Energy-aware Path Planning for Obtaining Fresh Updates in UAV-IoT MEC systems

Hao Chen*, Xiaoqi Qin*, Yixuan Li*, Nan Ma*†
*State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing, China
†Peng Cheng Laboratory, Shenzhen, China
Email:{2020110229, xiaoqiqin, lyixuan, manan}@bupt.edu.cn

2022 IEEE Wireless Communications and Networking Conference (WCNC) | 978-1-6654-4266-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/WCNC51071.2022.9771867

***Abstract*—The ubiquitous computing resource at UAVs and IoT devices can be exploited in conjunction to form a UAV-IoT edge computing system for low-cost and responsive environmental monitoring. Under stochastic computational task arrival at IoT devices, one major challenge is how to realize path control for multiple UAVs and energy efficient computation offloading in real time. Moreover, the freshness of obtained updates is of critical importance to the system performance under such time-critical scenarios. In this paper, we employ the concept of age of information (AoI) to quantify the timeliness of updates at IoT devices, and formulate an energy minimization problem by jointly considering UAV path planning, energy consumption of computation offloading and age evolution of updates. To solve the formulated problem, we propose a deep reinforcement learning based solution to achieve fast decision making. Simulation results show that the performance of proposed solution is competitive in terms of obtaining fresh updates at low energy cost.**

***Index Terms*—UAV, Path Planning, Age of Information, MEC, Deep Reinforcement Learning**

I. INTRODUCTION

Smart Internet of Things with pervasive computing capabilities have shown its great potential in detecting early signs of public hazards automatically with least human intervention, by monitoring the changes of physical environment (e.g., wildfire detection and intrusion management). Despite the salient advantage of reducing labor cost, it is hard to achieve precise hazard identification relying solely on power-and-computation-constrained IoT devices. Owing to their flexibility in deployment, Unmanned Aerial Vehicles (UAVs) can provide computing service in close proximity to IoT devices from bird's eye view [1]. The team-up between UAVs and IoTs results in a low-cost yet effective edge computing system for responsive environmental monitoring.

To achieve timely situational awareness at IoT devices, one major challenge is how to realize real-time path planning for multiple UAVs and energy efficient computation offloading under stochastic computational task arrival at devices. As for path planning, there have been extensive studies on optimizing the visiting order of IoT devices under straight-line and circular trajectory patterns [2] [3]. To better leverage the flexibility of UAVs, path planning with irregular trajectory patterns were proposed to control the flying directions of UAVs. In [4], the UAV's trajectory was approximated by a sequence of discretized locations to combat the infinite flying choices and solved by convex optimization techniques. In [5], Liu *et al.* employed deep reinforcement learning approach to realize adaptive path control for multiple UAVs. Other than UAV-assisted data collection, the on-board computation resource at UAVs can be exploited as a flying computing platform to assist task processing at IoT devices [6]. In [7], Peng *et al.* proposed a multi-agent reinforcement learning based computation offloading strategy for UAVs hovering above a specific area acting as complementary to base station. In [8], Wang *et al.* proposed a DDPG-based approach to achieve dynamic trajectory control and computation offloading.

In most existing work, the consecutively generated computation tasks at IoT devices are treated independently with everlasting value over time, and thus the strategy design focuses on energy efficiency [8], offloading throughput [9] and completion time [10]. However, as for time-critical control scenarios such as hazard detection, status updates extracted from computation tasks are temporally correlated and the freshness of updates is of critical importance to the detection accuracy. In this sense, we employ the concept of age of information (AoI) [11] to quantify the freshness of obtained updates, which is defined as the time elapsed since the last update is extracted from a buffered computation task. A larger value of age indicates that the status update at device is stale and is of less value for real-time cognition of monitored environment, thus the device should obtain a new update by executing computation task either locally or remotely at UAVs.

In this paper, we consider a UAV-IoT MEC system, where UAVs provide computing service to assist task execution at IoT devices. We develop an energy minimization problem by jointly considering UAVs path design, computation offloading and age evolution of updates. The formulated problem falls in the category of a mixed-integer nonlinear programming (MINLP) problem. To achieve real-time decision making, we propose a solution procedure based on double dueling deep Q-learning network (D3QN). Simulation results show that under our proposed strategy, IoT devices can obtain fresh updates at low energy costs.

The remainder of this paper is organized as follows. In Section II, we present the system model and problem formulation. In Section III, our proposed DRL-based solution is described. In Section IV, we present simulation results and analyze the results. Section V concludes this paper.

