Task Offloading and Position Optimization for Large Scale Unmanned Aerial Vehicle Networks: A Mean Field Learning Approach

Huixian Gu, *Graduate Student Member, IEEE*, Liqiang Zhao, *Member, IEEE*, Kai Liang, *Member, IEEE*, Gan Zheng, *Fellow, IEEE*, Kai-Kit Wong, *Fellow, IEEE* and Chan-Byoung Chae, *Fellow, IEEE*

***Abstract*—Unmanned aerial vehicle (UAV) networks have emerged as promising enablers in sixth generation (6G) communication system because they can support delay-sensitive and energy-constrained applications. However, the limited resources of UAVs and the high computational complexity of traditional methods complicate task offloading and position optimization. At scale, the task offloading and position optimization decisions yield non-stationary interactions among many agents, while standard multi-agent deep reinforcement learning (MADRL) suffers from poor scalability as the joint action space grows exponentially with the number of UAVs. We formulate joint task offloading and 2D position control as a Markov game that minimizes a weighted energy-delay cost per UAV under practical flight constraints (finite horizontal range, collision avoidance, and an elevation-angle limit) and resource constraints. We then develop a mean-field actor-critic (MFAC) framework that aggregates neighbors' influence into a mean action and conditions both the actor and the critic on local observations and the mean action. By approximating the interactions among a large number of agents through aggregating the influence of others into a mean action representation, the input dimensionality of the critic part is reduced from $M+KP$ to $M+2P$, yielding an approximately K-fold reduction and becoming independent of the agent population size compared to traditional MADRL methods. Numerical results demonstrate that our proposed algorithm can achieve an 80% reduction in the number of episodes, a 70% reduction in training time, a 38% reduction in energy consumption and a 28% reduction in task delay compared to state-of-the-art approaches, particularly under large-scale UAV deployment scenarios.**

***Index Terms*—UAV network, task offloading, position optimization, Markov game, mean field approximation, multi-agent deep reinforcement learning.**

This work was supported in part by in part by the Key Research and Development Program of Shaanxi (2024GX-YBXM-019), State Grid Jiangxi Electric Power Co., Ltd. Technology Project (5218G024000K), National Natural Science Foundation of China under Grant 62401421. The work of G. Zheng is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/X04047X/2 and EP/Y037243/1 for TITAN Telecoms Hub. The work of K. K. Wong is supported by the Engineering and Physical Sciences Research Council (EPSRC) under grant EP/W026813/1. The work of C.-B. Chae was supported by NRF/IITP (2024-00428780, 2021-0-00486) grants funded by the Korean government. *(Corresponding authors: Liqiang Zhao and Kai Liang)*

Huixian Gu (e-mail: kyyghx@aliyun.com) is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, 710071, China.

Liqiang Zhao (e-mail: lqzhao@mail.xidian.edu.cn) is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China, and also with the Guangzhou Institute of Technology, Xidian University, Guangzhou, 510100, China.

Kai Liang (e-mail: kliang@xidian.edu.cn) is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, 710071, China.

Gan Zheng (e-mail: gan.zheng@warwick.ac.uk) is with the School of Engineering, University of Warwick, Coventry, CV4 7AL, UK.

K. K. Wong (e-mail: kai-kit.wong@ucl.ac.uk) is affiliated with the Department of Electronic and Electrical Engineering, University College London, Torrington Place, WC1E 7JE, United Kingdom and he is also affiliated with Yonsei Frontier Lab, Yonsei University, Seoul, Korea.

Chan-Byoung Chae (e-mail: cbchae@yonsei.ac.kr) is with the School of Integrated Technology, Yonsei University, Seoul 03722, South Korea.

I. Introduction

THE widespread adoption of unmanned aerial vehicles (UAVs) in non-terrestrial networks (NTNs) has led to the rapid emergence of a wide range of new tasks, such as equipment inspection and crowd monitoring [1]. Market research indicates that the UAV industry will expand from USD 10.727 billion in 2025 to USD 19.083 billion by 2030, representing a compound annual growth rate (CAGR) of 12.21%. Meanwhile, large-scale deployment of UAVs enables automated sensing and control, which in turn generates a substantial volume of data traffic and processing demands [2]. These demands must be processed in a timely and efficient manner to ensure system responsiveness [3]. However, relying solely on the limited onboard resources of UAVs is often insufficient to fulfill these demands. To overcome this bottleneck, computation-intensive tasks can be offloaded to ground base stations (GBSs) equipped with edge servers or to cloud servers for more efficient execution [4], [5]. In this context, UAV networks integrated with multi-access edge computing (MEC) has emerged as a promising architectural paradigm that bridges the gap between high computational demands and the limited onboard capabilities of UAVs, enabling scalable, low-latency, and distributed computing for intelligent and delay-sensitive applications [6].

However, making effective task offloading and position optimization decisions is nontrivial, as it requires carefully balancing processing delay, energy consumption, and the availability of network resources [7]. This challenge is further exacerbated by the UAV mobility and the fluctuating quality of wireless links, which complicate real-time resource coordination. In [8], the authors addressed task offloading in a post-disaster UAV network and proposed a convex optimization-based algorithm for efficient MEC resource allocation, aiming to maximize system utility under resource constraints. In [9], the authors addressed task offloading and resource allocation in UAVs network for large-scale IoT networks. They applied a penalty successive convex approximation (PSCA) method with first-order Taylor expansion to solve the non-convex subproblem, improving solution quality under complex coupling constraints. In [10], the authors investigated task offloading and resource pricing in UAV networks using game-theoretic approaches, employing a Stackelberg game to jointly optimize server and user utilities. Song *et al.* [11] investigated a LEO-UAV cooperative edge computing framework, where multi-user devices (MUDs) offloaded tasks to resource-constrained

UAVs and LEO satellites. The task offloading problem was modeled as a potential game (LUTO-Game) considering UAV energy transfer and satellite coverage constraints. To jointly optimize UAV deployment and task offloading, the authors in [12] proposed a two-layer DECO algorithm, where differential evolution is applied to search for optimal 3D positions, elevation angles, and transmission power of UAVs, and convex optimization was employed to solve resource allocation and offloading decisions under complex constraints. However, traditional optimization methods, such as convex optimization and game theory, often struggle to handle the dynamic and uncertain nature of UAV task offloading scenarios [13], [14].

Recently, deep reinforcement learning (DRL) provides effective solutions for task offloading and position optimization, particularly in highly dynamic and unpredictable environments [15]. For example, the authors in [16] examined task offloading within a multi-UAV cooperative MEC framework, with a focus on UAV trajectory planning, offloading decisions, and the allocation of computational and communication resources, and developed a latent space-based DRL algorithm aimed at maximizing overall system performance. The authors in [17] introduced a UAV-assisted task offloading scheme, where an improved particle swarm optimization was applied to determine offloading decisions, while UAV trajectories were optimized using DRL. The authors in [18] proposed a D-DQN-based framework that combines convex optimization to jointly optimize UAV trajectory and task offloading in a UAV-MEC system. The authors in [19] proposed a UAV-enabled cloud-edge cooperative scheduling system that leverages deep deterministic policy gradient (DDPG) to jointly optimize user scheduling, UAV trajectory, and task offloading ratios, effectively minimizing processing delay while ensuring fairness in 5G URLLC scenarios with energy constraints and discrete variables. Their approach improves task execution efficiency and UAV energy savings, outperforming DQN and matching TSP-based solutions in key performance metrics. The centralized execution paradigm and the scale of large networks pose major challenges for DRL in achieving efficient decision-making [20].

To overcome the limitations of DRL, multi-agent DRL (MADRL) has emerged as a promising method, enabling decentralized decision-making and improved scalability by allowing agents to learn and act based on local observations while coordinating through shared environments. In [21], the authors formulated a joint optimization problem for UAV trajectory planning, task offloading, and bandwidth allocation in a UAV-assisted MEC network, aiming to maximize long-term energy efficiency. The problem was modeled as a Markov game (MG), and a multi-agent DRL (MADRL) algorithm was proposed to solve the problem under system dynamics and uncertainty. In [22], the authors formulated energy minimization in MEC-enabled air-ground networks as a multi-agent Markov decision process and proposed a multi-agent proximal policy optimization (MAPPO) algorithm to jointly optimize UAV trajectories, resource allocation, and queue-aware task offloading. The authors in [23] proposed a multi-agent actor-critic (MAAC) algorithm to jointly optimize trajectory planning and task offloading, the approach leverages causal reasoning to guide inter-UAV communication and employs a generalized Bellman operator to learn a unified policy representation across diverse objective preferences. In [24], the authors addressed the fair computation offloading problem in UAV-assisted MEC networks with heterogeneous task types and UAV capabilities. To adapt to dynamic user demands, an optimization-embedding MADRL algorithm was developed, where each UAV learns its trajectory via MADRL and computes offloading decisions by solving a mixed-integer nonlinear program. In [25], a Nash Q-learning-based algorithm was proposed to address task prioritization in MEC, enabling adaptive task offloading by integrating Nash equilibrium principles.

However, MADRL approaches and the Nash Q-learning algorithm suffer from high computational complexity when solving MGs as the number of agents increases [5]. Mean field theory has been introduced to MGs to simplify interaction among agents and thus reduce the computational complexity [26]. In [27], a scalable resource allocation scheme for UAV-assisted NOMA V2X networks was proposed by integrating mean-field theory with MARL, enabling distributed learning and reducing interaction overhead in large-scale agents. In [28], the authors addressed the scalability and learning inefficiencies of existing joint trajectory control and task offloading (JTCTO) algorithms in UAV-assisted MEC. They proposed a decentralized mean field-based multi-agent actor-critic (MFMAAC) algorithm, which combines policy transfer to accelerate learning and a mean field-based actor-critic approach to handle large-scale UAV scenarios. The authors in [29] proposed a MF deep Q network (MFDQN) framework for resource allocation in ultra-dense UAV networks by optimizing its trajectory, user association, and power control.

Compared with MAAC [23] and value-based mean-field DQN (MFDQN) [29], our MFAC differs in both modeling and algorithmic design. Specifically, large-scale UAV networks make standard MADRL brittle, centralized critics scale the input dimensionality as $M + KP$ and the joint action space grows exponentially, incurring high training cost and instability as K increases. Our key idea is to aggregate neighbors' influence via a mean-field action and condition *both* actor and critic on $(o_k, a_k, \bar{a}_k)$. This reduces the critic input dimensionality from $M+KP$ (centralized critics) to $M+2P$ without changing network depth/width (see Sec. III-D), thereby eliminating the linear dependence of the critic input on the population size K and improving stability in non-stationary environments. Unlike MFDQN, which injects mean-field information into value-based updates and is more sensitive to moving target policies, our policy-gradient updates with target networks and replay yield more stable convergence in large-scale settings. In this paper, we formulate a joint task offloading and UAV position optimization problem in large scale UAV networks. We present a mean field MADRL to solve it. The key contributions of this work are outlined below:

1) We formulate an MG for joint UAV task offloading and 2D position optimization that considers practical constraints, such as finite flight region, collision avoidance, and an elevation-angle limit, together with a weighted energy-delay objective per UAV.

