g\psi_g^u[n]=1\}}\tau_{g'}^u[n]\leq 1,\forall u\in\mathcal{U},\forall n\in[1,N], \tag{4b}$$
$$\sum_{u'\in\{\arg\limits_u\psi_g^u[n]=1\}}\tau_g^{u'}[n]\leq 1,\forall g\in\mathcal{G},\forall n\in[1,N], \tag{4c}$$
$$\sum_{g'\in\{\arg\limits_g\psi_g^u[n]=0\}}\tau_{g'}^u[n]=0,\forall u\in\mathcal{U},\forall n\in[1,N], \tag{4d}$$

in which (4a) specifies that τ_g^u is a non-negative real value that is no greater than one, (4b) and (4c) capture the practical consideration that the sum of non-overlapping sliced portions cannot exceed the length of time slot, and (4d) ensures that the non-associated UEs could not initiate task offloading.

Then, the uplink offloading rate in bits per second (bits/s) from UE g to UAV u can be given by $R_g^u[n]=B\log\big(1+P_g[n]h_g^u[n]/\sigma^2\big),\forall g\in\mathcal{G},\forall u\in\mathcal{U},\forall n\in[1,N]$, where B represents the transmission bandwidth, $h_g^u[n]$ indicates the channel gain that will be defined in Section II-D, σ^2 is from the additive white Gaussian noise (AWGN) that follows $\mathcal{CN}(0,\sigma^2)$, the transmit power $P_g[n]$ is subject to

$$P_g[n]\in[0,P_g^*],\forall g\in\mathcal{G},\forall n\in[1,N], \tag{5}$$

and P_g^* is the maximum transmit power at UE g.

Furthermore, the amount of offloaded data from UE g to UAV u will be $o_g^u[n]=\psi_g^u[n]\tau_g^u[n]\delta_t R_g^u[n]/c_o$, where the factor c_o represents the transmission overhead. Moreover, it is straightforward to calculate the required CPU-clock speed for successfully handling the offloaded data as $f_g^u[n]=c_u o_g^u[n]/(\tau_g^u[n]\delta_t)=c_u\psi_g^u[n]R_g^u[n]/c_o$. However, the offloaded data cannot exceed UAV server's computation capacity, leading UAVs CPU-clock speed to be subject to

$$f_g^u[n]=\min\left\{\frac{c_u\psi_g^u[n]R_g^u[n]}{c_o},f_u^*\right\},\forall g\in\mathcal{G},\forall u\in\mathcal{U},\forall n\in[1,N], \tag{6}$$

where c_u is the number of CPU cycles required for dealing with one bit of data at UAVs, and f_u^* represents the capacity of the UAV's CPU-cycle frequency. Then, the amount of offloaded data that can be processed at the drone servers, i.e., the corresponding executed data, is $d_g^u[n]=\tau_g^u[n]\delta_t f_g^u[n]/c_u$.

B. Task Queue Model

For time slot $n\in[1,N]$, new instances of the generated computation task at UE $g\in\mathcal{G}$, i.e., $A_g[n]\in\mathbb{R}^+$ in bits, will arrive to be accommodated by the local task queue. We assume that $A_g[n]$ is subject to a bounded second-order distribution [5], i.e., $\mathbb{E}\{|A_g[n]|^2\}\leq+\infty$. Then, the task queue model, i.e., the backlog [6], can be tracked by

$$\mathcal{Q}_g[n+1]=\min\big\{\mathcal{Q}_g[n]-d_g[n]+A_g[n],A_g^*\big\},\forall g\in\mathcal{G},\forall n\in[0,N], \tag{7}$$

where $A_g^*\in\mathbb{R}^+$ represents the queue capacity and $\mathcal{Q}_g[0]=0$ is adopted. Note that the total amount of executed task for

the UE g, i.e., $d_g[n]$ in (7), is defined as $d_g[n] = d_g^l[n] + \sum_{u\in\mathcal{U}} d_g^u[n]$. To ensure the task queue's stability, we have

$$\lim_{N\to+\infty} \frac{1}{N} \sum_{n=1}^{N} \sum_{g\in\mathcal{G}} \mathbb{E}\{\mathcal{Q}_g[n]\} < +\infty, \tag{8}$$

where the expectation is taken over the randomness of system parameters, e.g., $A_g[n]$ and $d_g[n]$. In order to avoid violating the information causality, i.e., the data being processed during time slot n could not exceed the amount of data currently being queued in the backlog, we have

$$\mathcal{Q}_g[n] \geq d_g[n], \forall g \in \mathcal{G}, \forall n \in [1, N]. \tag{9}$$

C. Energy Cost Model

1) Energy Consumption of Ground UEs: According to the dynamic voltage and frequency scaling (DVFS)-based computation power consumption model [7], the energy cost for local computation and task offloading will be $E_g^l[n] = \gamma_g (f_g[n])^3 \delta_t + E_g^* + \sum_{u\in\mathcal{U}} \tau_g^u[n] P_g[n] \delta_t$, where the constant γ_g stands for the effective capacitance coefficient related to hardware architecture, and E_g^* captures the power cost from other system members such as the random access memory.