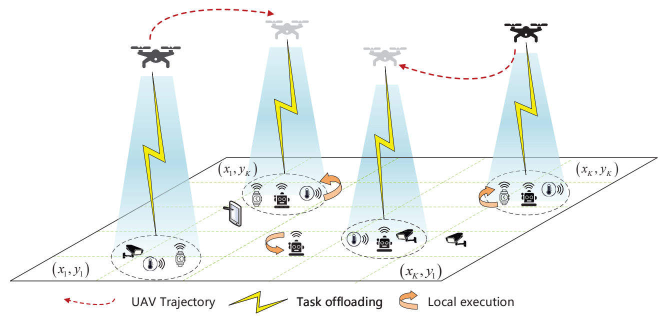


Fig. 1: System architecture.

II. MODELLING AND FORMULATION

In this section, we study an energy consumption minimization problem for UAV-IoT MEC systems. For a set of IoT devices, our objective is to minimize their energy consumption while maintaining the freshness of local updates by optimizing variables in UAV trajectory design and computation offloading decisions.

A. System Architecture

Consider a UAV-IoT edge computing system as shown in Fig 1, which consists of a set of $\mathcal{N}$ IoT devices and a set of $\mathcal{M}$ UAVs with constant altitude H, where $N = |\mathcal{N}|$ is the number of IoT devices and $M = |\mathcal{M}|$ is the number of UAVs. The UAVs collaboratively provide flying computing service to assist IoT devices to extract local updates from sequences of computational tasks. Each UAV selects a set of devices within its coverage to offload their tasks based on the trade-off between energy consumption of local execution and task offloading. Note that if the age of local update exceeds a certain threshold at an uncovered device, it will perform local execution. As for task offloading, OFDMA technique is utilized.

B. UAV Trajectory

To track the trajectory of each UAV, the geographical region of interest is equally partitioned into $K \times K$ small-size grids, and the UAV's location is approximately constant within each grid. Denote $\mathcal{L}$ as the set of locations of center of grids, where the center of the $k \times k$-th grid is represented by $L_{k\times k} = (x_k, y_k)$. We assume a time-slotted system, with a total number of T time slots of equal length τ. As for any UAV j, its trajectory U_j is approximated by a sequence of $\{U_j(1), U_j(2), ..., U_j(T)\}$, where $U_j(t) = \left(x_j^u(t), y_j^u(t)\right) \in \mathcal{L}$ is the location of grid that UAV j hovers over at time slot t, where $t \in \mathcal{T} = \{0, 1, 2, ..., T\}$. Then we have:

$$x_1 \le x_j^u(t) \le x_K \;, 1 \le j \le M \;, 1 \le t \le T \;. \tag{1}$$

$$y_1 \le y_j^u(t) \le y_K \;, 1 \le j \le M \;, 1 \le t \le T \;. \tag{2}$$

Denote $o_j(t) \in \mathcal{O} \triangleq \{N, S, E, W, I\}$ as moving direction of UAV j at time slot t, where N, S, E and W denote the north, south, east and west, while I indicates that UAV remains still. Denote L_0 as the spacing distance UAV moves during each time slot, then the relationship between UAV j's location at two consecutive time slots can be obtained as:

$$U_j(t+1) = \begin{cases} U_j(t) + (0, L_0) \;, & \text{if } o_j(t) = N, \\ U_j(t) - (0, L_0) \;, & \text{if } o_j(t) = S, \\ U_j(t) + (L_0, 0) \;, & \text{if } o_j(t) = E, \\ U_j(t) - (L_0, 0) \;, & \text{if } o_j(t) = W, \\ U_j(t) \;, & \text{if } o_j(t) = I. \end{cases} \tag{3}$$

C. Computation Offloading

As for each IoT device i, the generation of computation task is stochastic. Denote $S_i(t) = \{D_i(t), F_i(t)\}$ as the first task in device i's buffer at time slot t, where $D_i(t)$ is the size of task, $F_i(t)$ is the number of cycles required for task execution. Note that if the buffer is empty, we set $D_i(t) = F_i(t) = 0$. As for task $S_i(t)$, it can be handled either locally at device i (local execution or waiting), or at UAV (task offloading). Define $a_{i,j}(t)$ as a binary variable to indicate whether or not task $S_i(t)$ is executed at time slot t, where $j \in \mathcal{Z} = \{0, 1, 2, ..., M\}$. Note that $j = 0$ implies that the task is handled locally, i.e, $a_{i,0}(t) = 1$ if the task is executed locally, and $a_{i,0}(t) = 0$ if the task is waiting for processing. In the case when task $S_i(t)$ is offloaded to UAV $j \in \{1, 2, ..., M\}$, $a_{i,j}(t) = 1$. At the same time slot, we assume that only one of the three aforementioned actions can be taken. Then we have:

$$\sum_{j=0}^{M} a_{i,j}(t) \le 1 \;, 1 \le i \le N \;, 1 \le t \le T \;. \tag{4}$$