2) We develop an MFAC framework that conditions both the actor and the critic on the agent's local observations and a mean-field action of neighbors. Unlike MAAC (centralized critic) and value-based MFDQN, our actor-critic design yields a per-agent critic input of size $M+2P$ (vs. $M+KP$ in MAAC). Compared with MFDQN, the policy-gradient updates with target networks and replay further stabilize learning under non-stationarity. We provide a layer-wise complexity analysis showing that this removes the linear dependence on the agent population at the critic input, yielding an approximately K-fold reduction in input dimensionality without changing the network depth/width.
3) Numerical results demonstrate that our proposed algorithm can achieve a reduction of 80% and 60% in the number of episodes, a reduction of 70% and 10% in the training time compared with MFDQN and MAAC algorithms. Furthermore, our proposed algorithm can achieve a reduction of 16%, 22% and 35% in energy consumption and a reduction of 5%, 26% and 28% in task delay compared with MFDQN, MAAC and FUPC.

The remainder of this paper is organized as follows. Section II describes the system model and formulates the optimization problem. In Section III, we propose the problem-solving framework of the optimization problem. Section IV presents extensive experimental results to validate the proposed approach, and Section V concludes the paper.

II. System Model and Problem Formulation

As shown in Fig. 1, we consider a UAV task offloading scenario involving applications such as aerial sensing and smart logistics. These applications can be offloaded to GBSs, which are equipped with edge servers for computing, or to a cloud server (CS) for efficient processing. The task generated by UAV k is characterized by $I_k = \{S_k, C_k, d_k^{max}\}$, where S_k is the task size, C_k is the required number of CPU cycles to execute the task, and d_k^{max} is the maximum tolerable delay of task k. The proposed system consists of three layers: the UAV layer is responsible for generating computational tasks, the edge layer is responsible for providing computational resources for resource-limited UAVs, and the cloud layer is also responsible for supporting UAVs with additional computational power. The key difference between the edge and cloud servers lies in their characteristics: edge servers can provide low-latency computation by offering resources in close proximity to UAVs, whereas cloud servers offer significantly higher computational capacity but are located farther away, resulting in higher transmission delays. The set of GBSs and UAVs are denoted by $\mathcal{N} = \{1, 2, \cdots, N\}$, $\mathcal{K} = \{1, 2, \cdots, K\}$, respectively. Similar to [30], [31], to simplify the trajectory design, the UAVs are assumed to operate at a constant altitude h_k, the two-dimensional coordinates of GBSs and UAVs are represented by $\mathbf{c}_n = (x_n, y_n)$ and $\tilde{\mathbf{c}}_k = (x_k, y_k)$, respectively. This article ignores the heights of GBS and antennas and assumes that the location of GBSs are fixed, which is in line with [32], [33]. The main notations used in this paper are summarized in Table I. This paper adopts a binary offloading strategy. While [34] investigates partial task offloading with broad applicability, many real-world workloads are inherently indivisible. Thus, studying binary offloading remains practically important. Furthermore, because the size of the computed results is typically far smaller than that of the offloaded tasks, the downlink latency can be neglected [35].

TABLE I. The main notations used in the paper

Notation	Explanation
$\mathcal{N}$	The set of GBSs
$\mathcal{K}$	The set of UAVs
h_k	The fly hight of UAVs
$\mathbf{c}_n$	The two-dimensional coordination of GBS
$\tilde{\mathbf{c}}_k$	The two-dimensional coordination of UAV
S_k	The task size generated by UAV k
C_k	The CPU cycles required to process one bit task
d_k^{max}	The maximum tolerable delay of task generated by UAV k
L_{max}	The limited flight distance of UAV
D_{max}	The minimum distance between any two UAVs
θ_k	The maximum elevation angle of UAV k
$d_{k,n}$	The distance between UAV k and GBS n
$g_{k,n}$	The channel gain between UAV k and GBS n
β_0	The channel gain when the reference distance is 1m
α_1, α_2	The path loss exponent
$\gamma_{k,n}$	The SINR between UAV k and GBS n
$\sigma_{k,n}^2$	The Gaussian noise power between UAV k and GBS n
N_0	The noise power spectrum density
p_k	The transmission power of UAV k
$r_{k,n}$	The transmission rate between UAV k and GBS n
$r_{k,c}$	The transmission rate of UAV k and CS c
$\mu_{k,l}$	The task generated by UAV k processed locally
$\mu_{k,n}$	The task generated by UAV k processed at the GBS n
$\mu_{k,c}$	The task generated by UAV k processed at the CS

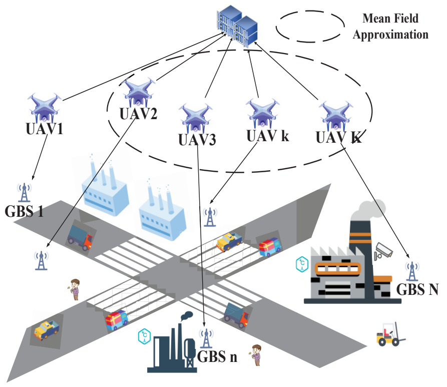


Fig. 1. System model.

A. UAV's Requirements

Due to the UAVs fight at the fixed hight, the 2D coordinate of UAV is denoted as $\tilde{\mathbf{c}}_k = (x_k, y_k)$. UAVs' range of motion is denoted as $0 \le x_k \le x_{max}$ and $0 \le y_k \le y_{max}$.

Assume that UAV k travels a horizontal distance of $l_k(t)$ in the direction specified by the angle $\theta_k(t) \in [0, 2\pi]$. Then, the coordinations of UAV k at time slot $t+1$ are given by

$x_k(t+1) = x_k(t) + l_k(t)\cos\theta_k(t)$ and $y_k(t+1) = y_k(t) + l_k(t)\sin\theta_k(t)$.

Due to the limited horizontal-flight speed, the limited flight distance of UAVs can be given by

$$l_k(t) = ||\tilde{\mathbf{c}}_k(t+1) - \tilde{\mathbf{c}}_k(t)|| \leq L_{max}. \quad (1)$$

On the other hand, due to the existing of multiple UAVs, the collision should be avoided. Then, the collision constraint of arbitrary two UAVs is represented as follows

$$||\tilde{\mathbf{c}}_k - \tilde{\mathbf{c}}_j|| \geq D_{max}, \forall k, j, k \neq j. \quad (2)$$

Due to the impact of maximum elevation angle, if UAV k wants to offload the task to GBS n, it must satisfy the constraint of the maximum elevation angle, i.e.,

$$h_k \geq \sqrt{||\tilde{\mathbf{c}}_k - \mathbf{c}_n||^2}\cot(\psi_k). \quad (3)$$

Following [36], [37], we adopt a simplified flight-energy model in which the energy per slot depends solely on the velocity vector. Specifically, $e_k^{fly} = \phi\left(\frac{l_k}{\tau}\right)^2$, with $\phi = 0.5M_k$, where M_k denotes the payload of UAV k, and τ denotes the fixed flight time of UAVs. We assume that each UAV offloads tasks to a GBS while hovering in the air after completing the position optimization. If UAVs offload tasks while flying, the communication links between UAVs and GBSs would change dynamically, which may lead to interruptions in task offloading or failure to receiving the results [21].

B. The Communication Model

To avoid inter-UAV interference, orthogonal frequency division multiplexing (OFDM) is adopted. Hence, the interference among UAVs is effectively eliminated. The distance between GBS n and UAV k is represented by $d_{k,n} = ||\mathbf{c}_n - \tilde{\mathbf{c}}_k||$, where $||\cdot||$ denotes the Euclidean norm. Given that UAVs fly at relatively high altitudes, the LoS channel is more predominant compared with other channels, such as shadowing and small-scale fading [38]. Hence, the LoS channel gain between GBS n and UAV k is modeled using the free-space path loss model [39], which can be represented as $g_{k,n} = \beta_0 d_{k,n}^{-\alpha_1}$, where β_0 denotes the channel gain measured at 1 meter, and α_1 is the path loss exponent.

Thus, the SINR from GBS n to UAV k can be expressed as $\gamma_{k,n} = \frac{p_k g_{k,n}}{\sigma_{k,n}^2}$, where $\sigma_{k,n}^2 = N_0 B_{k,n}$. N_0 is the noise power spectrum density. $B_{k,n} = \frac{B_n}{K}$ is the pre-allocated bandwidth between GBS n and UAV k, where B_n is the total bandwidth of GBS n. p_k denotes the transmit power of UAV k. Therefore, the transmission rate between UAV k and GBS n is

$$r_{k,n} = \frac{B_n}{K}\log_2(1+\gamma_{k,n}). \quad (4)$$

Similarly, the transmission rate between UAV k and CS $r_{k,c}$ is denoted by

$$r_{k,c} = \frac{B_c}{K}\log_2(1+\gamma_{k,c}). \quad (5)$$

where $\gamma_{k,c} = \frac{p_k g_{k,c}}{\sigma_{k,c}^2}$, $g_{k,c} = \beta_0 d_{k,c}^{-\alpha_2}$, $d_{k,c} = ||\tilde{\mathbf{c}}_k - \mathbf{c}_{cs}||$, $\frac{B_c}{K}$ is the pre-allocated bandwidth between UAV k and CS and the bandwidth of CS is $\leq B_c$, $\mathbf{c}_{cs} = [x_c, y_c, 0]$ is the fixed location of CS.

The above LoS/free-space and orthogonal-access assumptions simplify analysis and may bias absolute performance estimates in real deployments. In particular: (i) ignoring shadowing, small-scale fading, and Doppler often overestimates SINR and reduces temporal variance, leading to optimistic (lower) delay and energy consumption; (ii) neglecting inter-cell and inter-UAV interference may overstate the achievable throughput, especially in dense deployments. A robustness extension with probabilistic LoS, log-normal shadowing/Rician fading, and explicit interference modeling is left for future work.

C. Computing Model and Energy Model

In the following, we analyze the latency and energy consumption of our proposed system. After generating the task I_k, UAV k determines whether to handle the task onboard or offload it to edge or cloud computing resources. We denote task processing at UAV k, the GBS n, and CS as $\mu_{k,l} \in \{0,1\}$, $\mu_{k,n} \in \{0,1\}$ and $\mu_{k,c} \in \{0,1\}$, respectively.

1) UAV Computing: The UAV's processing latency and energy consumption include the execution delay $d_{k,l}^{ex} = \mu_{k,l}\frac{C_k}{f_k}$ and the energy consumption $e_{k,l}^{ex} = \kappa\mu_{k,l}C_k f_k^{v}$, where f_k is the computing resource of UAV k and κ is the effective switched capacitance, with $2.5 < v < 3$ [40].

2) GBS Computing: If the task I_k needs to offload to GBS for processing, the processing delay and energy consumption include the UAV's transmission delay $d_{k,n}^{tr} = \mu_{k,n}\frac{S_n}{r_{k,n}}$, the UAV's transmission energy consumption $e_{k,n}^{tr} = p_k d_{k,n}^{tr}$ and the GBS's execution delay $d_{k,n}^{ex} = \mu_{k,n}\frac{C_n}{f_{n,k}}$, where $f_{n,k}$ is the computing resource of GBS n pre-allocated to UAV k.