2) Energy Consumption from Computation at UAVs: The energy consumed by the UAVs for conducting edge computing on the offloaded tasks is given by $E_u[n] = \gamma_u \delta_t \sum_{g\in\mathcal{G}} \min\left\{\psi_g^u[n]\left(c_u R_g^u[n]/c_o\right)^3, (f_u^*)^3\right\} \tau_g^u[n] + E_u^*$, where γ_u represents the effective capacitance coefficient, and E_u^* indicates energy consumption from other system components.

3) Energy Consumption from UAV's Mobility: Following [6], UAV's propulsion power consumption in time slot n when a UAV u has a velocity $\mathbf{v}_u[n]$ can be calculated by $P_{\text{mo}}(\mathbf{v}_u[n]) = P_{bp}\left(1 + 3\|\mathbf{v}_u[n]\|^2/(v_{tip}^2)\right) + \frac{1}{2}\varrho\|\mathbf{v}_u[n]\|^3 + P_{ip}\sqrt{\sqrt{1 + \|\mathbf{v}_u[n]\|^4/(4v_0^4)} - \|\mathbf{v}_u[n]\|^2/(2v_0^2)}$, where $\varrho = \varrho_0\varrho_1\varrho_2\varrho_3$, ϱ_0 is the fuselage drag ratio, ϱ_1 means the air density, ϱ_2 indicates the rotor solidity, ϱ_3 represents the rotor disc area, v_{tip} denotes the rotor blade's tip speed, and v_0 is the average rotor induced velocity for hovering. Moreover, $P_{bp} = \varrho_1\varrho_2\varrho_3(\varrho_4\varrho_5)^3\varrho_6/8$ represents the blade profile power, while $P_{ip} = (1+\varrho_7)\varrho_8^{\frac{3}{2}}/\sqrt{2\varrho_1\varrho_3}$ captures the induced power for hovering, where ϱ_4 is the blade angular velocity in randians/s, ϱ_5 is the rotor radius in meters, ϱ_6 represents the profile drag coefficient, ϱ_7 is the incremental correction factor to the induced power, and ϱ_8 represents the UAV's weight in Newton [8]. Then, the UAV's propulsion energy consumption within a time slot can be expressed as $E_u^p[n] = P_{\text{mo}}(\mathbf{v}_u[n])\delta_t$.

D. THz Channel Model

The adopted pathloss model for THz G2A transmissions is $h_g^u[n] = 1/\text{PL}\left(f_c, q_g^u[n]\right)$, where $\text{PL}\left(f_c, q_g^u\right)$ in dB is subject to the carrier frequency f_c and the G2A propagation distance $q_g^u[n] = \sqrt{\|\mathbf{q}_u[n] - \mathbf{q}_g\|^2 + z_u^2}$. According to Friis's principle and Beer-Lambert law [9], we have $\text{PL}\left(f_c, q_g^u[n]\right) = 10\kappa(f_c,\mu) q_g^u[n] \lg(e) + 20\lg(4\pi f_c \left(q_g^u[n]\right)^{(\Xi/2)}/(C\sqrt{G_t G_r}))$, where C is the speed of light, $\kappa(f_c,\mu)$ is the absorption coefficient describing the relative absorbing area of the molecules in the medium per unit volume, μ denotes the volume mixing ratio of water vapor (humidity), e represents the Euler's number, Ξ regulates the large-scale spreading loss exponent, and G_t and G_r measure the antenna gains with respect to (w.r.t.) an isotropic radiator of the transmitting and receiving antennas, respectively. In the frequency range of 0.275-0.4 THz, we have $\kappa(f_c,\mu) = \kappa^a(f_c,\mu) + \kappa^b(f_c,\mu) + \kappa^c(f_c)$, in which $\kappa^a(f_c,\mu) = 0.2205\mu(0.1303\mu + 0.0294)/((0.4093\mu + 0.0925)^2 + (f_c/(100C) - 10.835)^2)$, $\kappa^b(f_c,\mu) = 2.014\mu(0.1702\mu + 0.0303)/((0.537\mu + 0.0956)^2 + (f_c/(100C) - 12.664)^2)$, $\kappa^c(f_c) = 5.54 \times 10^{-37} f_c^3 - 3.94 \times 10^{-25} f_c^2 + 9.06 \times 10^{-14} f_c - 6.36 \times 10^{-3}$, and $\mu = \phi p_\omega^*/(100p)$. Moreover, $\phi p_\omega^*/100$ represents the partial pressure of water vapor, where p_ω^* is determined by Buck equation, i.e., $p_\omega^* = 6.1121(1.0007 + 3.46 \times 10^{-6} p)\exp\left(17.502\,(T - 273.15)/(T - 32.18)\right)$. Herein, ϕ in percentage, T in degrees Kelvin (°K), and p in hectoPascal (hPa) capture the atmospheric conditions of relative humidity, temperature and pressure, respectively.