As for task offloading, device i can offload its task to UAV j only if it is within UAV j's coverage. Denote C as the horizontal coverage of a UAV, $d_{i,j}(t)$ as the horizontal distance between IoT device i and UAV j at time slot t, then we have:

$$\begin{aligned} & a_{i,j}(t) d_{i,j}(t) \le C \;, \\ & \qquad\qquad 1 \le i \le N \;, 0 \le j \le M \;, 1 \le t \le T \;. \end{aligned} \tag{5}$$

We assume that each UAV has a azimuth angle θ, then the horizontal coverage of a UAV hovering at a constant altitude H can be obtained as:

$$C = H \tan(\theta) \;. \tag{6}$$

Denote G_i as the coordinate of device i, then $d_{i,j}(t)$ can be calculated as:

$$d_{i,j}(t) = \begin{cases} \sqrt{\| (U_j(t) - G_i) \|^2} \;, & j \ne 0 \;; \\ 0 \;, & j = 0 \;. \end{cases} \tag{7}$$

Note that we set $d_{i,j}(t) = 0$ in the case when task is handled locally at device i ($j = 0$).

Moreover, as for the set of devices within UAV j's coverage, only a subset of devices can be served within the same time slot t due to limited communication resource. Assume that each IoT device i uses one subchannel for task offloading, and the total number of subchannels is Q, then we have:

$$\sum_{i=1}^{N} a_{i,j}(t) \le Q \;, 1 \le j \le M \;, 1 \le t \le T \;. \tag{8}$$

Without loss of generality, we assume that the execution of task $S_i(t)$ should be completed within one time slot, either locally at device i or remotely at UAV. Denote $T_{i,j}(t)$ as the required time for task execution, then we have:

$$T_{i,j}(t) \le \tau \ , i \in \mathcal{N} \ , j \in \mathcal{Z} \ , 1 \le t \le T \ . \tag{9}$$

Denote $T^{\text{loc}}_{i,j}(t)$ as the execution time of local computing, $T^{\text{off}}_{i,j}(t)$ as the time duration of task offloading, then we have:

$$\begin{aligned} T_{i,j}(t) = a_{i,j}(t) \left[sgn(j) T^{\text{off}}_{i,j}(t) + (1 - sgn(j)) \, T^{\text{loc}}_{i,j}(t) \right] , \\ 1 \le i \le N \ , 0 \le j \le M \ , 1 \le t \le T \ . \end{aligned} \tag{10}$$

In the case of local computing, denote $f^{\text{loc}}_i(t)$ as CPU frequency allocated for processing task $S_i(t)$ at device i, then the execution time can be obtained as:

$$T^{\text{loc}}_{i,j}(t) = \frac{F_i(t)}{f^{\text{loc}}_i(t)} \ , i \in \mathcal{N} \ , j = 0 \ , 1 \le t \le T \ . \tag{11}$$

In the case of task offloading, the time duration $T^{\text{off}}_{i,j}(t)$ consists of two parts: namely task transmission duration (denoted as $T^{\text{tr}}_{i,j}(t)$) and task execution duration at UAV j (denoted as $T^{\text{comp}}_{i,j}(t)$), then we have:

$$\begin{aligned} T^{\text{off}}_{i,j}(t) = T^{\text{tr}}_{i,j}(t) + T^{\text{comp}}_{i,j}(t) \ , \\ i \in \mathcal{N} \ , j \in \mathcal{M} \ , 1 \le t \le T \ . \end{aligned} \tag{12}$$

As for task transmission, we consider the randomness associated with the LoS and non-line-of-sight (NLoS) links. For ground-to-air communications, a specific IoT device typically has a LoS view towards UAV with a given probability. We introduce a common approach to model the LoS probability between IoT device i and UAV j as in [12]:

$$\begin{aligned} P^{\text{los}}_{i,j}(t) = \frac{1}{1 + \gamma \exp\left(-\psi \left[\zeta_{i,j}(t) - \gamma\right]\right)} \ , \\ i \in \mathcal{N} \ , j \in \mathcal{M} \ , t \in \mathcal{T} \ . \end{aligned} \tag{13}$$

where γ and ψ are constant parameters determined by the carrier frequency and type of environment, $\zeta_{i,j}(t) = \frac{180}{\pi} \sin^{-1}\left(\frac{H}{d_{i,j}(t)}\right)$ is the elevation angle at time slot t.

The channel gain $\beta_{i,j}(t)$ can be obtained as:

$$\beta_{i,j}(t) = \begin{cases} \frac{1}{\eta_1} \left(\frac{4\pi f_c d_{i,j}(t)}{c}\right)^{-\alpha} , & \text{LoS;} \\ \frac{1}{\eta_2} \left(\frac{4\pi f_c d_{i,j}(t)}{c}\right)^{-\alpha} , & \text{NLoS.} \end{cases} \tag{14}$$

where f_c is the carrier frequency, η_1 and η_2 ($\eta_2 > \eta_1 > 1$) are the excessive path loss coefficients in LoS and NLoS cases, respectively. c is speed of light, and α is path loss exponent.