3) Cloud Computing: Due to the powerful computing capacity of cloud servers, we ignore the computational latency at the CS. In addition, the results of the computed task are smaller than the task, hence, we ignore the downlink delay for sending the results to the UAV. The delay and energy consumption of cloud computing consist of the UAV's transmission delay $d_{k,c}^{tr} = \mu_{k,c}\frac{S_n}{r_{k,c}}$ and the UAV's transmission energy consumption $e_{k,c}^{tr} = p_k d_{k,c}^{tr}$.

D. Problem Formulation

Since task I_k only can be processed locally, at GBS n or at CS, the delay of UAV k can be expressed as

$$d_k = d_{k,l}^{ex} + \sum_{n=1}^{N}(d_{k,n}^{ex} + d_{k,n}^{tr}) + d_{k,c}^{tr} + \tau_{\{l_k \neq 0\}}, \quad (6)$$

where $\tau_{\{l_k \neq 0\}}$ indicates that if UAV k decides to change to a now position to execute the task, it incurs an additional flight time τ. Moreover, after completing the tasks, the energy consumption of UAV k can be represented as follows

$$e_k = e_{k,l}^{ex} + \sum_{n=1}^{N} e_{k,n}^{tr} + e_{k,c}^{tr} + e_k^{fly}. \quad (7)$$

Similar to [41], we adopt the weighted sum of energy consumption and task latency as UAV k's utility, that is $U_k = w_1 d_k + w_2 e_k$, where w_1 and w_2 denotes the relative importance oof energy consumption and task delay. $w_1 < w_2$ indicates the energy-saving scenarios, $w_1 > w_2$ denotes the delay sensitive scenarios. Due to the inconsistency in the value ranges of delay and energy consumption, we normalize the utility as follows to evaluate the impact of the weighting factors.

$$U_k^{nor} = w_1 \frac{d_k}{d_k^{max}} + w_2 \frac{e_k}{E_k}, \tag{8}$$

where d_k^{max} and E_k are the maximum processing delay and energy of UAV k. Then, each UAV needs to minimize U_k^{nor} by optimizing the UAV k's position c_k and task offloading decision $\mu_{k,l}$, $\mu_{k,n}$ and $\mu_{k,c}$. We use $\pi_k = \{c_k, \mu_{k,l}, \mu_{k,n}, \mu_{k,c}\}$ to denote UAV k's policy. Specifically, the optimization problem can be formulated as follows

$$\begin{aligned}
&\min_{\pi_k} U_k^{nor}, \forall k \in \mathcal{K}, \\
\text{s.t.} \quad & C1: \mu_{k,l}, \mu_{k,n}, \mu_{k,c} \in \{0,1\}, \\
& C2: \mu_{k,l} + \sum_{n=1}^{N} \mu_{k,n} + \mu_{k,c} = 1, \\
& C3: \sum_{n=1}^{N} \mu_{k,n} \le 1, \\
& C4: d_k \le d_k^{max}, \forall k, \\
& C5: \sum_{t=1}^{T} e_k \le E_k, \\
& C6: 0 \le x_k(t) \le x_{max}, 0 \le y_k(t) \le y_{max}, \forall k \in \mathcal{K}, \\
& C7: (1), (2), (3),
\end{aligned} \tag{9}$$

In the formulated problem, each UAV needs to find a policy π_k that maximizes the long-term cumulative expected reward. We use $\pi = \{\pi_1, \pi_2, \cdots, \pi_K\}$ to represent the joint strategy of all UAVs. Constraints $C1$, $C2$, and $C3$ denote the offloading constraints. Constraint $C4$ requires that the completion time of task I_k can not exceed its corresponding threshold. Constraint $C5$ represents that the energy consumption during flight can not exceed the battery energy. Constraints $C6$ and $C7$ denote the position and flight limitations of UAVs.

However, the above-stated problem has the following difficulties: 1) The decisions of each UAV not only depend on its own decisions, but also on those of other UAVs; 2) To compute the optimal solution for the previously mentioned problem, UAV k needs to maximize its long-term cumulative expected reward. However, optimizing the cumulative reward for UAV k relies on the joint actions of all UAVs. Therefore, the above statement is in line with Nash equilibrium (NE). In an NE, each UAV's optimal strategy is its best response to the decisions of all other UAVs.

III. The Mean Field Actor Critic (MFAC) Algorithm for Task Offloading and UAV Position Optimization

In this section, the above optimization problem is first formulated as an MG in Section III-A. Then, a mean field approximation approach is utilized to simplify the interaction of the MG in Section III-B. Finally, considering the complex environment of MEC environment, we propose a DRL-enabled algorithm to solve the optimization problem in Section III-C.

A. MG Formulation

Since UAVs need to make task offloading and position decisions to minimize their own cost based on other UAV's current actions, we formulate the optimization problem as an MG as follows.

1) Definition of an MG: The K-player MG can be defined as $\Gamma = (\mathcal{S}, \{\mathcal{A}_k\}_{k=1}^K, \{r_k\}_{k=1}^K, p, \gamma)$, and each element can be defined as follows

- *State Space* $\mathcal{S}$: At time slot t, the state consists of the coordinations of UAVs, the remaining energy $e_k^{re}(t)$, the task size $S_k(t)$ and required CPU cycles $C_k(t)$ of task I_k. Then, the state can be written as $\mathbf{s}(t) = \{\mathbf{c}_k(t), e_k^{re}(t), S_k, C_k\}_{k \in \mathcal{K}}$.
- *Action Space* $\mathcal{A}_k$: $\mathcal{A}_k$ denotes the action space of UAV k, and the joint action of UAVs is represented by $\mathbf{a}(t) = \{a_1(t), \cdots, a_K(t)\}$. Specifically, the action space $a_k(t)$ of UAV k at time slot t is $a_k(t) = \{\mu_{k,l}(t), \mu_{k,n}, \mu_{k,c}(t), l_k(t), \theta_k(t)\}$. The feasible ranges are: $l_k(t) \in [0, L_{max}]$, $\theta_k(t) \in [0, 2\pi)$, and $\mu_{k,l}(t), \mu_{k,c}(t) \in 0, 1$. The flying angle at time t is discretized into $[0, \pi/6, \pi/3, \ldots, 2\pi)$, and the flying distance into $[0, L_{max}/6, L_{max}/3, \ldots, L_{max}]$. The one-hot encoding technique is used for the action space, enabling decision-making via deep neural networks.
- *Reward Function* r_k: After taking action $a_k(t)$ at state $s(t)$, the immediate reward received by UAV k can be defined as r_k. Since our optimization objective is to minimize the weighted cost of UAV k, we define $r_k(s(t), a_k(t)) = -\sum_{t=1}^{T} U_k^{nor}(t)$.
- *State transition probability* p: UAV k selects action $a_k(t)$ and receives a reward $r_k(t)$, and the state $s_k(t)$ transitions to state $s_k(t+1)$. Therefore, the probability transition $p: \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_K \to p(\mathcal{S})$.
- *Discount Factor* γ: $\gamma \in [0, 1)$ indicates the impact of future rewards on the current decision.

2) NE Strategy: For an MG, an NE is a joint strategy under which each UAV will not change its strategy if the other UAVs' strategy do not change. Therefore, the NE can be represented as $\pi = \{\pi_1, \pi_2, \cdots, \pi_K\}$, such that for UAV k

$$v_k(s, \pi_k^*, \pi_{-k}^*) \ge v_k(s, \pi_k, \pi_{-k}^*), \tag{10}$$

where $\pi_{-k}^* = (\pi_1, \cdots, \pi_{k-1}^*, \cdots, \pi_{k+1}, \cdots, \pi_K^*)$ denotes the optimal strategies of UAVs except UAV k, $v_k(s) = \mathbb{E}_\pi[\sum_t \gamma^t \mathbb{E}(r_n^t)]$ is the value function of UAV k.

Since NE serves as the theoretical foundation for the convergence of MARL algorithms, the authors in [42] proposed a

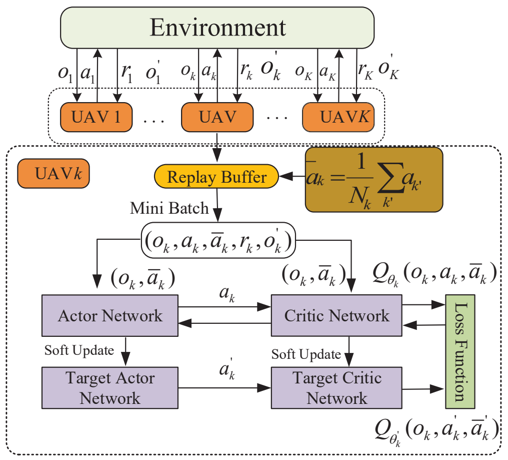


Fig. 2. The architecture of the proposed algorithm.

Nash Q-learning algorithm to identify the NE and proved its convergence. In this algorithm, the Q-function is updated as follows

$$Q_k^\pi(\mathbf{s},\mathbf{a}) \leftarrow Q_k^\pi(\mathbf{s},\mathbf{a}) + \alpha\left(R(\mathbf{s},\mathbf{a}) + \gamma v_k^{Nash}(\mathbf{s}') - Q_k^\pi(\mathbf{s},\mathbf{a})\right), \tag{11}$$

where $v_k^{Nash}(\mathbf{s}')$ is the equilibrium value for the agent k under state $\mathbf{s}'$.

B. A Mean-Field Learning Approach

In large-scale, partially observable UAV-MEC systems, computing a Nash equilibrium is intractable. We therefore approximate pairwise interactions by conditioning the critic on the agents observation o_k, its action a_k, and the average neighbor action $\bar{a}_k = \frac{1}{N^k}\sum_{k'\in\mathcal{N}(k)} a_{k'}$, and the joint-action critic admits the mean-field form (Appendix A, Theorem. 1)

$$Q(o_k, \mathbf{a}) \;\approx\; Q(o_k, a_k, \bar{a}_k), \tag{12}$$

which removes the linear dependence on the population size K at the critic input (from $M+KP$ to $M+2P$) while preserving decentralized execution. The mean-field Bellman update is then

$$Q(o_k^t, a_k^t, \bar{a}_k^t) \leftarrow (1-\eta)Q(\cdot) + \alpha\left[r_k^t + \gamma\, v^k(s')\right], \tag{13}$$

where $v^k(s') = \sum_{a_k} \pi_k(a_k|s', \bar{a}_k)\, \mathbb{E}_{\bar{a}_k(\mathbf{a}_{-k})}[Q(o_k, a_k, \bar{a}_k)]$. Full derivations are provided in Appendix A.

C. The Whole Algorithm

Due to the strong representational ability of neural network, we utilize neural networks to obtain an NE solution within the mean field learning framework. The framework of the proposed algorithm is shown in Fig. 2. To enhance decision-making performance, we adopt the actor-critic reinforcement learning framework, which builds upon traditional policy gradient methods. Furthermore, target networks and experience replay are integrated into the algorithm to reduce action-value overestimation and promote more stable training. The whole workflow of the proposed algorithm is described al follows: in the actor part, an action is output based on the input observation $o_k(t)$. In the critic part, the Q value is computed based on the input observation $o_k(t)$, the action of agent k and mean field action $\bar{a}_k(t)$.

