Deep-Reinforcement-Learning-Based Computation Offloading in UAV-Assisted Vehicular Edge Computing Networks

Junjie Yan, Xiaohui Zhao, and Zan Li, *Member, IEEE*

***Abstract*—Vehicular edge computing (VEC) is considered to be a key technology to improve the processing efficiency of computing tasks for the Internet of Vehicles (IoV). Using roadside units (RSUs) distributed on both sides of a road as edge servers, computation-intensive and latency-sensitive in-vehicle tasks can be responded to quickly. However, some Quality of Service (QoS) is often difficult to ensure due to clogged dense urban buildings or lack of infrastructure in remote areas. In this article, we propose a software-defined network (SDN)-driven partial offloading model for unmanned aerial vehicle (UAV)-assisted VEC networks, where the RSUs and UAVs jointly provide computing services to the vehicles and collect global information through centralized control using an SDN controller. To guarantee these vehicles obtain computing results in time and rationally utilize computing resources, we develop an optimal offloading mechanism using Age of Information (AoI), together with energy consumption and rental price as a comprehensive weighted cost of our above optimization objective. The total system cost of the performing tasks is minimized by jointly optimizing the UAV trajectory, user association, and offloading decision. Considering the mobility of the vehicles and UAVs and the dynamic network environment, we design a deep reinforcement learning (DRL)-based joint trajectory control and offloading allocation algorithm (DRL-TCOA) to solve the proposed computation offloading problem. Experimental results show that the proposed DRL-TCOA algorithm maintains better information freshness and lower system cost than the other baseline offloading strategies.**

***Index Terms*—Age of Information (AoI), computation offloading, deep reinforcement learning (DRL), unmanned aerial vehicles (UAVs), vehicular edge computing (VEC).**

I. Introduction

WITH continuous development of Internet of Vehicles (IoV) technology, various emerging in-vehicle applications are becoming widely used in smart vehicles for autonomous driving and real-time map navigation [1], [2]. Due to limited computing resources in these vehicles, they cannot meet high-reliability and low-latency processing requirements of the applications mentioned above. Therefore, vehicular edge computing (VEC) has been considered as an effective means to solve the problems [3]. By deploying some servers at edge of radio access network, vehicle users can offload their computing tasks to nearby roadside units (RSUs) to relieve computation burdens of corresponding cellular networks and provide users with more efficient and reliable network services [4], [5], [6], [7], [8].

In practice, statically deployed edge servers still face several drawbacks, such as remote areas without infrastructure coverage or urban agglomeration areas with a high density of vehicles, which may not satisfy necessary computation requests of vehicle users. Fortunately, unmanned aerial vehicle (UAV)-enabled VEC can effectively solve these problems. With the advantages of easy deployment, high mobility, and strong Line-of-Sight (LoS) links, UAVs have been widely used both in military and civil applications as one of the important wireless communication network access methods [9], [10], [11]. Recently, UAV-enabled mobile-edge computing (MEC) architectures by integrating UAVs into MEC networks have been proposed [12] and extensively studied [13], [14], [15], [16], [17], [18], [19], [20], [21]. In this architecture, UAVs can be used as mobile-edge servers to provide additional computation and communication services for vehicle users to reduce heavy pressure on terrestrial computing and communication and to fill gaps in VEC network coverage. In addition, UAV-assisted VEC systems with more reliable LoS links can offer more flexible on-demand services and respond more quickly to users' requests [22].

Recently, extensive research has been conducted on how UAV-assisted VEC systems can minimize latency or energy consumption. Among them, for most intelligent in-vehicle applications that require real-time information processing, freshness of computing results is a key factor in improving Quality of Service (QoS). Since outdated information is somehow worthless or even harmful, guaranteeing timeliness of task execution is crucial for vehicle users to make right decisions. However, traditional latency performance metrics cannot adequately and accurately characterize the essential requirements for computing timeliness results. For example, if vehicle users generate or send tasks infrequently, even if the systems perform well in terms of latency, they will not be able to meet rigid requirements of real-time computing applications. Therefore, Age of Information (AoI) has been proposed [23] and widely studied [24], [25], [26], [27], [28], [29], [30] as a useful performance metric to measure information freshness. Moreover, since all computing services

Manuscript received 25 September 2023; revised 22 November 2023 and 11 January 2024; accepted 16 February 2024. Date of publication 27 February 2024; date of current version 23 May 2024. This work was supported by the National Natural Science Foundation of China under Grant 62171197. *(Corresponding author: Zan Li.)*