E. Problem Formulation

The weighted sum of energy consumption within each time slot for local computations, task offloading, data execution at edge servers, and the UAVs' propulsion will be $E[n] = \sum_{g\in\mathcal{G}} E_g^l[n] + \sum_{u\in\mathcal{U}} \left(E_u[n] + \mathfrak{R} E_u^p[n]\right)$, where $\mathfrak{R}$ captures the energy regulation factor. Moreover, the total amount of data executed within each time slot can be given by $d[n] = \sum_{g\in\mathcal{G}} d_g[n]$. UAVs and UEs have limited onboard energy supply, which makes it critical to properly handle energy efficiency for UAV-aided computation offloading. Towards this end, we seek to maximize the expected energy efficiency for UAV-aided computation offloading alongside flights, stated as

$$\begin{aligned} \max_{\{\mathbf{v}_u[n], f_g[n], \psi_g^u[n], \tau_g^u[n], P_g[n]\}} & \frac{1}{N} \sum_{n=1}^{N} \frac{d[n]}{E[n]}, \\ \text{s.t.} \quad & (1), (2), (3), (4), (5), (6), (8), (9) \end{aligned} \tag{10}$$

where the UAV's velocity $\mathbf{v}_u[n]$, the local CPU-cycle frequency $f_g[n]$, the UAV-UE association indicator $\psi_g^u[n]$, the TS2 factor $\tau_g^u[n]$ and the offloading power $P_g[n]$ are targeted to be jointly optimized in a multi-variable optimization fashion.

III. The Proposed Distributed DRL Solution

To solve the energy efficiency maximization problem (10) considering multiple UAVs and UEs by multi-agent deep reinforcement learning (MADRL) frameworks, it is crucial to map the original problem into Markov game, stated as follows.

1) State Space: The state space includes UAV agents' locations/coordinates and the amount of data remaining in the backlogs of the UEs. The state at time slot n is given by

$$s[n] = \{\mathbf{q}_u[n], \mathcal{Q}_g[n]\}, \forall g \in \mathcal{G}, \forall u \in \mathcal{U}, \forall n \in [0, N], \tag{11}$$

2) Action Space: After observing the current state $s[n]$, as per the latest tuned policy, each UAV agent must choose its action, including the velocity $\mathbf{v}_u[n]$, the ground UEs' local CPU clock speed $f_g[n]$, the UAV-UE association indicator $\psi_g^u[n]$, the TS2 portion of segmented time slot $\tau_g^u[n]$, and the ground UEs' transmit powers $P_g[n]$. The action for $\forall g \in \mathcal{G}, \forall u \in \mathcal{U}, \forall n \in [1, N]$ is defined as

$$a_u[n] = \{\mathbf{v}_u[n], f_g[n], \psi_g^u[n], \tau_g^u[n], P_g[n]\}, \tag{12}$$

where all other elements are continuous values except the binary UAV-UE pairing factor $\psi_g^u[n] \in \{0,1\}$. To handle the challenge raised by mixed integers, the UAV-UE association variable is relaxed from binary values into a ranged floating-point parameter, i.e., $\psi_g^u[n] \in (0,1)$.

3) Reward Function and Reward Shaping: As the UAV agents are supposed to cooperate to serve the ground UEs with high-quality computation offloading, the corresponding mapped stochastic game has to be a *fully cooperative* Markov one, where the reward functions $\{\mathcal{R}_u\}_{u\in\mathcal{U}}$ are typically identical, i.e., $\mathcal{R}_1 = \cdots = \mathcal{R}_U$, followed by common returns $R_1[s|\boldsymbol{\pi}] = \cdots = R_U[s|\boldsymbol{\pi}]$. In this regard, the UAV agents share the unified goal of maximizing the common return. Then, in line with the objective of problem (10), it is straightforward to define the immediate reward as $r[n] = d[n]/E[n]$. Nonetheless, the global reward for cooperative multi-agent learning might lead to less informative or sparse training signals for individual agents, which will cause inefficient or even trivial policy training performance. To circumvent this issue, *reward shaping* technique that aims to provide additional feedback for agents to help stimulate desired behaviours, is adopted. To deal with the sparse reward issue of occasional cases where UAVs crash into the boundaries or collide with others, additional feedbacks of *direction-aware* reward components are added. Specifically, the direction of generated velocity $\mathbf{v}_u[n]$ can be derived as $\mathbf{v}_u[n]/\|\mathbf{v}_u[n]\|$, while that of UAV located at $\mathbf{q}_u[n]$ towards the centroid of $\mathbb{k}$ is given by $(\mathbf{q}_{\mathbb{k}} - \mathbf{q}_u[n])/\|\mathbf{q}_{\mathbb{k}} - \mathbf{q}_u[n]\|$. Then, the similarity of these two normalized directions can be reflected by their dot product, given by $\jmath_u^{\mathbb{k}}[n] = \mathbf{v}_u[n]\cdot(\mathbf{q}_{\mathbb{k}} - \mathbf{q}_u[n])/(\|\mathbf{v}_u[n]\|\cdot\|\mathbf{q}_{\mathbb{k}} - \mathbf{q}_u[n]\|) \in [-1,1]$. To prevent UAVs from being trapped in flying straight towards the centroid, a $\angle 45^\circ$ tolerance is posed by adjusting $\jmath_u^{\mathbb{k}}[n] = 1$ in the case of $\jmath_u^{\mathbb{k}}[n] \geq \sqrt{2}/2$. When the next locations of any two UAVs are within $T_c D$ distance, a direction-aware collision penalty will be triggered, where T_c represents the range of activating the direction-aware collision penalty. Specifically, the corresponding similarity can be measured by $\jmath_u^{u'}[n] = \mathbf{v}_u[n]\cdot\mathbf{v}_{u'}[n]/(\|\mathbf{v}_u[n]\|\cdot\|\mathbf{v}_{u'}[n]\|) \in [-1,1]$. To avoid the direction-aware collision penalty from disturbing too much on the overall policy, we force $\jmath_u^{u'}[n] = -1$ for $\jmath_u^{u'}[n] \leq -\sqrt{2}/2$, and $\jmath_u^{u'}[n] = 0$ otherwise. Moreover, to incentivize executing more data while enhancing energy efficiency, and help maintain a stable task queue required by (8), an additional feedback of *queue-aware* reward component is introduced, given by $\digamma_g[n] = A_g^*/\mathcal{Q}_g[n+1]$.