1) Neural Network Initialization: UAV k trains its own neural network independently. In this paper, the actor-critic architecture is adopted to improve learning efficiency. The actor is responsible for determining the action, and the critic part responsible for calculating the value function.

2) Action Execution: The actor part consists of the actor network and the target actor network. At time slot t, the UAV k chooses action a_k^t according to its policy π_k, which is generated by the actor network based on the observation $o_k(t)$ and the mean field action $\bar{a}_k(t)$. The selected action determines UAV k's offloading strategy and movement strategy. Meanwhile, the target actor network is updated to maximize the long-term discounted cumulative reward, ensuring stable policy learning. Furthermore, a Boltzmann action selection strategy is adopted to select actions and can be denoted as

$$\pi_k^t(a_k|s,\bar{a}_k) = \frac{\exp(\beta Q_k^t(s,a_k,\bar{a}_k))}{\sum_{a_k'\in\mathcal{A}}\exp(\beta Q_k^t(s,a_k',\bar{a}_k))}, \tag{14}$$

where β is the temperature used to control the exploration rate.

3) Batch Data Collection: To enable the neural network to obtain good strategies, the local observations, actions, mean field action, and next states are recorded in batches for each UAV. In this manner, each UAV can train its learning model through minimizing its loss function.

4) Model Training: UAV k trains its individual learning model. Given the adoption of the actor-critic architecture, the loss function for the critic is

$$L(\theta_k) = E[(Y_k^t - Q_{\theta_k}(o_k^t, a_k^t, \bar{a}_k^t))^2], \tag{15}$$

where

$$Y_k^t = r_k^t + \sum o_k \pi_k^t(o_k|s',\bar{a}_k)\mathbb{E}_{\bar{a}_k(\mathbf{a}_{-k}\sim\pi_{-k}^t)}\left[Q_k^t(s',o_k,\bar{a}_k)\right]. \tag{16}$$

Furthermore, the gradient for the actor part is defined as

$$\nabla L_\pi(\omega_k) = E\left[\nabla_{\omega_k}\log\pi_{\omega_k}(o_k^t)Q_{\theta_k}(o_k^t, a_k^t, \bar{a}_k^t)\right]. \tag{17}$$

The whole algorithm is shown in Algorithm 1.

D. Complexity Analysis of the Mean Field Learning Approach

We compare the computational complexity of the multi-agent actor-critic (MAAC) algorithm and the mean field actor-critic (MFAC) algorithm by analyzing the structure of their actor and critic networks. Specifically, the analysis focuses on three components: the input layer, hidden layers, and the output layer. For both algorithms, the actor and critic networks are assumed to have the same L hidden layers with Z neurons in each layer.

Let M denote the local observation dimension of each agent, P denote the dimension of the action space, and K denote the total number of agents.

Algorithm 1 MFAC algorithm for Task Offloading and Position Optimization Problem

```
Require:
    The parameters: episodes $N_{epi}$, each episode length $T$, batch size $N_{batch}$, soft update $\tau$, and $\bar{a}_k$ for all $k \in \{1, \cdots, K\}$.
    thresholds $\varepsilon_A, \varepsilon_C$, patience $P$, warmup $U_{\min}$, streak $\leftarrow 0$, $u \leftarrow 0$.
Ensure:
    Optimal decisions $\mathcal{A}_t$, which include optimal task offloading decision and bandwidth allocation decision decision;
    Randomly initialize critic network $Q_\theta$ and actor network $\pi_\omega$ with parameters $\theta$ and $\omega$, respectively;
    Randomly initialize the target networks $Q^{'}_{\theta'}$ and network $\pi^{'}_{\omega'}$ with $\theta' \leftarrow \theta$ and $\omega' \leftarrow \omega$;
    Randomly initialize mean field action $\bar{a}_k$;
1:  for episode $0, 1, \cdots, N_{epi} - 1$ do
2:      for $t = 0, 1, \cdots, T - 1$ do
3:          Select an action based on Boltzmann policy $a_k(t) = \pi_\omega(o_k(t), \bar{a}_k^t)$
4:          Execute action $a_k(t)$ in the system, observe cost $r_k(t)$ and the next state $o_k(t+1)$, and compute mean field action $\bar{a}_k(t)$;
5:          Store transition $(o_k(t), a_k(t), \bar{a}_k(t), r_k(t), o_k(t+1))$ in replay buffer$R$;
6:          Sample a $N_{batch}$ from replay buffer $R$ ;
7:          Set $y_i = r_i + \gamma Q_{\theta_k'}(o_i(t), a_i(t), \bar{a}_i(t))$, for $i = 1, \cdots, N_{batch}$;
8:          Compute the critic:
            $L(\theta_k) = \frac{1}{N_{batch}} \sum_i (y_i - Q_{\theta_k}(o_i(t), a_i(t), \bar{a}_i(t))^2$;
9:          Compute the actor policy using the gradient $\nabla_\omega J$:
            $\nabla_{\omega_k} J(\omega_k) \approx \frac{1}{N_{batch}} \sum_i \log \pi_{\omega_k}(o_i^t) Q_{\theta_k}(o_i^t, a_i^t, \bar{a}_i^t)$
10:         $g_C \leftarrow \|\nabla_\theta L\|_2, \quad g_A \leftarrow \|\nabla_\omega J\|_2$
11:         if $g_C \le \varepsilon_C$ or $g_A \le \varepsilon_A$ then
12:             break
13:         end if
14:         Update target networks:
15:         $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$;
16:         $\omega' \leftarrow \tau\omega + (1 - \tau)\omega'$;
17:     end for
18: end for
19: return $\pi'_{\omega'}$
```

1) MAAC: For the actor network, each agent uses its local observation (dimension M) to output an action (dimension P). The network has L layers, and each hidden layer has Z neurons. The computational complexity is: $\mathcal{O}(MZ + (L-1)Z^2 + PZ)$.

For the critic network, the input combines the agent's observation and the actions of all K agents, with a total input size of $M + KP$. The computational complexity is $\mathcal{O}((M + KP)Z + (L-1)Z^2 + Z)$.

2) MFAC: The actor network in MFAC takes the local observation of dimension M and the mean field action of dimension P as input, its output is an action of dimension P. Consequently, its complexity is given by $\mathcal{O}((M + P)Z + (L-1)Z^2 + PZ)$.

TABLE II. Computational Complexity of the Algorithm

Algorithm	Actor	Critic
MAAC	$\mathcal{O}(MZ+(L-1)Z^2+PZ)$	$\mathcal{O}((M + KP)Z + (L - 1)Z^2 + Z)$
MFAC	$\mathcal{O}((M + P)Z + (L - 1)Z^2 + PZ)$	$\mathcal{O}((M + 2P)Z + (L - 1)Z^2 + Z)$

The critic network, however, differs significantly due to the adoption of mean field approximation. Instead of using the full joint action of all agents, the MFAC critic takes as input the local observation (M), the agent's own action (P), and the mean action of its neighbors (P). This reduces the input dimension from $M + KP$ to $M + 2P$. Therefore, the complexity of the critic network becomes $\mathcal{O}((M + 2P)Z + (L-1)Z^2 + Z)$.

3) Comparison and Discussion: The key difference lies in the critics input. In MAAC, the critic takes all agents actions, resulting in an input size of $\mathcal{O}(KP)$. In contrast, MFAC uses a mean action, reducing this to $\mathcal{O}(P)$. This greatly lowers the input size and computational cost, especially when the number of agents K is large. As a result, the input dimensionality of the critic part is reduced from $M+KP$ to $M+2P$, yielding an approximately K-fold reduction and becoming independent of the agent population size compared with traditional MADRL methods without changing the network structure. In summary, MAAC relies on a centralized critic whose input concatenates all agents actions, resulting in an $M+KP$ critic input per agent and degraded scalability. MFDQN injects mean-field information into a value-based learner, but remains sensitive to non-stationarity in multi-agent training. Our MFAC keeps the benefits of mean-field aggregation while leveraging policy-gradient updates and target networks; empirically this yields more stable convergence and linear wall-clock scaling in K. The complexity comparison is summarized in Table II.

IV. Performance Evaluation

This section presents the experimental evaluation using Tensorflow 2.2 and Python 3.8 to demonstrate the effectiveness of the proposed algorithm. The number of UAVs and GBSs are set to 20 and 3, respectively. Specifically, UAVs are randomly deployed in a square area of $1000m \times 1000m$, and the hight of the UAVs is fixed at $100m$.

In the MFAC algorithm, the simulation parameters include a maximum elevation angle of $\theta_k = 44.24°$, allocated bandwidths $B_k = 10MHz$, transmit powers $p_n = 0.2W$ and $p_k = 1W$, path loss exponents $\alpha_1 = \alpha_2 = 3$, noise power spectral density of $N_0 = -174dBm/Hz$, and a channel gain of $\beta_0 = -50dB$. The computational capability of a UAV is $F_k = 5Gcycles/s$, and its payload capacity $M_k = 15kg$. The computational capability of CS is $F_c = 15Gcycles/s$. Furthermore, the task size is uniformly distributed in [100, 200] kbits, and the required CPU cycles are uniformly distributed in [5, 15] G-cycles. We adopt an actor-critic with a mean field approximation of the joint action space. Specifically, the actor's input layer has 133 dimensions and consists of the observation o_k of agent k and the mean field action. The

TABLE III. Network Parameters

Parameter	Value
Maximum elevation angle θ_k	44.24°
Allocated bandwidth between IIoT and UAV B_k	10 MHz
Allocated bandwidth between UAV and cloud server B_{max}	30 MHz
Transmit power of ID p_n	0.2 W
Transmit power of UAV p_k	1 W
Path loss exponent α_1 and α_2	3
Noise power spectral density N_0	-174 dBm/Hz
Channel gain with 1 meter distance β_0	-50 dB
The hight of UAV h_k	100 m
The horizontal movement range of UAV x_k and y_k	100 m, 100 m
Computational capability of UAV F_k	5 G-cycles/s
Effective switched capacitance κ	10^{-27}
Payload of UAV M_k	15 kg
Computational capability of cloud server F_k	15 G-cycles/s

numbers of neurons in each hidden layer are 1024, 512 and 256. We use ReLU as the activation function. The output is a 128 dimensional discrete action. Similarly, the critic's input layer has 261 dimensions and consists of the observation o_k, the action a_k, and the mean field action $\bar{a}_k$. The numbers of neurons in the subsequent layers are 1024, 512, 256, and 128. The output of the critic network is state-action value $Q(o_k, a_k, \bar{a}_k)$. The training process is conducted over 1000 episodes, with each episode consisting of 50 steps. A replay buffer with a capacity of 1000 transitions is employed to store agent experiences for off-policy learning. During each update, a mini-batch of 128 samples is randomly drawn from the buffer for parameter optimization. The actor network is trained with a learning rate of $\alpha_a = 0.001$, while the critic network is trained with a learning rate of $\alpha_c = 0.002$, and the discount factor is $\gamma = 0.95$. We also adopt soft target updates for both actor and critic at every gradient step with $\tau = 0.002$. Furthermore, we employ a gradient-based stopping rule: after a warm-up of $U_{\min} = 50000$ updates, training is stopped once the ℓ_2 norms of the actor and critic gradients satisfy $g_A = \|\nabla_\omega J\|_2 < 10^{-3}$ and $g_C = \|\nabla_\theta L\|_2 < 5 \times 10^{-3}$. The parameter settings are summarized in Table III. Unless explicitly stated otherwise, all parameters not being swept in a figure are fixed to the values in Table III.