The authors are with the College of Communication Engineering, Jilin University, Changchun 130012, Jilin, China (e-mail: yanjj19@mails.jlu.edu.cn; xhzhao@jlu.edu.cn; zanli@jlu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2024.3370553

are currently paid, if a computing task needs to be offloaded, a relatively high corresponding fee has to be paid to an operator [21]. In this article, we focus on reducing AoI cost, energy consumption, and rental price for computation offloading in UAV-assisted VEC networks. Compared with single airborne or ground-based VEC paradigms, our proposed UAV-assisted VEC system offers more flexibility and comprehensive service resources.

However, we face some challenges. Due to the given high-speed mobile nature of UAVs and vehicles, and unknown complex wireless channel environment, all these time-varying network environment makes it difficult to rely on traditional optimization algorithms to solve our problem. Instead, we propose a solution using deep reinforcement learning (DRL) for the VEC offloading problem in this complex and dynamic environment. Importantly, different from the single offloading optimization, a DRL-based approach usually focuses on a long-term offloading performance which is key to the time-varying dynamic systems. Moreover, unlike the traditional Markov decision process (MDP)-based solutions, the DRL-based approaches can learn offloading strategies by directly interacting with environment without prior knowledge. Meanwhile, deep neural networks (DNNs) in DRL have strong perceptual computation capability and can support large state and action spaces to fulfill an optimization task.

Since state update of vehicle tasks is random and accurate information is not available a priori, it is impossible to derive an accurate analytical solution about AoI based on theoretical knowledge such as queuing theory. Moreover, due to the time-accumulative nature of AoI itself, UAV trajectory control and task offloading decisions are tightly coupled, which requires joint optimization. In consideration of the above difficulties, to solve the problem from our proposed networks. The main contributions are summarized as follows.

1) To make rational computation offloading decisions, we propose a software-defined network (SDN)-based optimization frame for UAV-assisted VEC networks. The SDN controller keeps information exchange by collecting global information and sending it to each vehicle. Unlike common UAV-assisted VEC networks, we consider a joint aerial (UAV) and ground (RSU) edge computing service, which supports partial computation offloading instead of a simple binary one. In particular, we consider the randomness of task arrivals by having task buffers at both the edge nodes (ENs) and vehicles, where some computing tasks are offloaded to the edge task queues and the rest are queued in the local computation queues. The system uses a first-come–first-served (FCFS) protocol, thus the computing tasks can be calculated in multiple time slots, which is more realistic.
2) To guarantee freshness of the computing results, we take an optimization metric AoI as a part of the total system cost. Then, we develop an optimal offloading mechanism by minimizing the weighted costs of AoI, energy consumption, and rental price. Without a probability distribution for the queuing system, we present a general system model in which all computing tasks are randomly generated. Since dynamic optimization of AoI within multiple time slots contradicts the time-accumulation property of AoI, we use Peak AoI (PAoI) to characterize information freshness. The results show that this PAoI contains enough information about the working environment state to satisfy the DRL optimization requirements.
3) Considering the highly dynamic nature of the proposed UAV-assisted VEC networks (UAV and vehicle locations, task information, queue backlog, and channel state), and the high degree of coupling between the optimization variables, we transform this optimization into a reinforcement learning problem based on the MDP and solve it by our proposed DRL-based joint trajectory control and offloading allocation (DRL-TCOA) algorithm in dynamic scenarios with continuous and large-scale action spaces. We use twin-delayed deep deterministic policy gradient (TD3) with a DRL-based actor–critic (AC) framework for the continuous actions to avoid overestimation of Q-value and increase training stability through the cooperation between the multiple actors and critical networks.

The remainder of this article is organized as follows. Section II introduces important related works. In Section III, we present the system model of the SDN-based UAV-assisted VEC network. An optimization problem formulation is illustrated in Section IV. Then, we propose our DRL-TCOA algorithm in Section V. Section VI shows the simulation results. Finally, Section VII concludes this article.