Then, the original sparse reward can be reformulated as (13), where p_o is the outbound penalty for UAVs stepping out of the sub-region $\mathbb{k}$, p_c is the collision penalty for UAV colliding with other UAVs, and the Dirichlet function $\mathbb{1}_{(\cdot)}$ is redefined to describe whether a constraint is violated, i.e., $\mathbb{1}_{(\cdot)} = 1$ if so, and $\mathbb{1}_{(\cdot)} = 0$ if not. The logarithmic function is invoked to aid in stabilizing the training process.

4) Distributed Actors: For clarity, the generated action of the parametrized actor u corresponding to UAV server $u \in \mathcal{U}$ over time slot n is given by

$$a_u[n] = \pi_u\left(s[n]|\boldsymbol{\Theta}^{a_u}\right), \tag{14}$$

where $\boldsymbol{\Theta}^{a_u}$ is the trainable/tunable parameters, e.g., weights and biases inside neural networks (NNs), of the actor. Then, the joint action can be formulated as $\mathbf{a}[n] = [a_u[n]]_{u\in\mathcal{U}}$.

5) Shared Critic: To deal with the non-stationary issue of multi-agent environment, and due to the cooperative learning nature of the formulated stochastic game, a global critic network that is shared by all distributed actors is implemented. In this regard, the shared critic calculates the Q value of the pair of state $s[n]$ and joint action $\mathbf{a}[n]$, expressed as

$$Q(s[n],\mathbf{a}[n]|\boldsymbol{\Theta}^c) = \mathbb{E}\left[\sum_{n=0}^{+\infty}\gamma^n r[n]|s_0 = s[n], \mathbf{a}_0 = \mathbf{a}[n]\right], \tag{15}$$

where $\boldsymbol{\Theta}^c$ is the trainable parameters of the shared critic.

6) Modularized Inputs: To handle the dimension imbalance matter, the inputs of both the distributed actors and the shared critic are modularized before being fed into NNs. For ease of exposition, the i-th parametrized module is represented as $M_i(x_i|\boldsymbol{\Theta}^{M_i}) : \mathbb{R}^{1\times|x_i|} \to \mathbb{R}^{1\times|M_i|}$, in which $i \in [1, N_M] \subset \mathbb{Z}^+$, N_M measures the total number of modules, x_i is the input to module i, $|x_i|$ represents the dimension of x_i, $|M_i|$ weighs that of the current module's output, and $\boldsymbol{\Theta}^{M_i}$ indicates the corresponding tunable parameters. Then, the ultimate results of the modular network are captured by a mapped composition of outputs from individual modules, i.e.,

$$\begin{aligned}\boldsymbol{M}\left(x|\boldsymbol{\Theta}^{\boldsymbol{M}}\right) : \mathbb{R}^{1\times\sum_i|x_i|} \to \mathbb{R}^{1\times|\boldsymbol{M}|} = f_{\boldsymbol{M}}\big[&M_1\left(x_1|\boldsymbol{\Theta}^{M_1}\right),\\ &M_2\left(x_2|\boldsymbol{\Theta}^{M_2}\right),\cdots,M_{N_M}\left(x_{N_M}|\boldsymbol{\Theta}^{M_{N_M}}\right)|\boldsymbol{\Theta}^{f_{\boldsymbol{M}}}\big],\end{aligned} \tag{16}$$

where $f_{\boldsymbol{M}}$ with trainable parameter $\boldsymbol{\Theta}^{f_{\boldsymbol{M}}}$ aggregates individual modules' outputs, $|\boldsymbol{M}| = \sum_i |M_i|$ measures dimension of modular network's output, and x and $\boldsymbol{\Theta}^{\boldsymbol{M}}$ denote collections of $\{x_i\}$ and $\{\boldsymbol{\Theta}^{M_i}, \boldsymbol{\Theta}^{f_{\boldsymbol{M}}}\}, i \in [1, N_M]$, respectively.

7) Perturbed Actors: To manage poor/insufficient exploration issue, exploratory actors are introduced. Specifically, each online distributed actor sorely learns its policy steered by the shared critic, while a copy of it, defined as the exploratory actor perturbed by parameter-wise noises, is adopted to generate the action. Besides, the tunable parameters of the exploratory actor keep being updated alongside the training process, as per $\boldsymbol{\Theta}^{p_u} \leftarrow \boldsymbol{\Theta}^{a_u} + \boldsymbol{\Theta}^{p}$, where $\boldsymbol{\Theta}^{p_u}$ captures the learnable parameters of the u-th exploratory actor, and each element inside $\boldsymbol{\Theta}^{p}$ with identical shape of $\boldsymbol{\Theta}^{a_u}$ follows a normal distribution with zero mean and standard deviation σ_p.