To evaluate the performance of **our proposed** algorithm, we conduct the following benchmark solutions.

- The **FUPC** scheme, unlike the proposed method, only optimizes task offloading under a fixed UAV position, in this scheme, we assume all the UAVs are uniformly distributed with the square [11].
- The **MAAC** scheme, similar to the proposed algorithm, involves agents making decisions based on the actions of others, with policy gradient methods adapted for discrete action spaces [23].
- The **MFDQN** scheme integrates MFT into DQN, allowing each agent to use the mean field approximation of its neighbors' actions instead of considering their individual actions [29].
- The **SWA**: First, with UAV positions fixed, each UAV chooses local/GBS/cloud execution by minimizing a SINR-based surrogate of the weighted energy-delay cost (rate estimated from the current SINR). Then, keeping the offloading decisions fixed, each UAV slightly adjusts its heading and step length subject to inter-UAV distance and elevation-angle constraints [43], [44].

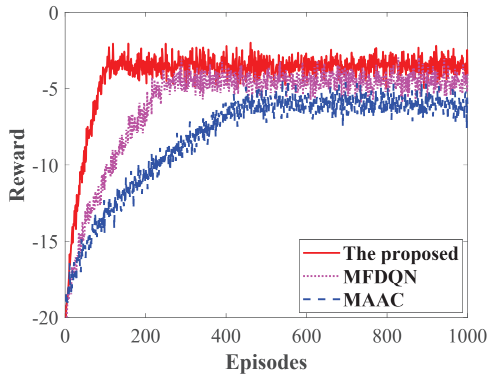


Fig. 3. The convergence performance.

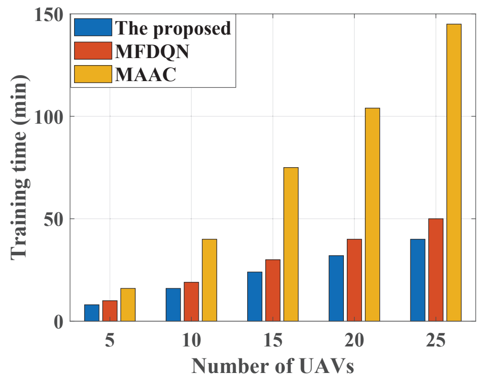
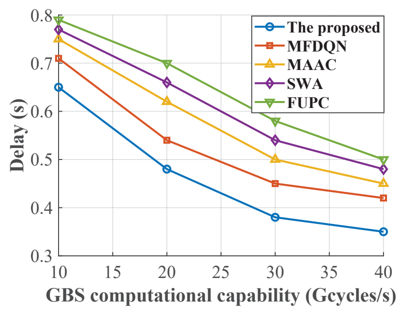
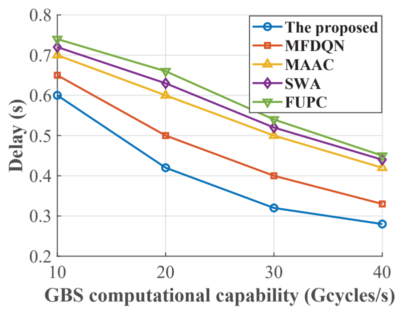


Fig. 4. Training time vs. number of UAVs.

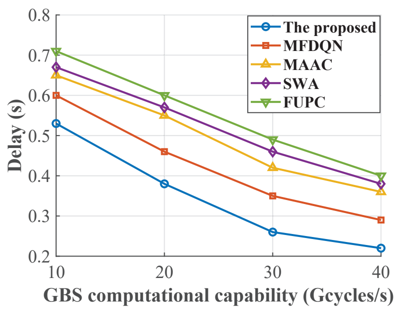
Fig. 3 compares the convergence performance of MFAC, MAAC and MFDQN algorithms. As can be seen from the figure, the three algorithms all can converge, and the proposed MFAC algorithm achieves an 80% and a 60% reduction in the number of episodes required to reach convergence compared with MAAC and MFDQN algorithms, respectively. This is primarily because the mean field approximation in MFAC simplifies each agents learning problem by replacing interactions with all other agents by an aggregated mean action. As a result, the neural network's input dimensionality is significantly reduced, thereby lowering computational complexity and accelerating convergence. Second, MFAC outperforms MFDQN in convergence speed not only due to reduced input complexity, but also to fundamental differences in algorithmic structure. While MFDQN integrates mean field information into the Q-function, it still requires the Q-network to adapt to the evolving action distributions of neighboring agents. This introduces greater learning complexity and increases sensitivity to non-stationarity, as the underlying policy dynamics of other agents continuously shift during training. In contrast, MFAC directly applies mean field approximation within the



(a) $w_1 = 0.1, w_2 = 0.9$

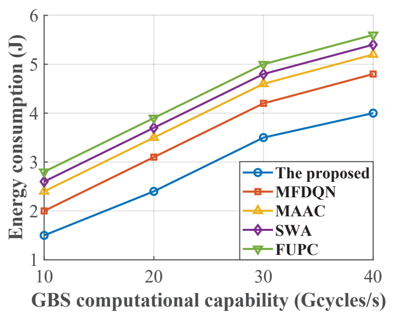


(b) $w_1 = 0.5, w_2 = 0.5$

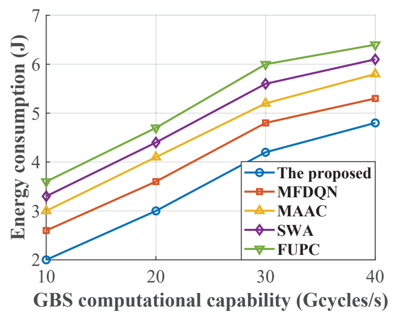


(c) $w_1 = 0.9, w_2 = 0.1$

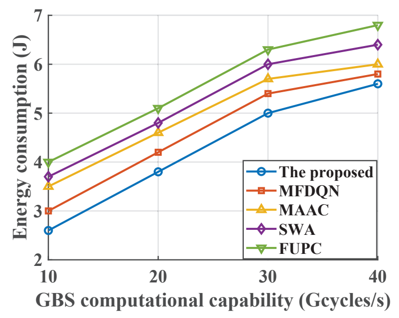
Fig. 5. Delay vs. GBS computational capability.



(a) $w_1 = 0.1, w_2 = 0.9$



(b) $w_1 = 0.5, w_2 = 0.5$



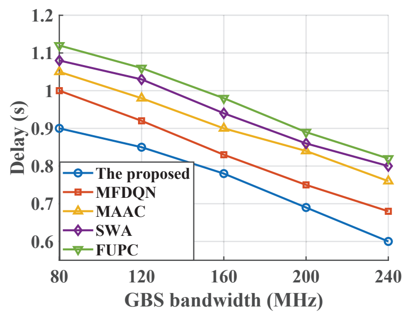
(c) $w_1 = 0.9, w_2 = 0.1$

Fig. 6. Energy consumption vs. GBS computational capability.

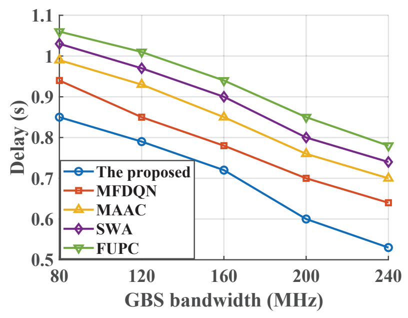
actor-critic framework, allowing each agent to optimize its policy based only on local observations and the average behavior of others. This design enables scalability with respect to the number of agents and promotes greater learning stability. Furthermore, MFAC benefits from the inherent advantages of policy gradient methods, which often yield more stable convergence in multi-agent environments compared with value-based approaches such as DQN. Third, after convergence, the proposed algorithm achieves a higher reward compared with MAAC and MFDQN algorithms. This is primarily attributed to the use of mean field approximation, which enables scalable and stable policy learning by reducing the dependence on high-dimensional joint action spaces. Moreover, the actor-critic structure of MFAC, combined with policy gradient updates, facilitates better convergence to near-optimal strategies in large-scale multi-agent settings.

Fig. 4 compares the training time of MFDQN, MAAC, and MFAC algorithms with different numbers of UAVs. As can be seen from the figure, first, the training time of both MFAC and MFDQN increases approximately linearly with the number of UAVs, and the training time of MFDQN is slightly higher than that of MFAC. Second, the training time of the MAAC algorithm grows significantly with the number of agents and incurs the highest computational cost among the three methods. For example, when the number of UAVs is 25, the training time of MFAC achieves reductions of 10% and 70% compared with MFDQN and MFAC. This phenomenon can be explained as follows. In MFAC and MFDQN, the use of mean field approximation decouples the dependency on the joint agent actions, resulting in fixed-size network inputs for each agent. Consequently, the overall computational complexity scales linearly with the number of agents. Although both methods benefit from this scalability, MFDQN typically requires more training time than MFAC due to the greater instability and learning complexity associated with value-based methods, which often need more updates and longer convergence periods compared with actorCcritic methods. In contrast, MAAC requires each agent's critic network to process the full joint observation and action space, whose dimensionality increases with the number of agents. This leads to significantly higher computational demands for policy learning and results in slower convergence as the network must explore a much larger input space.

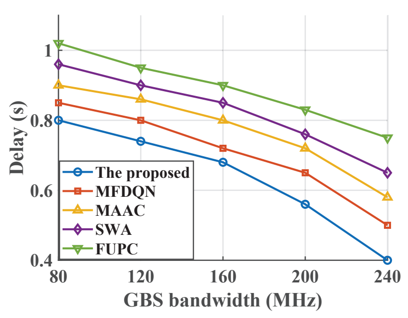
Figs. 5 and 6 compare the delay and energy consumption under varying GBS computational capabilities and different weighted coefficients. As can be seen from the figures, on the one hand, as the GBS computational capability increases, delay decreases and energy consumption increases under the four solutions. On the other hand, the MFAC algorithm achieves the lowest delay and energy consumption, while FUPC has the highest delay and energy consumption for a fix GBS computational capability. For example, with a GBS computational capability of 10 Gcycles/s, MFAC achieves delay reductions of 8%, 14%, 15.5% and 17.7% compared with MFDQN, MAAC, SWA and FUPC. Furthermore, MFAC achieves energy consumption reductions of 25%, 37.5%, 42% and 46% compared with MFDQN, MAAC, SWA and FUPC. The reasons behind this phenomenon are that, first, stronger



(a) $w_1 = 0.1, w_2 = 0.9$

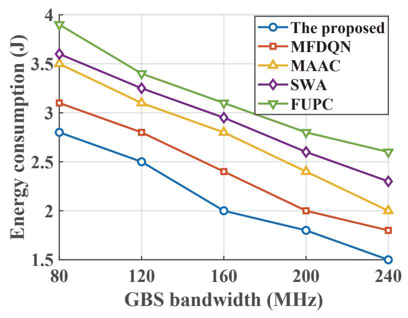


(b) $w_1 = 0.5, w_2 = 0.5$

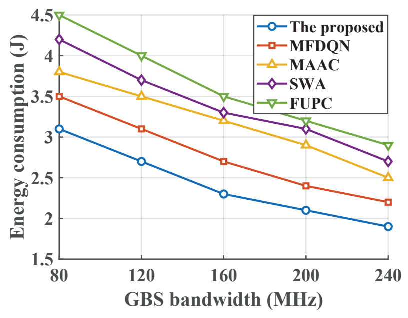


(c) $w_1 = 0.9, w_2 = 0.1$

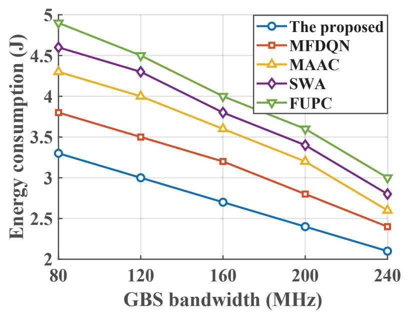
Fig. 7. Delay vs. GBS bandwidth.