II. Related Work

Recently, UAV-assisted MEC has received a lot of attention, and how to develop an optimal offloading mechanism has become an important research topic. Usually, latency is the most fundamental performance metric for most delay-sensitive tasks. In particular, Wang et al. [13] and Wu et al. [14] considered additionally the queuing delay of MEC. In [13], the average delay under statistical significance is given based on the queuing theory. In [14], considering the task arrival randomness, the authors construct the computation queues and formulate the optimization problem as a constrained MDP (CMDP) to minimize a long-term service latency via the well-known Lyapunov stability theory and DRL. For the UAV-assisted MEC networks, the limited energy of devices and UAVs is also a necessarily considered issue when designing an offloading scheme, e.g., to minimize total system energy consumption by jointly optimizing UAV trajectories and resource allocation [15]. In order to make a tradeoff between the throughput and energy consumption, the authors maximize the energy efficiency (EE) as their optimization objective [16]. In another perspective, to offer the flexibility of QoS requirements, many current studies typically consider a weighted sum of delay and energy consumption [17], [18], [19], [20]. The work of [17] develops an offloading scheme with the objective of minimizing the weighted cost of energy consumption and delay to involve a joint optimization of task offloading and MEC server selection decisions, power allocation, UAV trajectory, and CPU frequency allocation. Similarly, Hoa et al. [19] minimized the energy consumption and latency of the system by optimizing the offloading task size at each node in a multihop edge computing system. On this basis, Zhao et al. [21] considered the paid cost for leasing computing resources. In order to allocate resources realistically and rationally, they solved a multiuser computation offloading sequential game problem with the goal of jointly optimizing execution time, energy consumption, and rental price. All the above works mainly concentrate on the optimization strategies designed for classical performance metrics, such as latency and energy consumption, but they do not pay attention to the requirements of vehicle users for freshness of computing tasks and timeliness of computing results.

For the UAV-assisted MEC networks, AoI has been introduced as an effective performance metric to measure information freshness. Qin et al. [24] studied an AoI minimization problem for the air-ground collaborative MEC to jointly optimize UAV trajectory, offloading scheduling, and computing resource allocation under some computing resource constraint and energy constraint of the UAV. In [25], a heterogeneous federated multiagent reinforcement learning algorithm has been proposed to jointly optimize the computation offloading, UAV trajectory and communication resources by regarding users, UAVs and cloud center as heterogeneous agents. Zhao et al. [26] maximized a long-term system throughput under the AoI constraint by transforming it into a queue stability constraint through the introduction of a virtual queue. The authors solve this problem separately by decoupling it into multiple subproblems using the Lyapunov stability theory. Similarly, under the AoI constraints, Chen et al. [27] proposed an algorithm based on a double dueling deep Q-learning network (D3QN) to optimize UAV trajectories to reduce network energy consumption. Some works have also investigated AoI as part of the optimization objective, considering tradeoffs with other performance metrics. In [28], the expressions of average AoI and EE are derived under three retransmission schemes, and then the average AoI–EE ratio is minimized to achieve a tradeoff between AoI and energy. Diao et al. [29] derived an analytical expression for the average PAoI to minimize the weighted sum of PAoI and average energy consumption by jointly optimizing some offloading parameters and UAV trajectory. In [30], the system cost of an in-home health monitoring system is determined by the trio of AoI, energy consumption, and medical criticality. Most of the above AoI-oriented works are still about the offloading scheduling problems, which principally take binary offloading strategies to optimize the system performance by choosing whether to offload remotely or compute locally. These require that the computation task needs to be completed within a single time slot, which is often difficult to guarantee in real problems.

III. System Model

This section presents a UAV-assisted MEC architecture in software-defined vehicular networks. As shown in Fig. 1, the overall system architecture consists of two parts: 1) the control plane and 2) the data plane, where the former is constructed using cellular communication and the latter using dedicated short-range communications (DSRCs) [31]. In the data plane, there are M deployed RSUs and a rotary-wing UAV as EN to provide computing services to N vehicles within their transmission range, and the set of the RSUs and the vehicles are denoted by $\mathcal{M} = \{1, 2, \ldots, m, \ldots, M\}$ and $\mathcal{N} = \{1, 2, \ldots, n, \ldots, N\}$, respectively. In addition, each EN is equipped with a computing server and an SDN switch to communicate with the SDN controller. We denote a new set $\mathcal{M}' = \{1, 2, \ldots, m, \ldots, M, M+1\}$ to represent the set of ENs, where $m = M + 1$ means that the edge server is UAV. The control plane consists mainly of the SDN controller and the DRL agent, connected via a wired backhaul link. The SDN controller forwards the collected global information (vehicle coordinate position, task information, etc.) to the DRL agent, which makes the offload decisions and sends them to each device through the controller. In addition, the main notations of this article are shown in Table I.