8) Proposed MADRL Solution: In short, the skeleton of the proposed MADRL framework involves U distributed agents and a shared global critic. Specifically, each agent u has three components, i.e., the *online actor* $\pi_u\left(s[n]|\boldsymbol{\Theta}^{a_u}\right)$ that is learning from the interactions with the environment, the *shadow actor* $\pi_u(s[n]|\boldsymbol{\Theta}^{a_u^-})$ whose tunable parameter $\boldsymbol{\Theta}^{a_u^-}$ is a soft copy of the online actor's as per $\boldsymbol{\Theta}^{a_u^-} \leftarrow \tau_s\boldsymbol{\Theta}^{a_u} + (1-\tau_s)\boldsymbol{\Theta}^{a_u^-}$,

$$r[n] = \begin{cases}\frac{1}{U}\sum_{u\in\mathcal{U}}\Big(\jmath_u^{\mathbb{k}}[n] + \sum_{u'\in\mathcal{U}\setminus u}\jmath_u^{u'}[n]\mathbb{1}_{(\|\mathbf{q}_u[n+1]-\mathbf{q}_{u'}[n+1]\|\leq T_c D)}\Big) + \log_2\Big(1+\frac{d[n]}{E[n]}\Big) + \log_2\Big(1+\frac{1}{G}\sum_{g\in\mathcal{G}}\digamma_g[n]\Big), & \text{Constraints Met}\\ -\big[p_o\mathbb{1}_{(1b)} + p_c\mathbb{1}_{(1d)}\big], & \text{Otherwise}\end{cases} \tag{13}$$

and the *exploratory actor* $\pi_u\left(s[n]|\mathbf{\Theta}^{p_u}\right)$. It is worth noting that the Polyak averaging coefficient τ_s is adopted herein to manage the rate at which the shadow network parameters are updated towards those of the online network, which is typically an insignificant value. The use of a shadow actor network allows us to generate the next action given the next state for the purpose of computing the target Q value amid critic network training. In this way, it is beneficial to enhance the *stability* and *consistency* of the overall training process by providing a more stable policy than the online actor. The perturbed exploratory actor is responsible for perceiving the state of environment and then delivering the corresponding action to formulate the joint action. While the exploratory actor does facilitate enhanced environmental exploration by the agents, imposing *non-parametric exploration noise* on its output further amplifies the scope of exploration, driving a more extensive performance in probing the environment. Therefore, given the current state $s[n]$, the corresponding individual action generated by each agent should be reformulated from (14) to $a_u[n] = \pi_u\left(s[n]|\mathbf{\Theta}^{p_u}\right) + \mathbf{N}$, where $\mathbf{N}$ denotes the exploration noise, which may come from popular random distributions, e.g., Gaussian/Normal or the Ornstein-Uhlenbeck (OU) process. Then, the resulting joint action can be formulated as $\mathbf{a}[n] = [a_u[n]]_{u\in\mathcal{U}}$.

Alongside the online interaction between UAV-agents and the environment, once the number of stored experiences in the experience replay buffer becomes greater than the size of mini-batch, the training process will be embarked upon replaying experienced transitions on a mini-batch sampling basis. Then, mean squared error (MSE) loss function of the global critic is

$$\ell(\mathbf{\Theta}^{c_j}) = \mathop{\mathbb{E}}_{(s_m,\mathbf{a}_m,s_{m+1},r_m)\in\mathcal{D}\sim\mathcal{B}} \left[\mathbf{y}_m - Q(s_m,\mathbf{a}_m|\mathbf{\Theta}^{c_j})\right]^2, \quad (17)$$

in which

$$\mathbf{y}_m = r_m + \gamma \min_{j=0,1} Q\left(s_{m+1}, \oplus_{u=1}^{U}\left\{\pi_u\left(s_{m+1}|\mathbf{\Theta}^{a_u^-}\right)+\mathbf{N}^-\right\}|\mathbf{\Theta}^{c_j^-}\right), \quad (18)$$

measures the target Q value, $\oplus$ denote the concatenation operation for vectors, the expectation operator $\mathbb{E}$ calculates over the sampled experiences in the mini batch $\mathcal{D}$ of size $\mathbf{D}$, and $\mathbf{\Theta}^{c_j}$, $\mathbf{\Theta}^{c_j^-}$ and $\mathbf{N}^-$ indicate the critic $j \in \{0,1\}$'s and the corresponding shadow critic's tunable parameters, and the additive noise for shadow actors' outputs, respectively. We note that in (18), the *capped duo-Q estimation* technique is adopted, which aims to take the more conservative (lower) estimate to ensure that the Q-value updates are less susceptible to overestimations. Moreover, a *shadow policy tempering* is invoked to serve as a regularizer for promoting the target Q value not to be much obsessed by spiky estimates via generating the next action by the ruffled shadow policies, which is beneficial to prevent rough oscillations in policy updates. Then, the mini-batch gradient descent technique rule will be posed to update the trainable parameters of the duo critics, as per $\mathbf{\Theta}^{c_j} \leftarrow \mathbf{\Theta}^{c_j} - \alpha_c \nabla_{\mathbf{\Theta}^{c_j}} \ell\left(\mathbf{\Theta}^{c_j}\right)$, where α_c denote the learning rate and $\nabla_{\mathbf{\Theta}^{c_j}} \ell\left(\mathbf{\Theta}^{c_j}\right)$ calculates the gradients of MSE loss function $\ell\left(\mathbf{\Theta}^{c_j}\right)$ w.r.t. $\mathbf{\Theta}^{c_j}$.