(a) $w_1 = 0.1, w_2 = 0.9$



(b) $w_1 = 0.5, w_2 = 0.5$



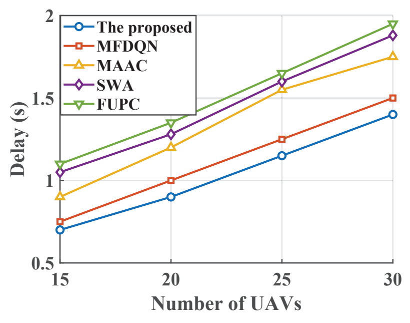
(c) $w_1 = 0.9, w_2 = 0.1$

Fig. 8. Energy consumption vs. GBS bandwidth.

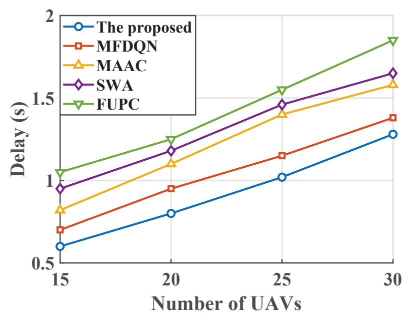
edge computing encourages more task offloading and triggers more aggressive repositioning and uplink transmissions. Although each transmission is faster, the cumulative flight and transmission energy grow, leading to higher overall energy cost. Second, in the FUPC method, UAVs operate under static deployment without adapting their positions to task demand or channel conditions, leading to inefficient communication links, longer transmission delays, and unbalanced resource usage. Static UAV positioning limits the system's ability to optimize link quality and workload distribution, which ultimately results in poor task offloading performance. SWA method adopts a greedy SINR-based association followed by local position adjustment, ignoring the energy-delay trade-off and lacking coordination among UAVs. This makes it prone to congestion and constraint conflicts, ultimately leading to higher delay and energy consumption. MFDQN adopts mean field approximation, with value-based updates making it more sensitive to non-stationarity. MAAC depends on a centralized critic whose input dimension grows linearly with the number of UAVs, which degrades stability as K increases. The MFAC integrates mean field theory with the actor-critic architecture, allowing each agent to make decisions based on the average behavior of others in a scalable manner. By leveraging this approximation, the MFAC algorithm effectively captures the interactions among agents while significantly reducing computational complexity. Consequently, it enables more efficient allocation of communication and computational resources, leading to improved energy efficiency and lower task latency, while satisfying delay constraints.

Furthermore, it can be observed that for any scheme, as w_1 increases, and w_2 decreases, with the computational capability of UAVs fixed, the delay shows a downward trend, indicating that the system prioritizes lower task delay under higher delay weights. Meanwhile, energy consumption gradually rises because energy consumption optimization is given lower priority. These results clearly illustrate the trade-off between energy consumption and delay, with the magnitude and rate of change varying across different optimization schemes, reflecting their respective adaptability and effectiveness under dynamically adjusted weights.

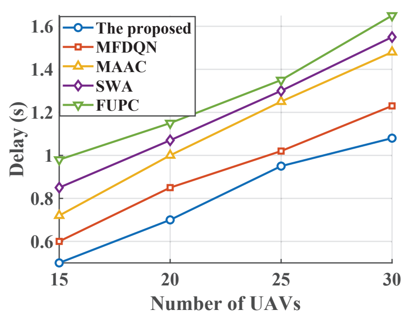
As can be seen from Figs. 7 and Fig. 8, on the one hand, increasing the bandwidth of the GBSs leads to a reduction in delay and energy consumption. On the other hand, under the same bandwidth, the MFAC algorithm achieves the lowest delay and energy consumption, whereas the FUPC scheme achieves the highest. For example, with a GBS bandwidth of 80 MHz, the delay of MFAC achieves a reduction of 10%, 14%, 16.7% and 19.6% compared with MFDQN, MAAC, SWA and FUPC. Furthermore, MFAC achieves energy consumption reductions of 9.6%, 20%, 22% and 28% compared with MFDQN, MAAC, SWA and FUPC. The reasons behind this phenomenon are as follows, first, as the available bandwidth increases, the transmission rate rises. A higher rate shortens the end-to-end task delay. A shorter transmission time also reduces transmission energy. Second, in MFAC, task offloading and position decisions are jointly optimized using a mean field approximation, MFAC accounts for the influence of neighbors, and avoids the input dimensionality increasing



(a) $w_1 = 0.1, w_2 = 0.9$

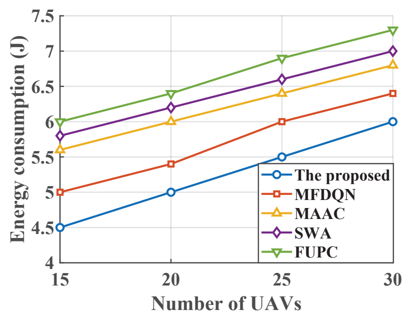


(b) $w_1 = 0.5, w_2 = 0.5$

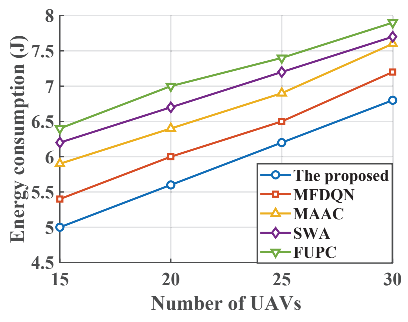


(c) $w_1 = 0.9, w_2 = 0.1$

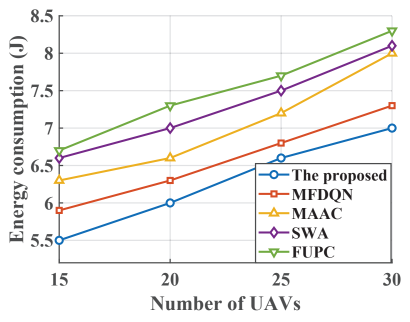
Fig. 9. Delay vs. number of UAVs.



(a) $w_1 = 0.1, w_2 = 0.9$



(b) $w_1 = 0.5, w_2 = 0.5$



(c) $w_1 = 0.9, w_2 = 0.1$

Fig. 10. Energy consumption vs. number of UAVs.

with the number of agents, thereby enabling decisions that achieve the lowest delay and energy consumption. MFDQN incorporates mean-field approximation but uses a value-based update, which is more sensitive to nonstationary in multi-agent systems and thus suffers larger estimation bias. As a result, its overall performance is worse than MFAC. Nevertheless, because it leverages mean-field approximation, it outperforms MAAC. SWA tends to associate many UAVs with a few high-SINR GBSs, causing contention for computing resources; despite high per-link rates, the overall execution delay becomes longer. Meanwhile, SWA only performs local position adjustments and lacks multi-UAV coordination under geometric constraints, which increases mobility energy and degrades link quality. By contrast, learning-based methods adaptively balance load under dynamic environments, thus achieving lower delay and energy under the same bandwidth. In the FUPC method, UAVs operate with fixed positions, which prevents them from dynamically responding to variations in task distribution, channel quality, and network load. As a result, tasks may be offloaded inefficiently, leading to longer transmission distances, increased queuing delays, and suboptimal use of computational resources. Furthermore, it can be observed that for any scheme, as w_1 increases, and w_2 decreases, with the GBS bandwidth fixed, the delay shows a downward trend, indicating that the system prioritizes lower task delay under higher delay weights. Meanwhile, energy consumption gradually rises because energy consumption optimization is given lower priority.

In Figs. 9 and 10, we compare the delay and energy consumption under different number of UAVs and different weighting coefficients. As can be seen from the figures, as the number of UAVs increases, MFAC achieves the lowest delay and energy consumption, and FUPC achieves the highest energy consumption and delay. For example, when the number of UAVs is 15, MFAC achieves delay reductions of 6%, 22%, 33%, and 36% compared with MFDQN, MAAC, SWA, and FUPC. Furthermore, MFAC achieves energy-consumption reductions of 10%, 19.6%, 22.4%, and 25% compared with MFDQN, MAAC, SWA, and FUPC. The reasons behind this phenomenon are that, first, when more UAVs are present, the transmission rate per UAV decreases. The slower rate increases transmission delay, and the longer delay increases energy consumption. Second, MFAC utilizes mean field approximation within the actor-critic framework, enabling each UAV to make decisions based on local observations and the average behavior of others. This leads to highly efficient and scalable resource allocation, especially as the number of UAVs increases. MAAC, while effective in small-scale settings due to its centralized critic structure, suffers from scalability issues as it requires modeling the joint action space, resulting in increased training complexity and degraded performance in larger networks. MFDQN partially mitigates this issue by employing value-based learning with mean field approximation, but its reliance on Q-learning makes it more sensitive to non-stationarity and less stable during training. SWA ignores multi-UAV coordination and does not balance the trade-off between energy and delay. As a result, it tends to attach many UAVs to a few high-SINR GBSs, causing compute-resource

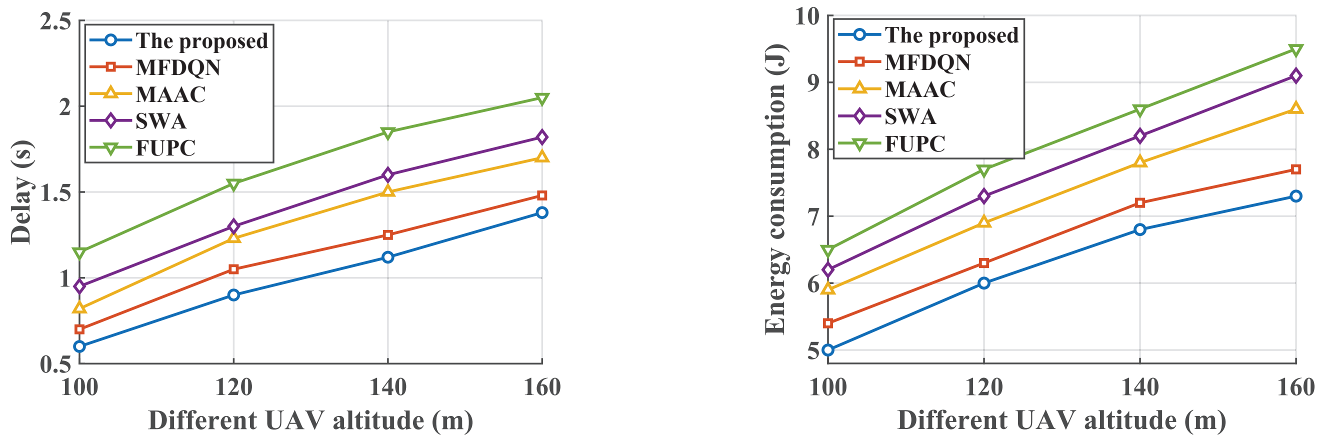


Fig. 11. Delay and Energy consumption vs. different UAV altitude.