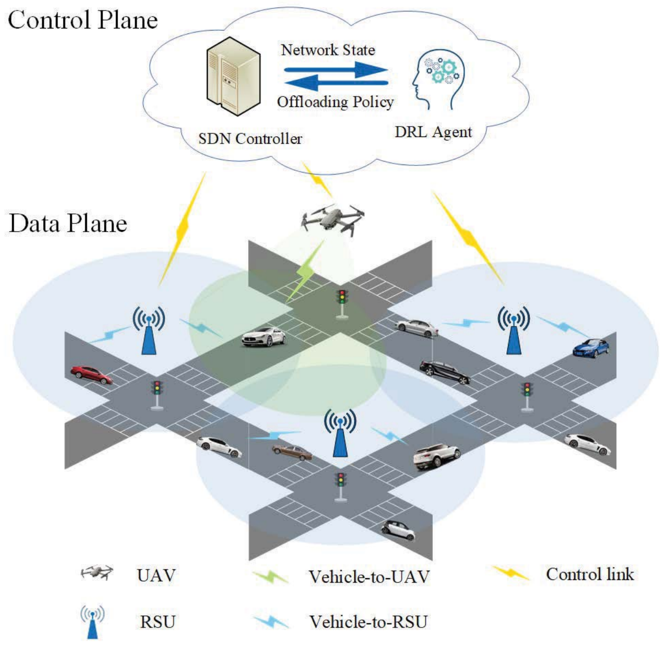


Fig. 1. SDN-based UAV-assisted VEC framework.

A. Communication Model

In the communication model, the vehicles transmit their partial computing tasks to be offloaded to ENs. We divide the finite time horizon into T time slots with length τ. Without loss of generality, we consider a 3-D Cartesian coordinate system in our model. We denote the coordinate of RSU m as $\mathbf{w}_m = (x_m, y_m, 0)$, the coordinate of vehicle n as $\mathbf{w}_n(t) = (x_n(t), y_n(t), 0)$, and the location of the UAV in time slot t is $\mathbf{w}_u(t) = (x_u(t), y_u(t), H)$. The flight direction of UAV is determined by the angle of $\theta_u(t) \in [0, 2\pi)$ and the flight velocity of $v_u(t) \in [0, v_{\max}]$. Thus, we have

$$x_u(t) = x_u(t-1) + \tau v_u(t)\cos(\theta_u(t)) \quad \forall t \in \mathcal{T} \tag{1}$$

$$y_u(t) = y_u(t-1) + \tau v_u(t)\sin(\theta_u(t)) \quad \forall t \in \mathcal{T}. \tag{2}$$

For this UAV-assisted VEC network, we consider a widely used air-to-ground (A2G) probabilistic path loss model in

TABLE I
LIST OF NOTATIONS

Notation	Description
$\mathbf{w}_m(t)$,$\mathbf{w}_n(t)$,$\mathbf{w}_u(t)$	The Location of RSU m, vehicle n, the UAV
$PL_{n,u}^{A2G}(t)$,$PL_{n,m}^{G2G}(t)$	The path loss of A2G channel, G2G channel
$\alpha_{n,m}(t)$	User association variable
$\beta_{n,m}(t)$	bandwidth resources allocation variable
N_M^{max}	Maximum number of vehicles supported for RSU
W_R, W_U	Total uplink bandwidth of RSU, UAV
$p_n(t)$	Transmission power of vehicle n
$R_{n,m}(t)$	Achieved transmission rate
$D_n(t)$	Input data size
$T_n^{max}(t)$	Maximum tolerant latency of task
X_c	Number of CPU cycles
$o_n(t)$	Computation offloading ratio
$Q_n(t)$,$Q_m(t)$	Backlogs of the tasks computation queues for vehicle n, EN m
F_n, F_R, F_U	Computation capacity of vehicle n, RSU, UAV
$D_n^{comp}(t)$,$D_m^{comp}(t)$	Data size of the tasks computed by vehicle n and EN m in time slot t
$\tau_n^{local}(t)$,$E_n^{local}(t)$	Local service latency, energy consumption
$\tau_n^{queue}(t)$,$\tau_n^{comp}(t)$	Local queuing time, computation time
$\tau_n^{off}(t)$,$E_{n,m}^{off}(t)$	Offloading service latency, energy consumption of vehicle n
$\tau_{n,m}^{trans}(t)$,$\tau_{n,m}^{queue}(t)$, $\tau_{n,m}^{comp}(t)$	Offloading transmission time, queuing time, computation time
$E_u^{comp}(t)$,$E_u^{propul}(t)$	UAV computation energy consumption, propulsion energy consumption