Denote $\beta_0 = \left(\frac{4\pi f_c}{c}\right)^{-\alpha}$ as the channel gain when the distance is set as 1 meter. Then we have:

$$\begin{aligned} \beta_{i,j}(t) = P^{\text{los}}_{i,j}(t) \frac{1}{\eta_1} \beta_0 \left(d_{i,j}(t)\right)^{-\alpha} + \left(1 - P^{\text{los}}_{i,j}(t)\right) \frac{1}{\eta_2} \times \\ \beta_0 \left(d_{i,j}(t)\right)^{-\alpha} \ , i \in \mathcal{N} \ , j \in \mathcal{M} \ , t \in \mathcal{T} \ . \end{aligned} \tag{15}$$

The achievable task offloading rate can be obtained as:

$$r_{i,j}(t) = B \log_2 \left(1 + \frac{P^{\text{tr}} \beta_{i,j}(t)}{\sigma^2}\right) \ , i \in \mathcal{N} \ , j \in \mathcal{M} \ , t \in \mathcal{T} \ . \tag{16}$$

where B is the subchannel bandwidth, P^{tr} is the transmitting power, and σ^2 is the noise power.

Then the task transmission duration can be obtained as:

$$T^{\text{tr}}_{i,j}(t) = \frac{D_i(t)}{r_{i,j}(t)} \ , i \in \mathcal{N} \ , j \in \mathcal{M} \ , 1 \le t \le T \ . \tag{17}$$

As for task execution at UAV j, denote $F^{\max}$ as its total CPU frequency, and $f^{\text{uav}}_{i,j}(t)$ as the number of CPU frequency allocated to device i at time slot t, we have:

$$\sum_{i=1}^{N} a_{i,j}(t) f^{\text{uav}}_{i,j}(t) \le F^{\max} \ , 1 \le j \le M \ , 1 \le t \le T \ . \tag{18}$$

Then the task execution duration can be obtained as:

$$T^{\text{comp}}_{i,j}(t) = \frac{F_i(t)}{f^{\text{uav}}_{i,j}(t)} \ , i \in \mathcal{N} \ , j \in \mathcal{M} \ , 1 \le t \le T \ . \tag{19}$$

D. Energy Consumption

Let $E_{i,j}(t)$ be the energy consumption at device i for processing task $S_i(t)$. Denote $E^{\text{loc}}_{i,j}(t)$ and $E^{\text{off}}_{i,j}(t)$ as the energy consumption of local computing and task offloading, respectively. We have:

$$\begin{aligned} E_{i,j}(t) = a_{i,j}(t) \left[sgn(j) E^{\text{off}}_{i,j}(t) + (1 - sgn(j)) \, E^{\text{loc}}_{i,j}(t) \right] \ , \\ 1 \le i \le N \ , 0 \le j \le M \ , 1 \le t \le T \ . \end{aligned} \tag{20}$$

As for local computing, denote k_i as the effective switched capacitance and μ_i is a positive constant, which is typically set to 3 [13], then we have:

$$E^{\text{loc}}_{i,j}(t) = k_i (f^{\text{loc}}_i(t))^{\mu_i} T^{\text{loc}}_{i,j}(t) \ , i \in \mathcal{N}, j = 0, 1 \le t \le T. \tag{21}$$

As for task offloading, the energy consumption at device i can be obtained as:

$$E^{\text{off}}_{i,j}(t) = P^{\text{tr}} T^{\text{tr}}_{i,j}(t) \ , i \in \mathcal{N} \ , j \in \mathcal{M} \ , 1 \le t \le T \ . \tag{22}$$

E. Age of Update

We employ the concept of age of information (AoI) [11] to quantify the timeliness of local updates obtained at each IoT device. Denote $A_i(t)$ as age of update at device i, which is defined as the elapsed time since the most recent task is executed either locally or remotely at UAV. Then we have:

$$\begin{aligned} A_i(t) = t + 1 - \max\left\{t' | a_{i,j}(t') = 1, 1 \le t' \le t\right\} \ , \\ i \in \mathcal{N} \ , j \in \mathcal{Z} \ , t \in \mathcal{T} \ . \end{aligned} \tag{23}$$

Based on constraints (4), $\sum_{j=0}^{M} a_{i,j}(t) = 0$ indicates that task $S_i(t)$ is not executed at time slot t. Then we have:

$$A_i(t+1) = \begin{cases} A_i(t) + 1 \ , & \sum_{j=0}^{M} a_{i,j}(t) = 0 \ ; \\ 1 \ , & \text{otherwise.} \end{cases} \tag{24}$$