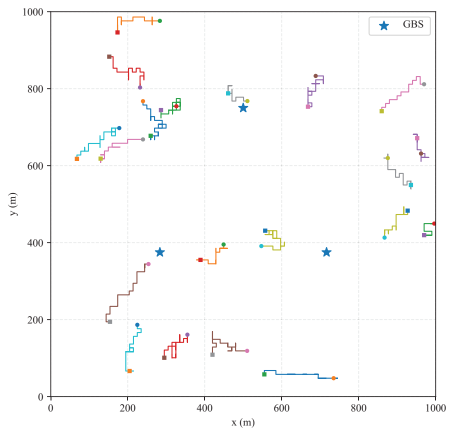


Fig. 12. The trajectory of 20 UAVs.

congestion and conflicts with geometric constraints, which in turn leads to higher execution latency and energy consumption. FUPC, which assumes fixed UAV positions and lacks adaptive learning or coordination mechanisms, is unable to dynamically adjust to changing task loads or network conditions. As a result, it demonstrates the worst performance in terms of both energy consumption and delay, particularly as the number of UAVs increases and resource contention intensifies.

In addition to the LoS baseline, we include a Rician-fading model where each link gain is $g_{k,n} = \beta_0\, d_{k,n}^{-\alpha}\, |h_{k,n}|^2$, $h_{k,n} = \sqrt{\frac{K}{K+1}}\, e^{j\phi_{k,n}} + \sqrt{\frac{1}{K+1}}\,\tilde{h}_{k,n}$, where K is the Rician K-factor, $\phi_{k,n} \sim \mathcal{U}[0, 2\pi)$, and $\tilde{h}_{k,n} \sim \mathcal{CN}(0, 1)$, and we examine the impact of UAV altitude by sweeping h_k over a representative range while keeping all other parameters fixed, and report the resulting delay and energy consumption under both channel models. As can be seen from Fig. 11, as the UAV flies higher, the link distance to ground stations becomes longer. This increases path loss, which lowers the signal to noise ratio. A lower signal to noise ratio leads to a lower data rate, so the same amount of data takes more time to transmit. The longer transmission time raises the overall delay. Energy consumption also increases. Transmit energy grows with the time spent sending data, and UAVs may need to use more power to maintain a reliable link. In some cases, the UAV system shifts more tasks to local computing to meet latency constraints when the link is poor, which adds computing energy. Higher altitude can also expose the UAV to more LoS interference from other UAVs, further reducing the effective link and amplifying the increase in both delay and energy consumption.

As shown in the Fig. 12, the learned trajectories exhibit three consistent properties: first, UAVs autonomously migrate toward high-SINR regions and nearby GBSs, shortening links and reducing path loss, thereby markedly lowering end-to-end latency and communication energy. Second, the mean-field policy encourages spatial dispersion by referencing neighbors average behavior, avoiding crowding at any single base station while improving resource utilization and system throughput. Third, trajectories progressively settle into stable dwell points with substantially reduced oscillation in later stages, demonstrating good convergence and training stability, with greater robustness to bandwidth and load perturbations.

V. Conclusion

In this paper, we investigated the task offloading and UAV position optimization in UAV networks. By integrating mean field approximation and MADRL, we developed a low-complexity MFAC algorithm for task offloading and position optimization in large-scale UAV networks. The representation capability of mean field theory, combined with the adaptability of DRL in handling uncertain and dynamic environments, enables the proposed algorithm to make robust decisions in large-scale UAV networks. Numerical results show the effectiveness of the proposed scheme. MFAC outperforms the MFDQN, MAAC and FUPC schemes with respect to convergence performance, training time, energy consumption and delay. The results highlight the decision-making efficiency of MFAC, positioning it as an effective approach for task offloading in large-scale UAV-assisted networks. In our future work, we will focus on the following potential directions. First, we aim to extend the current framework to heterogeneous UAV scenarios with different computing and communication capabilities. Second, incorporating advanced reflected intelligent surface (RIS) and fluid antenna (FA) would further enhance resource utilization in large-scale UAV networks. Third, we will investigate three-dimensional UAV position optimization and incorporate the effects of downlink transmission on system latency and energy consumption. Lastly, we intend to carry out real-world deployments to further assess the feasibility and applicability of the proposed approach in practical UAV networks.

Appendix A
Mean-Field Approximation

A. Preliminaries

We consider the K-player Markov game with local observations o_k, joint action $\mathbf{a}$, and value

$$v_k(s) = \mathbb{E}_\pi\left[\sum_t \gamma^t r_k^t\right] = \mathbb{E}_\pi\left[\sum_t \gamma^t \sum_{\mathbf{a}(t)} r_k(\mathbf{s}(t), \mathbf{a}(t))\, \pi(\mathbf{s}(t), \mathbf{a}(t))\right]. \tag{A.1}$$

Thus,

$$v_k^*(\mathbf{s}(t)) = \max_{\pi_k} \mathbb{E}_{\pi_k}\left[Q_{\pi^*_{-k}}(\mathbf{s}(t), \mathbf{a}(t))\right], \tag{A.2}$$

with

$$Q_{\pi^*_{-k}}(\mathbf{s}(t), \mathbf{a}(t)) = \mathbb{E}_{\pi^*_{-k}}\Big[r_k(\mathbf{s}(t), \mathbf{a}(t)) + \gamma \sum_{\mathbf{s}(t+1)} p(\mathbf{s}(t+1)|\mathbf{s}(t), \mathbf{a}(t))\, v_k(\mathbf{s}(t+1))\Big]. \tag{A.3}$$

Variable $Q_{\pi^*_{-k}}(\mathbf{s}(t), \mathbf{a}(t))$ represents the marginal value for selecting action $a_k(t)$ at state $\mathbf{s}(t)$ by UAV k. Due to each UAV only can obtain local information of the system to make decisions. Thus, we use $o_k(t)$ to replace $\mathbf{s}(t)$.

B. Pairwise Factorization and Mean-Field Action

Calculating the Q-function $Q^*_{\pi_{-k}}(o_k(t), \mathbf{a}(t))$ depends on the joint action $\mathbf{a}(t)$ of all UAVs, however, UAV k can not obtain other UAVs' actions $\mathbf{a}_{-k}(t)$, and with the increasing of UAVs, learning standard Q-function $Q_{\pi^*_{-k}}(o_k(t), \mathbf{a}(t))$ becomes infeasible due to the large action space of joint action. To address this issue, we use pairwise local interaction to factorized the Q-function $Q^*_{\pi_{-k}}(o_k(t), \mathbf{a}(t))$ as

$$Q_{\pi^*_{-k}}(o_k^t, \mathbf{a}^t) = \frac{1}{N^k} \sum_{k' \in \mathcal{N}(k)} Q_{\pi^*_{-k}}(o_k^t, a_k^t, a_{k'}^t). \tag{A.4}$$

where the mean action $\bar{a}_k^t = \frac{1}{N^k}\sum_{k'} a_{k'}^t$ and $a_{k'}^t = \bar{a}_k^t + \xi a_{k,k'}^t$.

Theorem 1 (Mean-field approximation): The Q-function $Q_{\pi^*_{-k}}(o_k^t, \mathbf{a}^t)$ of TV n can be approximated by mean field action $\bar{a}_k^t$ and action a_k^t, i.e.,

$$Q_{\pi^*_{-k}}(o_k^t, \mathbf{a}^t) = Q_{\pi^*_{-k}}(o_k^t, a_k^t, \bar{a}_k^t). \tag{A.5}$$

To facilitate understanding, interested readers are referred to the similar proof presented in [26], [45]–[47]. Thus, with the aid of pairwise interaction and mean field approximation, the $Q_{\pi^*_{-k}}(o_k^t, \mathbf{a}^t)$ can be approximated by $Q_{\pi^*_{-k}}(o_k^t, a_k^t, \bar{a}_k^t)$.

C. Mean-Field Bellman Update

Using (A.5), the mean-field critic update is

$$Q(o_k^t, a_k^t, \bar{a}_k^t) \leftarrow (1-\eta)\, Q(o_k^t, a_k^t, \bar{a}_k^t) + \alpha\left(r_k^t + \gamma\, v_t^k(s')\right), \tag{A.7}$$

where

$$v_t^k(s') = \sum_{a_k} \pi_k^t(a_k|s', \bar{a}_k)\, \mathbb{E}_{\bar{a}_k(\mathbf{a}_{-k}) \sim \boldsymbol{\pi}^t_{-k}}\left[Q(o_k^t, a_k^t, \bar{a}_k^t)\right]. \tag{A.8}$$

References

[1] H. Xu *et al.*, "A Survey on UAV Applications in Smart City Management: Challenges, Advances, and Opportunities," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 16, pp. 8982-9010, Sep. 2023.

[2] Z. Zhou, Y. Tian, J. Xiong, J. Ma and C. Peng, "Blockchain-Enabled Secure and Trusted Federated Data Sharing in IIoT," *IEEE Trans. Ind. Informat.*, vol. 19, no. 5, pp. 6669-6681, May 2023.

[3] X. Zhang *et al.*, "A Data Trading Scheme With Efficient Data Usage Control for Industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 18, no. 7, pp. 4456-4465, Jul. 2022.

[4] Y. Wang *et al.*, "Task Offloading for Post-Disaster Rescue in Unmanned Aerial Vehicles Networks," *IEEE/ACM Trans. Netw.*, vol. 30, no. 4, pp. 1525-1539, Aug. 2022.

[5] H. Gu, L. Zhao, Z. Han, G. Zheng and S. Song, "AI-Enhanced Cloud-Edge-Terminal Collaborative Network: Survey, Applications, and Future Directions," *IEEE Commun. Surveys Tuts.*, vol. 26, no. 2, pp. 1322-1385, 2ndquarter 2024.

[6] Z. Ning *et al.*, "Mobile Edge Computing and Machine Learning in the Internet of Unmanned Aerial Vehicles: A Survey", *ACM Comput. Surv.*, vol. 56, no. 1, pp. 1-31, Jan. 2024.

[7] F. Khoramnejad, A. Syed, W. Sean Kennedy and M. Erol-Kantarci, "Energy and Delay Aware General Task Dependent Offloading in UAV-Aided Smart Farms," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 5, pp. 5033-5048, Oct. 2024.