In contrast, the online actor within each UAV agent aims to maximize its expected return, via mini-batch gradient ascent approach, given by

$$\mathbf{\Theta}^{a_u} \leftarrow \mathbf{\Theta}^{a_u} + \alpha_a \mathop{\mathbb{E}}_{s_m\in\mathcal{D}}\left[\nabla_{\pi_u} Q\left(s_m, \oplus_{u=1}^{U}\{\pi_u(s_m|\mathbf{\Theta}^{a_u})\}|\mathbf{\Theta}^{c_0}\right) \times \nabla_{\mathbf{\Theta}^{a_u}} \pi_u(s_m|\mathbf{\Theta}^{a_u})\right], \quad (19)$$

where α_a indicates the learning rate and the global critic 0 is anchored to perform the chain rule of calculating the gradients. To reduce the computational overhead for actor updates and meanwhile contain volatility of learnt policies, *staggered policy renewal* technique is adopted, where the online actors, the shadow actors/critics will be updated every N_s times the online critics are trained. The tailored MADRL solution is thereafter termed the duo-staggered perturbed actor-critic with modular networks (DSPAC-MN), wrapped in Algorithm 1.

Algorithm 1: The Proposed DSPAC-MN Solution

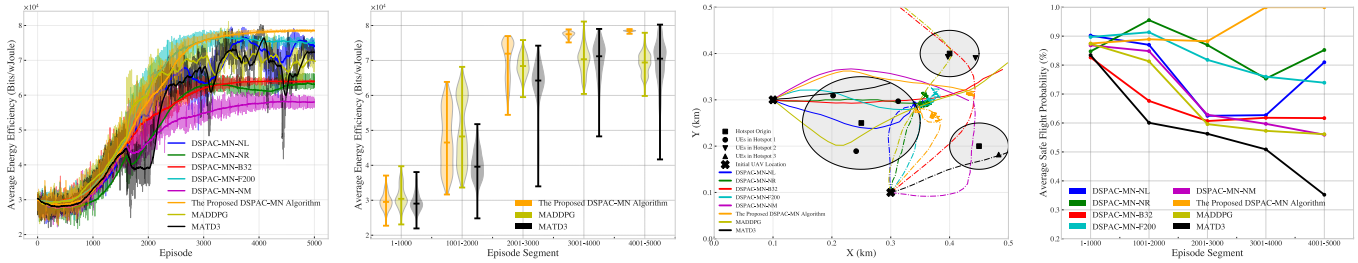
Initialization: Initialize online NNs' layers, as per OWI. Synchronize the exploratory actor networks and shadow networks via $\mathbf{\Theta}^{p_u} \leftarrow \mathbf{\Theta}^{a_u}$, $\mathbf{\Theta}^{a_u^-} \leftarrow \mathbf{\Theta}^{a_u}$ and $\mathbf{\Theta}^{c_j^-} \leftarrow \mathbf{\Theta}^{c_j}$. Initialize replay buffer $\mathcal{B}$ of size $\mathbf{B}$ and the mini-batch simpler $\mathcal{D}$ of size $\mathbf{D}$. Set total training step $n_t = 0$;
for $te = [1, te_{max}]$ **do**
  Reset time step $n = 0$, UAVs' locations to $\mathbf{q}_u[n]$ and queues to $\mathcal{Q}_g[n] = 0$, then the current state $s[n] = \{\mathbf{q}_u[n], \mathcal{Q}_g[n]\}$ is generated;
  repeat
    Perturb each exploratory actor via $\mathbf{\Theta}^{p_u} \leftarrow \mathbf{\Theta}^{a_u} + \mathbf{\Theta}^{p}$;
    Each UAV observes $s[n]$ and outputs $a_u[n] = \pi_u\left(s[n]|\mathbf{\Theta}^{p_u}\right) + \mathbf{N}$, then the joint action $\mathbf{a}[n] = [a_u[n]]_{u\in\mathcal{U}}$ is formulated;
    Execuate the joint action $\mathbf{a}[n]$, observe the next state $s[n+1]$ and receive the immediate common reward $r[n]$;
    if $|\mathcal{B}| < \mathbf{B}$ **then**
      Archive experience $\langle s[n], \mathbf{a}[n], s[n+1], r[n]\rangle$ into $\mathcal{B}$;
    else
      Replace the earliest stored experiences in $\mathcal{B}$ with the new transition $\langle s[n], \mathbf{a}[n], s[n+1], r[n]\rangle$;
    if $|\mathcal{B}| \geq \mathbf{D}$ **then**
      Randomly sample a mini-batch of size $\mathbf{D}$ from $\mathcal{B}$ into $\mathcal{D}$, i.e., $(s_m, \mathbf{a}_m, s_{m+1}, r_m) \in \mathcal{D} \sim \mathcal{B}$;
      for $u \in \mathcal{U}$ **do**
        The shadow actor outputs $\pi_u\left(s_{m+1}|\mathbf{\Theta}^{a_u^-}\right) + \mathbf{N}^-$ to calculate the target Q value;
        Update the dual online critics' trainable parameters $\mathbf{\Theta}^{c_j}$ by batch gradient descent on MSE loss $\ell\left(\mathbf{\Theta}^{c_j}\right)$ in (17);
        Increment the total training step $n_t \leftarrow n_t + 1$;
        if $n_t \% N_s == 0$ **then**
          for $u \in \mathcal{U}$ **do**
            The online actor generates $\pi_u\left(s_m|\mathbf{\Theta}^{a_u}\right)$;
            Update all the online actors' tunable parameters $\mathbf{\Theta}^{a_u}$ by batch gradient ascent by the chain rule as per (19);
            Update shadow networks $\mathbf{\Theta}^{a_u^-} \leftarrow \tau_s \mathbf{\Theta}^{a_u} + (1-\tau_s)\mathbf{\Theta}^{a_u^-}$ and $\mathbf{\Theta}^{c_j^-} \leftarrow \tau_s \mathbf{\Theta}^{c_j} + (1-\tau_s)\mathbf{\Theta}^{c_j^-}$;
      Trigger time step incrementation $n \leftarrow n + 1$;
  until $\|\mathbf{q}_u - \mathbf{q}_{u'\in\{\mathcal{U}\setminus u\}}\| < D, \exists u, \exists u' \;||\; \mathbf{q}_u(n) \notin \flat, \exists u \;||\; n = N_{\max}$;