To guarantee the freshness of local updates, the age of update at each device cannot exceed a threshold A_{th}. That is, as for device i, if $A_i(t) = A_{\text{th}}$ at time slot t , the first buffered task $S_i(t)$ has to be executed either locally ($a_{i,0}(t) = 1$) or remotely ($a_{i,j}(t) = 1, j \in \mathcal{M}$). Then we have:

$$(1 - a_{i,j}(t)) A_i(t) < A_{\text{th}}, 1 \le i \le N, j \in \mathcal{Z}, 1 \le t \le T. \tag{25}$$

F. Problem Formulation

In this study, we are interested in minimizing the energy consumption at IoT devices over T time slots. The problem can be formulated as follows:

$$\begin{array}{ll} \textbf{OPT-P} & \\ \min & \sum_{i=1}^{N}\sum_{j=0}^{M}\sum_{t=1}^{T} a_{i,j}(t)E_{i,j}(t) \\ \text{s.t} & \text{UAV trajectory: (1)–(3) ;} \\ & \text{Computation offloading: (4) – (19) ;} \\ & \text{Energy consumption: (20) – (22) ;} \\ & \text{Age of update: (23) – (25) .} \end{array}$$

In this formulation, $o_j(t)$ and $a_{i,j}(t)$ are discrete variables, $f_{i,j}^{\mathrm{uav}}(t)$ are continuous variables. It falls in the category of MINLP, which is intractable in general. The optimal solution can be obtained offline only if prior knowledge of all the system state information is known, which is impractical. To cope with the stochastic task generation at IoT devices and time-varying channel conditions, we propose a deep reinforcement learning based algorithm to facilitate fast decision-making.

III. DRL-BASED UAV PATH PLANNING AND COMPUTATION OFFLOADING ALGORITHM

In this section, we propose an online decision making approach for UAV-IoT MEC systems, in which at each time slot t, the moving direction of each UAV j ($o_j(t)$) and computation offloading decisions ($a_{i,j}(t)$ and $f_{i,j}^{\mathrm{uav}}(t)$) are optimized in order to minimize the energy consumption.

This can be achieved by transforming the formulated problem OPT-P into an MDP problem, which is defined by a tuple $\{S^c, A^c, T^c, R^c\}$, where S^c is the set of system states, A^c is set of movement actions, $T^c = \left\{p(s^{c\prime}|s^c, a^c)\right\}$ is the set of transition probabilities, and $R^c : S^c \times A^c \to R^c$ is a real-value reward function. A policy π is a mapping from S^c to A^c. Then the MDP problem is defined as follows.

1) **State**: At time slot t, the system state is defined the set of UAV coordinates, $s^c(t) = \left\{[x_j^u(t), y_j^u(t)], \forall j \in \mathcal{M}\right\}$.
2) **Action**: At time slot t, each UAV can move towards one direction, $a^c(t) = \{o_j(t), \forall j \in \mathcal{M}\}$.
3) **Reward**: To minimize the energy consumption, we employ the energy consumption at all devices as reward function $r^c(t)$, which is defined as:

$$r^c(t) = f\left(\sum_{i=1}^{N}\sum_{j=0}^{M} a_{i,j}(t)E_{i,j}(t)\right) - P\ . \tag{26}$$

where $f(\cdot)$ is the negative exponential function that acts as a normalization. The penalty P is to prevent UAVs from flying out of the given area.

Due to the stochastic task generation at IoT devices, the state transition probability ($p[s^c(t+1)|s^c(t), a^c(t)]$) is difficult to model. Since the action space and state space are discrete in the MDP problem, we employ double dueling deep Q-learning network(D3QN) [14] based approach to facilitate fast decision making, which divides the Q value into value of state and advantage of action to enable fast environment adaptability. Note that D3QN avoids overestimation of Q value by decoupling the action selection and Q value estimation, which further improves the learning performance.

The current network and the target network are parameterized by θ and θ^- respectively. Given a sampled mini-batch $\mathbb{B}$ (s^b, a^b, r^b, s^{b+1}), the target value y^b can be obtained as:

$$y^b = r^b + \gamma \max Q\left(s^{b+1}, \operatorname{argmax}_a Q\left(s^{b+1}, a; \theta\right); \theta^-\right)\ . \tag{27}$$

Then the mean square error $L(\theta)$ for current network parameter update can be obtained as follows:

$$L(\theta) = \frac{1}{|\mathbb{B}|}\left(y^b - Q\left(s^b, a^b; \theta\right)\right)^2\ . \tag{28}$$

Therefore, we can have the update formula for weights θ:

$$\theta = \theta + \frac{1}{|\mathbb{B}|}\rho\left[y^b - Q\left(s^b, a^b; \theta\right)\right]\nabla_\theta Q\left(s^b, a^b; \theta\right)\ . \tag{29}$$

where ∇_θ denotes the gradient with respect to θ. Note that the current network updates every step while the target network copy the parameters of current network every O steps.