[8] G. Sun et al., "Joint Task Offloading and Resource Allocation in Aerial-Terrestrial UAV Networks With Edge and Fog Computing for Post-Disaster Rescue," *IEEE Trans. Mobile Comput.*, vol. 23, no. 9, pp. 8582-8600, Sept. 2024.

[9] H. He, X. Yang, F. Huang, H. Shen and H. Tian, "Enhancing QoE in Large-Scale U-MEC Networks via Joint Optimization of Task Offloading and UAV Trajectories," *IEEE Internet Things J.*, vol. 11, no. 21, pp. 35710-35723, Nov. 2024.

[10] Z. Chen, Y. Yang, J. Xu, Y. Chen and J. Huang, "Task Offloading and Resource Pricing Based on Game Theory in UAV-Assisted Edge Computing," *IEEE Trans. Serv. Comput.*, vol. 18, no. 1, pp. 440-452, Jan. 2025.

[11] Y. Chen, J. Zhao, Y. Wu, J. Huang and X. S. Shen, "Multi-User Task Offloading in UAV-Assisted LEO Satellite Edge Computing: A Game-Theoretic Approach," *IEEE Trans. Mobile Comput.*, vol. 24, no. 1, pp. 363-378, Jan. 2025.

[12] Z. Kuang, H. Wang, J. Li and F. Hou, "Utility-Aware UAV Deployment and Task Offloading in Multi-UAV Edge Computing Networks," *IEEE Internet Things J.*, vol. 11, no. 8, pp. 14755-14770, Apr. 2024.

[13] M. Landers and A. Doryab, "Deep Reinforcement Learning Verification: A Survey", *ACM Comput. Surv.*, vol. 55, no. 14, pp. 1-31, Jul. 2023.

[14] L. Jia *et al.*, "Game Theory and Reinforcement Learning for Anti-Jamming Defense in Wireless Communications: Current Research, Challenges, and Solutions," *IEEE Commun. Surv. Tutor.*, vol. 27, no. 3, pp. 1798-1838, Jun. 2025.

[15] N. C. Luong *et al*., "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," *IEEE Commun. Surveys Tuts*., vol. 21, no. 4, pp. 3133-3174, 4thquarter 2019.

[16] H. Hao, C. Xu, W. Zhang, S. Yang and G. -M. Muntean, "Joint Task Offloading, Resource Allocation, and Trajectory Design for Multi-UAV Cooperative Edge Computing With Task Priority," *IEEE Trans. Mobile Comput*., vol. 23, no. 9, pp. 8649-8663, Sep. 2024.

[17] J. Zhang, G. Zhang, X. Wang, X. Zhao, P. Yuan and H. Jin, "UAV-Assisted Task Offloading in Edge Computing," *IEEE Internet Things J*., vol. 12, no. 5, pp. 5559-5574, Mar. 2025.

[18] C. Liu, Y. Zhong, R. Wu, S. Ren, S. Du and B. Guo, "Deep Reinforcement Learning Based 3D-Trajectory Design and Task Offloading in UAV-Enabled MEC System," *IEEE Trans. Veh. Technol*., vol. 74, no. 2, pp. 3185-3195, Feb. 2025.

[19] H. Chen, H. Cui, J. Wang, P. Cao, Y. He and M. Guizani, "Computation Offloading Optimization for UAV-Based Cloud-Edge Collaborative Task Scheduling Strategy," *IEEE Trans. Cogn. Commun. Netw*., early access, doi: 10.1109/TCCN.2025.3544822.

[20] X. Wang *et al*., "Deep Reinforcement Learning: A Survey," *IEEE Trans. Neural Netw. Learn. Syst.*., vol. 35, no. 4, pp. 5064-5078, Apr. 2024.

[21] J. Li *et al*., "A Learning-Based Stochastic Game for Energy Efficient Optimization of UAV Trajectory and Task Offloading in Space/Aerial Edge Computing," *IEEE Trans. Veh. Technol*., vol. 74, no. 6, pp. 9717-9733, Jun. 2025.

[22] M. Hevesli, A. M. Seid, A. Erbad and M. Abdallah, "Multi-Agent DRL for Queue-Aware Task Offloading in Hierarchical MEC-Enabled Air-Ground Networks," *IEEE Trans. Cog. Commun. Netw*., early access, doi: 10.1109/TCCN.2025.3555440.

[23] Z. Gao, J. Fu, Z. Jing, Y. Dai and L. Yang, "MOIPC-MAAC: Communication-Assisted Multiobjective MARL for Trajectory Planning and Task Offloading in Multi-UAV-Assisted MEC," *IEEE Internet Things J*., vol. 11, no. 10, pp. 18483-18502, May 2024.

[24] X. Li, X. Du, N. Zhao and X. Wang, "Computing Over the Sky: Joint UAV Trajectory and Task Offloading Scheme Based on Optimization-Embedding Multi-Agent Deep Reinforcement Learning," *IEEE Trans. Commun*., vol. 72, no. 3, pp. 1355-1369, Mar. 2024.

[25] K. Li, Y. Hu, J. Wang, L. Li, S. Zhang and G. Luo, "Task Prioritization in Multiagent Environments: A Novel Approach Using Nash Q-Learning," *IEEE Trans. Consum. Electron*., early access, doi: 10.1109/TCE.2025.3542833.

[26] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *Proc. Mach. Learn. Res. (ICML)*., Jul. 2018, pp. 5567-5576.

[27] Y. Xu, L. Zheng, X. Wu, Y. Tang, W. Liu and D. Sun, "Joint Resource Allocation for UAV-Assisted V2X Communication With Mean Field Multi-Agent Reinforcement Learning," *IEEE Trans. Veh. Technol*., vol. 74, no. 1, pp. 1209-1223, Jan. 2025.

[28] Z. Gao, G. Wang, L. Yang and Y. Dai, "Transfer Learning for Joint Trajectory Control and Task Offloading in Large-scale Partially Observable UAV-Assisted MEC," *IEEE Trans. Mobile Comput*., early access, doi: 10.1109/TMC.2025.3579748.

[29] F. Song, Z. Wang, J. Li, L. Shi, W. Chen, S. Jin, "Dynamic Trajectory and Power Control in Ultra-Dense UAV Networks: A Mean-Field Reinforcement Learning Approach", arXiv:2411.14052.

[30] S. Akter, D. Van Anh Duong and S. Yoon, "Joint Optimization of AAV Trajectory, Task Offloading, and Resource Allocation in AAV-Aided Emergency Response Operations," *IEEE Internet Things J*., vol. 12, no. 12, pp. 21944-21959, Jun. 2025.

[31] M. Wu, H. Wu, W. Lu, L. Guo, I. Lee and A. Jamalipour, "Security-Aware Designs of Multi-UAV Deployment, Task Offloading and Service Placement in Edge Computing Networks," *IEEE Trans. Mobile Comput*., doi: 10.1109/TMC.2025.3574061.

[32] S. Shakoor, Z. Kaleem, D.-T. Do, O. A. Dobre, and A. Jamalipour, "Joint optimization of UAV 3D placement and path loss factor for energy efficient maximal coverage," *IEEE Internet Things J*., vol. 8, no. 12, pp. 9776-9786, Jun. 2021.

[33] L. Zhang and N. Ansari, "Approximate algorithms for 3-D placement of IBFD enabled drone-mounted base stations," *IEEE Trans. Veh. Technol*., vol. 68, no. 8, pp. 7715-7722, Aug. 2019.

[34] J. Tian, D. Wang, H. Zhang and D. Wu, "Service Satisfaction-Oriented Task Offloading and UAV Scheduling in UAV-Enabled MEC Networks," *IEEE Trans. Wireless Commun*., vol. 22, no. 12, pp. 8949-8964, Dec. 2023.

[35] L. Huang, S. Bi, and Y. -J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput*., vol. 19, no. 11, pp. 2581-2593, Nov. 2020.

[36] Y. Zeng, S. Chen, J. Li, Y. Cui and J. Du, "Online Optimization in UAV-Enabled MEC System: Minimizing Long-Term Energy Consumption Under Adapting to Heterogeneous Demands," *IEEE Internet Things J*., vol. 11, no. 19, pp. 32143-32159, Oct. 2024.

[37] Y. Zeng and R. Zhang, "Energy-Efficient UAV Communication With Trajectory Optimization," *IEEE Trans. Wireless Commun*., vol. 16, no. 6, pp. 3747-3760, Jun. 2017.

[38] J. Ji, K. Zhu, C. Yi, and D. Niyato, "Energy consumption minimization in UAV-assisted mobile-edge computing systems: Joint resource allocation and trajectory design," *IEEE Internet Things J*., vol. 8, no. 10, pp. 8570-8584, May 2021.

[39] D. Yang, Q. Wu, Y. Zeng, and R. Zhang, "Energy trade-off in ground-to-UAV communication via trajectory design," *IEEE Trans. Veh. Technol*., vol. 67, no. 7, pp. 6721-6726, Jul. 2018.

[40] S. Guo, J. Liu, Y. Yang, B. Xiao and Z. Li, "Energy-Efficient Dynamic Computation Offloading and Cooperative Task Scheduling in Mobile Cloud Computing," *IEEE Trans. Mob. Comput*., vol. 18, no. 2, pp. 319-333, Feb. 2019.

[41] Z. Yu, Y. Gong, S. Gong, and Y. Guo, "Joint task offloading and resource allocation in UAV-enabled mobile edge computing," *IEEE Internet Things J*., vol. 7, no. 4, pp. 3147-3159, Apr. 2020.

[42] J. Hu and M. P. Wellman, "Nash Q-Learning for General-Sum Stochastic Games", *J. Mach. Learn. Res.*, vol. 4, pp. 1039-1069, Nov. 2003.

[43] X. Meng, W. Wang, Y. Wang, V. K. N. Lau, and Z. Zhang, "Closed-form delay-optimal computation offloading in mobile edge computing systems," *IEEE Trans. Wireless Commun*., vol. 18, no. 10, pp. 4653-4667, Oct. 2019.

[44] Y. Wang, M. Kong, G. Zhang, W. Wang, T. Nakachi and J. Liou, "Adaptive Task Offloading for Mobile Edge Computing With Forecast Information," *IEEE Trans. Veh. Technol*., vol. 74, no. 3, pp. 4132-4147, Mar. 2025.

[45] X. Wang, Z. Ning, L. Guo, S. Guo, X. Gao and G. Wang, "Mean-Field Learning for Edge Computing in Mobile Blockchain Networks," *IEEE Trans. Mobile Comput*., vol. 22, no. 10, pp. 5978-5994, Oct. 2023.

[46] G. Wu, H. Wang, H. Zhang, Y. Shen, S. Shen and S. Yu, "Mean-Field Game-Based Task-Offloaded Load Balance for Industrial Mobile Edge Computing Systems Using Software-Defined Networking," *IEEE Trans. Mobile Comput*., vol. 23, no. 12, pp. 13773-13786, Dec. 2024.

[47] Z. Zhang, L. Lu, Q. Li, Y. Chai, D. Wu and Y. Zhang, "Joint AI Service Placement, Task Scheduling, and Resource Allocation for IoT in 6G Networks," *IEEE Internet Things J*., vol. 12, no. 18, pp. 39042-39060, Sep. 2025.