IV. Numerical Results

Settings for system parameters and the learning hyperparameters are in line with Table I, while the terrestrial UEs' locations and the initial UAV coordinates are particularized in Fig. 1(c). Note that Linear transformation is adopted to serve as the aggregation function f_M in (16), where the NN of each module consists of three dense layers of sizes 64, 32 and 16. Then, following their specific modular network, the NN of perturbed/online/shadow actor or online/shadow critic comprises three dense layers with 512, 256 and 128 neurons, respectively. To highlight the impacts of various techniques applied to enable the proposed DSPAC-MN solution and compare the training performance, the following representative baselines are considered: 1) *MADDPG*: an extension of DDPG to handle multi-agent scenarios [10]; 2) *MATD3*: an extension of TD3 to reduce overestimation of MADDPG [11]; 3) *DSPAC-MN-NM*: DSPAC-MN without modular networks; 4) *DSPAC-MN-NR*: Regularization-less DSPAC-MN; 5) *DSPAC-MN-NL*: DSPAC-MN without learning rate scheduling; 6) *DSPAC-MN-B32*:

Table I: Setups for System Parameters and Hyperparameters of the Learning Process

Parameters	Values	Parameters	Values	Parameters	Values
Number of terrestrial UEs $\lvert\mathcal{G}\rvert = G$	6	Number of UAVs $\lvert\mathcal{U}\rvert = U$	2	Replay buffer $\mathcal{B}$'s capacity $\mathbf{B}$	10^5
Length of time slot δ_t	0.5 s	Queue capacity A_g^*	5×10^7 bits	Mini-batch sampler $\mathcal{D}$'s size $\mathbf{D}$	256
Safety distance for avoiding collision D	8 m	UEs' computation intensity c_g	10^3 cycles/bit	Exploration noise $\mathbf{N}$	Normal $(0, 5)$
UEs' maximum CPU-cycle frequency f_g^*	0.5 GHz	UEs' maximum transmit power P_g^*	30 dBm	Exploration noise variance decaying rate	0.999/episode
UEs' transmission bandwidth B	20 GHz	AWGN variance σ^2	-90 dBm	Staggered policy renewal frequency N_S	2
Computation overhead c_O	2	UAVs' computation intensity $c_{\mathcal{U}}$	10^3 cycles/bit	Shadow policy tempering noise $\mathbf{N}^-$	Normal $(0, 1)$
UAVs' CPU-cycle frequency budget f_u^*	10 GHz	New task instances' variance A_g	2×10^6 bits	Outbound/collision penalty p_o/p_c	100; 100
Minimum/maximum UAV speed $\mathbf{v}^-/\mathbf{v}^+$	10 m/s; 50 m/s	UAV altitude z_u	200 m	Dropout rate for online actors/critics	0.2
UEs' effective capacitance coefficient γ_g	10^{-28}	UEs' non-CPU power cost E_g^*	0 Joule	Learning rates α_a/α_c	10^{-4}; 10^{-3}
UAVs' effective capacitance coefficient γ_u	10^{-28}	UAVs' non-CPU power cost E_u^*	0 Joule	Critic's/actors' Exponential learning rate scheduler factor	0.9999; 0.9999
Fuselage drag ratio ϱ_0/Rotor solidity ϱ_2	0.6; 0.05	Air density ϱ_1	1.225 kg/m^3	Discount factor γ; Parameter-wise noise variance σ_p^2	0.99; 0.1
Rotor disc area ϱ_3	0.503 m^2	Blade angular velocity ϱ_4	300 radians/s	Polyak averaging coefficient τ_S	10^{-6}
Rotor radius ϱ_5	0.4 m	Profile drag Coefficient ϱ_6	0.012	Maximum training episodes $te_{\max}$	5000
Incremental correction factor to induced power ϱ_7	0.1	UAV weight ϱ_8	20 Newton	Step threshold $N_{\max}$	50
Average rotor induced velocity v_0	4.03 m/s	Rotor blade tip speed v_{tip}	120 m/s	Direction-aware collision penalty triggering factor T_c	1
Relative pressure	1013.25 hPa	Carrier frequency of THz channel f_c	0.3 THz	Dimension of each module's output $\lceil M_i \rceil$	10
Speed of light C	3×10^8 m/s	Antenna gains G_t/G_r	20 dBi; 0 dBi	Number of modules inside each actor	$3\,(U+1)$
Relative humidity; Energy regulation factor $\Re$	0.5; 0.02	Relative temperature	296.15 °K	Number of modules inside the shared cirtic	$4\,(U+1+2U)$