As for computation offloading decisions, we design a low complexity solution based on one-dimensional search. More specifically, at each time slot, given the moving direction of each UAV j ($o_j(t)$), its coordinate can be calculated based on constraints (3). The set of IoT devices that are within UAV j's coverage area can be obtained by constraints (5)–(7). Due to the limited number of sub-channels, we select at most Q devices for task offloading by ranking the candidate devices according to the energy consumption of local computing and task offloading, based on constraints (8)–(22). Note that in the case when device i's age of update reaches the age threshold, it will perform local execution if it is not selected for task offloading or it is not covered by any UAV, as specified in constraints (23)–(25).

Complexity discussion: The computational complexity of deciding moving directions ($o_j(t)$) using a fully-connected layer is $\Theta(l)$, where l is the number of neurons. The complexity of computation offloading decision ($a_{i,j}(t)$ and $f_{i,j}^{\mathrm{uav}}(t)$) is $\Theta(MN)$. It yields an overall complexity of $\Theta\left(T\left(l + MN\right)\right)$.

IV. PERFORMANCE EVALUATION

In this section, we present simulation results to demonstrate the performance of our proposed solution. We consider a UAV-IoT MEC system where two UAVs are deployed to provide computing service to 50 IoT devices that are randomly distributed within a square area of 200 m $\times$ 200 m. As for computation tasks, the task size and required number of CPU cycles follow uniform distribution with $D_i(t) \in [10, 50]$ KB and $F_i(t) \in \left[10^9, 2\times 10^9\right]$ cycles/bit. The computation capability at each UAV is set as 1.2×10^{10} cycles/s. As for task offloading, the transmission power is set as 1000 mW, while the bandwidth is 1 MHz. As for learning parameters, the capacity of experience replay buffer is 100000. The neural network we employ includes one fully-connected hidden layer with 512 neurons. The remaining parameters are presented in Table I.

Algorithm 1 Proposed DRL-based Algorithm

```
1:  Initialize the replay memory $D$, the current network parameter $\theta$, the target network parameter $\theta^{-} = \theta$, candidate list $\mathbf{C}_j$ and offloading decision $\mathbf{A}_O$.
2:  for $episode = 1 : E$ do
3:      Obtain an initial state.
4:      for $t = 1 : T$ do
5:          Select a random action $a^c(t)$ at with probability $\epsilon$.
6:          Otherwise $a^c(t) = \text{argmax}_{a^c(t)} Q\left(s^c(t), a^c(t); \theta\right)$.
7:          Execute $a^c(t)$ and get $s^c(t+1)$.
8:          for UAV $j = 1, 2, ..., M$ do
9:              for IoT device $i = 1, 2, ..., N$ do
10:                 if constraints (5) is met then
11:                     Obtain $E^d_{i,j}(t) = E^{loc}_{i,j}(t) - E^{tr}_{i,j}(t)$ and store the index $i$ into $\mathbf{C}_j$ when $E^d_{i,j}(t) > 0$.
12:                 end if
13:             end for
14:             Sort $\mathbf{C}_j$ in descending order..
15:         end for
16:         repeat
17:             for UAV $j = 1, 2, ..., M$ do
18:                 Let $i$ be the first item in $\mathbf{C}_j$ and obtain $f^{\text{uav}}_{i,j}(t)$ according to constraints (9).
19:                 if (8) and (18) are met then
20:                     if $E^{tr}_{i,j}(t) < E^{tr}_{i,\mathbf{A}_O(i)}(t)$ or $\mathbf{A}_O(i) = 0$ then
21:                         Let $\mathbf{A}_O(i) = j$ and remove $i$ from $\mathbf{C}_j$.
22:                     end if
23:                 end if
24:             end for
25:         until The elements in $\mathbf{C}_j$ have been traversed.
26:         Obtain $\mathbf{A}_O$ and calculate reward according to (26).
27:         Store transition $(s^c(t), a^c(t), r^c(t), s^c(t+1))$.
28:         if replay memory buffer if full then
29:             Sample a random batch of transitions $\mathbb{B}$ from $D$.
30:             Calculate the target value $y^b$ according to (27).
31:         end if
32:         Update network according to (29).
33:     end for
34: end for
```

To demonstrate the performance benefit of our proposed solution, we choose random visit (Random), per-slot optimal (Per-slot), and DQN-based strategy (DQN) as the benchmarks. As for "Random", the moving direction of UAVs are randomly chosen with equal probability at each time slot. As for "Per-slot", prior knowledge of all the system state parameters is assumed to be available, the moving directions of UAVs are optimized to minimize the energy consumption of next time slot by traversing all possible combinations. As for "DQN", the trained network consists of the same number of layers and neurons as our proposed D3QN-based solution. Note that all three benchmark algorithms employ the same low-complexity offloading decision procedure as our proposed solution. Fig. 2 shows the trend of energy consumption at all IoT devices under four strategies during 60 time slots. As shown in the figure, "Random" algorithm has the worst performance, which yields the necessity of a rigorous UAV path design. "Per-slot" algorithm performs the best at the cost of prior knowledge of system states and high computation complexity of $\Theta\left(T\left(MN \times |\mathcal{O}|^M\right)\right)$. On the other hand, our proposed algorithm achieves a good energy efficiency with simple algebraic calculations, attributing to the trained dueling double deep Q-network. Moreover, since our D3QN-based solution can avoid overestimation of Q value with better environment adaptability, it performs better than "DQN" algorithm.