which there are LoS links and Non-LoS (NLoS) links for the communication between the UAV and the vehicles. According to [32], the path loss between the UAV and vehicle n can be given as

$$PL_{n,u}^{A2G}(t) = 20\log\left(\|\mathbf{w}_u\ (t) - \mathbf{w}_n(t)\|\frac{4\pi f}{C}\right) + P_{n,u}^{LoS}(t)\eta_{LoS} + \left(1 - P_{n,u}^{LoS}(t)\right)\eta_{NLoS} \quad (3)$$

where f is the carrier frequency, C is the speed of light, and η_{LoS} and η_{NLoS} are the mean additional path loss of LoS links and NLoS links, respectively. $P_{n,u}^{LoS}(t) = [1/(1 + a\exp(-b(\theta_{n,u}(t)) - a))]$ is the LoS link probability, a and b are the environment parameters, and $\theta_{n,u}(t) = \arcsin(H/\|\mathbf{w}_u\ (t) - \mathbf{w}_n(t)\|)$ is the elevation angle between the UAV and vehicle n in time slot t.

For the communication links between the RSU and the vehicles, the Rayleigh fading channels model is adopted. The path loss between RSU m and vehicle n in the tth slot can be expressed (in dB) as

$$PL_{n,m}^{G2G}(t) = 20\log\left(\|\mathbf{w}_m - \mathbf{w}_n(t)\|\frac{4\pi f}{C}\right) + \eta_{\text{Rayleigh}} \quad (4)$$

where η_{Rayleigh} is the Rayleigh fading coefficients of G2G channels.

In order to avoid transmission interference, we use the frequency-division multiple access (FDMA) protocol. We define a binary variable $\alpha_{n,m}(t) \in \{0, 1\}, m \in \mathcal{M}'$ to denote the user association, where $\alpha_{n,m}(t) = 1$ indicates that vehicle n is associated with and served by EN m in time slot t; otherwise, $\alpha_{n,m}(t) = 0$. We assume that each vehicle can only select one EN to offload its task and that each EN serves at most $N_M^{\max}$ vehicles in each time slot. Then, we have

$$\sum_{m=1}^{M+1} \alpha_{n,m}(t) = 1 \quad \forall n \in \mathcal{N}\ \forall t \in \mathcal{T} \quad (5)$$

$$\sum_{n\in\mathcal{N}} \alpha_{n,m}(t) \le N_M^{\max} \quad \forall m \in \mathcal{M}'\ \forall t \in \mathcal{T}. \quad (6)$$

In addition, we define another variable $\beta_{n,m}(t) \in [0, 1]$ to denote the ratio of bandwidth resources allocated by EN $m, m \in \mathcal{M}'$ to vehicle n in time slot t. For the sake of simplicity, we assume $\beta_{n,m}(t) = [1/(\sum_{n\in\mathcal{N}} \alpha_{n,m}(t))]$, i.e., all vehicles that need to offload to EN m and will share the bandwidth of EN m equally.

According to the Shannon theorem, the achieved transmission rate between vehicle n and EN m in time slot t can be described as

$$R_{n,m}(t) = \beta_{n,m}(t)W_M \log_2\left(1 + \left(\frac{p_n(t)h_{n,m}(t)}{\sigma_{n,m}^2(t)}\right)\right) \quad (7)$$

where $W_M = W_R$ (or W_U) is the total uplink bandwidth of each RSU (or UAV), $p_n(t)$ is the transmit power of vehicle n, $\sigma_{n,m}^2(t) = \beta_{n,m}(t)N_0W_M$ denotes the average power of the additive white Gaussian noise (AWGN) at EN m, and N_0 is the AWGN spectral density. $h_{n,m}(t) = 10^{(-PL_{n,m}(t)/10)}$, where $PL_{n,m}(t) = PL_{n,m}^{G2G}(t), m \in \mathcal{M}$ (or $PL_{n,m}(t) = PL_{n,m}^{A2G}(t), m = M + 1$).