(a) Curves vs training episode (b) Violin plot vs episode segments (c) Designed paths in the last episode (d) Average safe flight probability

Figure 1: Performance comparison on average energy efficiency, designed trajectory, and average safe flight probability

DSPAC-MN with batch size of 32; and 7) *DSPAC-MN-F200*: DSPAC-MN with policy renewal frequency of 200.

Fig. 1(a) and Fig. 1(b) compare average energy efficiency over the UAVs' flights. The unit of metric is bits/wJoule, where wJoule stands for weighted Joule due to the weighted sum energy consideration for magnitude fairness between computation/offloading and propulsion energy costs. One can see that the proposed DSPAC-MN outperforms other baselines significantly, which showcases the effectiveness and efficiency of the tailored building blocks, e.g., the perturbed actors, the shared critic and the modularized inputs, for solving the formulated optimization problem in (10). From Fig. 1(c), it is interpreted that the proposed DSPAC-MN solution can generate trajectories that are not only sufficiently separated from each other but also away from the borders of $\mathbb{k}$, violating no constraints in (10). Unfortunately, baselines DSPAC-MN-B32, MADDPG and MATD3 can not avoid UAVs from crushing onto borders, violating the mobility constraint (1b), while benchmarks DSPAC-MN-NL, DSPAC-MN-NR, DSPAC-MN-F200 and DSPAC-MN-NM design trajectories that end up colliding with each other, infringing the collision constraint (1d). Moreover, Fig. 1(d) presents the average safe flight probability over episode segments, where the safe flight criterion is violated if any UAV flies outbound of $\mathbb{k}$ or any pair of UAVs collide. It is observed that the proposed DSPAC-MN approach is the only candidate that can achieve 100% safe flight navigations, while other baselines will direct the UAVs to break the rules regularized by (10) with a higher chance.

V. Conclusion

In this paper, we have proposed a UAV-assisted MEC task offloading method on the THz band. To achieve better energy efficiency, a performance maximization problem on the average energy efficiency, regarding joint optimization on UAVs' mobilities, radio resources and computation capabilities, was formulated. Then, an AI-native DSPAC-MN solution was proposed after mapping the focused optimization goal into a Markov game, in a decentralized MADRL fashion. Numerical results demonstrated the effectiveness of the proposed DSPAC-MN solution, while performance comparison against MADDPG and MATD3, and partial realizations of DSPAC-MN, demonstrated the corresponding efficacy and superiority.

References

[1] X. Tang, H. Zhang, R. Zhang, D. Zhou, Y. Zhang, and Z. Han, "Robust trajectory and offloading for energy-efficient UAV edge computing in industrial internet of things," *IEEE Trans. Ind. Inform.*, 2023.

[2] H. Peng and X. Shen, "Multi-agent reinforcement learning based resource management in MEC-and UAV-assisted vehicular networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 131–141, 2020.

[3] Y. Li and A. H. Aghvami, "Radio resource management for cellular-connected UAV: A learning approach," *IEEE Trans. Commun.*, vol. 71, no. 5, pp. 2784–2800, 2023.

[4] O. S. Badarneh, F. El Bouanani, and F. Almehmadi, "A general framework for UAV-aided THz communications subject to generalized geometric loss," *IEEE Trans. Veh. Technol.*, 2023.

[5] Y. Ding, Y. Feng, W. Lu, S. Zheng, N. Zhao, L. Meng, A. Nallanathan, and X. Yang, "Online edge learning offloading and resource management for UAV-assisted MEC secure communications," *IEEE J. Sel. Top. Signal Process.*, vol. 17, no. 1, pp. 54–65, 2022.

[6] Z. Yang, S. Bi, and Y.-J. A. Zhang, "Dynamic offloading and trajectory control for UAV-enabled mobile edge computing system with energy harvesting devices," *IEEE Trans. Wireless Commun.*, vol. 21, no. 12, pp. 10 515–10 528, 2022.

[7] Y. Xu, T. Zhang, D. Yang, Y. Liu, and M. Tao, "Joint resource and trajectory optimization for security in UAV-assisted MEC systems," *IEEE Trans. Commun.*, vol. 69, no. 1, pp. 573–588, 2020.

[8] 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, 2019.

[9] J. Ye, S. Dang, G. Ma, O. Amin, B. Shihada, and M.-S. Alouini, "On outage performance of terahertz wireless communication systems," *IEEE Trans. Commun.*, vol. 70, no. 1, pp. 649–663, 2021.

[10] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Adv. Neural Inf. Process.*, vol. 30, 2017.

[11] J. Ackermann, V. Gabler, T. Osa, and M. Sugiyama, "Reducing overestimation bias in multi-agent domains using double centralized critics," *arXiv preprint arXiv:1910.01465*, 2019.