TABLE I: Parameters

Parameter	Value
Number of Time Slots T	60
Time slot duration τ	1 s
Noise power σ^2	-90 dB
UAV flying height H	40 m
Azimuth angle θ	$\pi/4$
Path loss exponent α	2
Excessive path loss coefficient in LoS η_1	3 dB [15]
Excessive path loss coefficient in NLoS η_2	23 dB [15]
Learning rate ρ	0.0001
Mini-batch size T	128
Reward decay rate δ	0.95
Optimizer	RMSProp
Activation function	ReLU

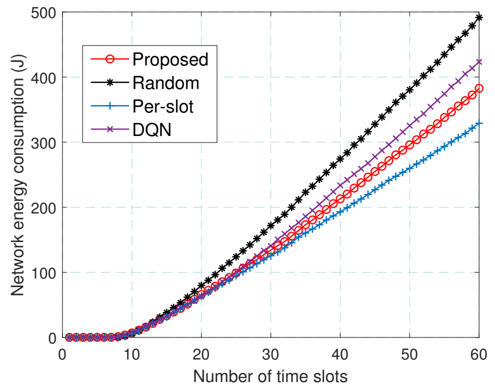


Fig. 2: Energy consumption under four algorithms.

Fig. 3 shows the trend of averaged age of updates under four strategies during 60 time slots. We set the age threshold to 23. As shown in the figure, "Random" algorithm keeps the averaged age barely below the threshold due to unplanned UAV trajectory. Compared with the myopic "Per-slot" algorithm, our proposed D3QN-based algorithm achieves a better performance due to the exploration feature of deep reinforcement learning. As shown in Fig. 4, it is easy to observe that under the trajectory obtained by our proposed solution, the UAVs are guided to collaboratively move around the area so that the IoT devices have a fair chance to offload their tasks, which improves the timeliness of local updates.

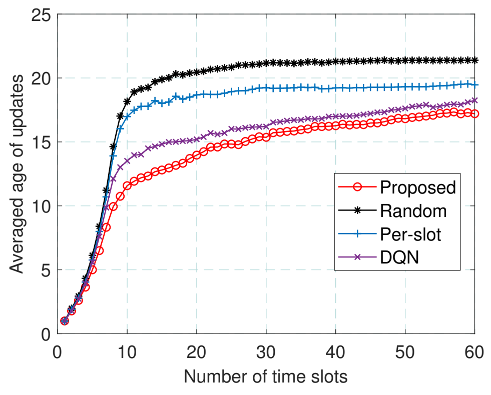
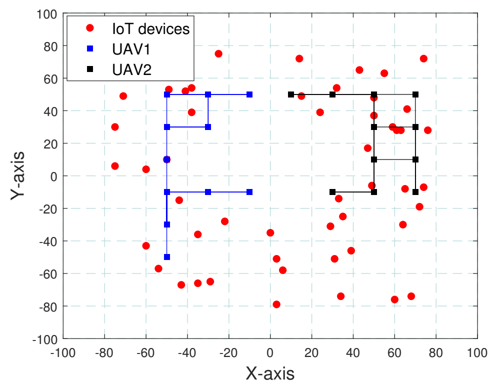
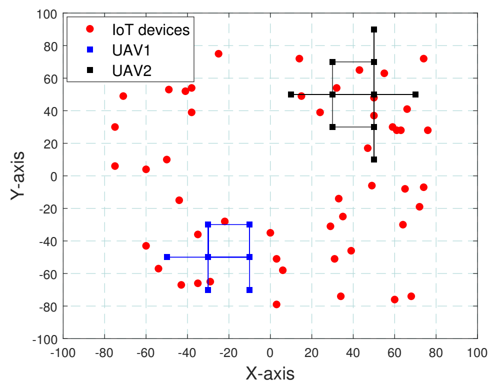


Fig. 3: Averaged age of update under four algorithms.



(a) Trajectory obtained by proposed algorithm



(b) Trajectory obtained by Per-slot algorithm

Fig. 4: Examples of UAV trajectory.

V. Conclusion

In this paper, we investigated an energy-aware path planning and computation offloading strategy for UAV-IoT MEC system, where UAVs provide computing service to IoT devices during flight. To ensure timeliness of updates, we employed the concept of age of information to quantify the freshness of updates. We formulated an energy minimization problem by jointly considering path planning for multiple UAVs, energy consumption of computation offloading, and age of updates. To achieve real-time decision making under stochastic computation task arrival, we proposed a solution procedure based on double dueling deep Q-learning network (D3QN). Simulation results demonstrated that the performance of our proposed strategy is competitive in terms of energy consumption and freshness of updates.

Acknowledgment

This paper is supported by National Key $R\&D$ Program of China under Grant 2018YFB1800800, 2001208 supported by NSFC, and National Natural Science Foundation for Young Scientists of China under Grant No.62001050.

References

[1] O. Bushnaq, A. Chaaban and T. Al-Naffouri, “The Role of UAV-IoT Networks in Future Wildfire Detection,” *IEEE Internet of Things Journal,* May. 2021.

[2] D. Yang, Q. Wu, Y. Zeng and R. Zhang, “Energy Tradeoff in Ground-to-UAV Communication via Trajectory Design,” *IEEE Transactions on Vehicular Technology,* vol. 67, no. 7, pp. 6721–6726, July. 2018.

[3] X. Yuan, T. Yang, Y. Hu, J. Xu and A. Schmeink, “Trajectory Design for UAV-Enabled Multiuser Wireless Power Transfer With Nonlinear Energy Harvesting,” *IEEE Transactions on Wireless Communications,* vol. 20, no. 2, pp. 1105–1121, Feb. 2021.

[4] Z. Li, M. Chen, C. Pan, N. Huang, Z. Yang and A. Nallanathan, “Joint Trajectory and Communication Design for Secure UAV Networks,” *IEEE Communications Letters,* vol. 23, no. 4, pp. 636–639, Apr. 2019.

[5] C. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, “Energy-Efficient UAV Control for Effective and Fair Communication Coverage: A Deep Reinforcement Learning Approach,” *IEEE Journal on Selected Areas in Communications,* vol. 36, no. 9, pp. 2059–2070, Sept. 2018.

[6] F. Jiang, K. Wang, L. Dong, C. Pan, W. Xu and K. Yang, “AI Driven Heterogeneous MEC System with UAV Assistance for Dynamic Environment: Challenges and Solutions,” *IEEE Network,* vol. 35, no. 1, pp. 400–408, Jan. 2021.

[7] H. Peng and X. Shen, “Multi-Agent Reinforcement Learning Based Resource Management in MEC- and UAV-Assisted Vehicular Networks,” *IEEE Journal on Selected Areas in Communications,* vol. 39, no. 1, pp. 131–141, Jan. 2021.

[8] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam and A. Nallanathan, “Deep Reinforcement Learning Based Dynamic Trajectory Control for UAV-assisted Mobile Edge Computing,” *IEEE Transactions on Mobile Computing,* Feb. 2021.

[9] M. Hua, L. Yang, C. Li, Q. Wu and A. Swindlehurst, “Throughput Maximization for UAV-Aided Backscatter Communication Networks,” *IEEE Transactions on Communications,* vol. 68, no. 2, pp. 1254–1270, Feb. 2020.

[10] Y. Zeng, X. Xu and R. Zhang, “Trajectory Design for Completion Time Minimization in UAV-Enabled Multicasting,” *IEEE Transactions on Wireless Communications,* vol. 17, no. 4, pp. 2233–2246, Apr. 2018.

[11] S. Kaul, R. Yates and M. Gruteser, “Real-time status: How often should one update?,” *2012 Proceedings IEEE INFOCOM,* Orlando, FL, USA, pp. 2731–2735, Mar. 2012.

[12] A. Al-Hourani, S. Kandeepan and S. Lardner, “Optimal LAP Altitude for Maximum Coverage,” in *IEEE Wireless Communications Letters,* vol. 3, no. 6, pp. 569–572, Dec. 2014.

[13] F. Jiang, K. Wang, L. Dong, C. Pan, W. Xu and K. Yang, “Deep-Learning-Based Joint Resource Scheduling Algorithms for Hybrid MEC Networks,” *IEEE Internet of Things Journal,* vol. 7, no. 7, pp. 6252–6265, July. 2020.

[14] Z. Wang, T .Schaul, M. Hessel, H. Van Hasselt, M. Lanctot and N. De Freitas, “Dueling Network Architectures for Deep Reinforcement Learning,” *Proceedings of The 33rd International Conference on Machine Learning, PMLR,* New York, NY, USA, pp. 1995–2003, June. 2016.

[15] M. Mozaffari, W. Saad, M. Bennis and M. Debbah, “Mobile Unmanned Aerial Vehicles (UAVs) for Energy-Efficient Internet of Things Communications,” in *IEEE Transactions on Wireless Communications,* vol. 16, no. 11, pp. 7574–7589, Nov. 2017.