B. Computation Model

In the proposed UAV-assisted VEC computation model, the partial offloading strategy is adopted, which means that tasks can be offloaded to RSUs or the UAV at any ratio. In time slot t, vehicle n can randomly generate computing task $U_n(t)$, and the task arrival rate follows the Poisson distribution with parameter λ_u. Specifically, $U_n(t)$ can be described as $U_n(t) = \left(D_n(t), T_n^{\max}(t)\right)$, where $D_n(t)$ (in bit) denotes the input data size, and $T_n^{\max}(t)$ (in second) denotes the maximum tolerant latency of task $U_n(t)$. Let $o_n(t) \in [0, 1]$ denote the computation offloading ratio of $U_n(t)$, which means that vehicle n offloads $o_n(t)$ part of $U_n(t)$ to its associated EN m for remote computation, and the rest part $(1 - o_n(t))$ is executed locally. Let $Q_n(t)$ and $Q_m(t)$ be the backlogs of the tasks computation queues for vehicle $n \in \mathcal{N}$ and EN $m \in \mathcal{M}'$ used to cache the local computing tasks of vehicle n and the offloading tasks of EN m, respectively. Therefore, $Q_n(t)$ and $Q_m(t)$ evolve over different time slots as

$$Q_n(t+1) = \max\left\{Q_n(t) - D_n^{\text{comp}}(t), 0\right\} + (1 - o_n(t))D_n(t) \quad (8)$$

$$Q_m(t+1) = \max\left\{Q_m(t) - D_m^{\text{comp}}(t), 0\right\} + \sum_{n\in\mathcal{N}} \alpha_{n,m}(t)o_n(t)D_n(t) \quad (9)$$

where $D_n^{\text{comp}}(t) = \min\{\tau F_n/X_c, Q_n(t)\}$ and $D_m^{\text{comp}}(t) = \min\{\tau F_m/X_c, Q_m(t)\}$ are the data size of the tasks computed by vehicle n and EN m in time slot t, respectively. F_n represents the local computation capacity of vehicle n (in units of CPU cycles per second), $F_m = F_R$ (or F_U) is the computation capability of EN m (in CPU cycles per second), X_c denotes the number of CPU cycles required to compute 1 bit. Then, we give the service latency and energy consumption for the two cases of the local computation and computation offloading, respectively.

1) Local Computation: For $U_n(t)$ with the local computation ratio $(1 - o_{n,m}(t))$, the local service latency includes the queuing time and computation time calculated by

$$\tau_n^{\text{local}}(t) = \tau_n^{\text{queue}}(t) + \tau_n^{\text{comp}}(t) \tag{10}$$

where

$$\tau_n^{\text{queue}}(t) = \frac{X_c Q_n(t)}{F_n} \tag{11}$$

$$\tau_n^{\text{comp}}(t) = \frac{(1 - o_n(t)) X_c D_n(t)}{F_n}. \tag{12}$$

According to the widely adopted power consumption model as $k_n(F_n)^{v_n}$ as in [33], [34], [35], and [36], where $k_n \geq 0$ is the effective capacitance coefficient dependent on the processor chip architecture, and v_n is typically set to 3. Thus, the computation energy consumption of vehicle n in time slot t can be calculated by

$$\begin{aligned} E_n^{\text{local}}(t) &= k_n(F_n)^3 \tau_n^{\text{comp}}(t) \\ &= k_n(F_n)^2 (1 - o_n(t)) X_c D_n(t). \end{aligned} \tag{13}$$

2) Computation Offloading: If $\alpha_{n,m}(t) = 1, m \in \mathcal{M}'$, $o_n(t)$ part of the task $U_n(t)$ will be offloaded to EN m. The offloading service latency for this part is the sum of the transmission time, queuing time, and computation time calculated by

$$\tau_n^{\text{off}}(t) = \tau_{n,m}^{\text{trans}}(t) + \tau_{n,m}^{\text{queue}}(t) + \tau_{n,m}^{\text{comp}}(t) \tag{14}$$

where

$$\tau_{n,m}^{\text{trans}}(t) = \frac{o_n(t) D_n(t)}{R_{n,m}(t)} \tag{15}$$

$$\tau_{n,m}^{\text{queue}}(t) = \frac{X_c Q_m(t)}{F_m} \tag{16}$$

$$\tau_{n,m}^{\text{comp}}(t) = \frac{o_n(t) X_c D_n(t)}{F_m}. \tag{17}$$

Note that the time for EN m to return the results to vehicle n can be ignored since the size of the computed results is negligible compared to that of the original task data. Accordingly, the offloading energy consumed by vehicle n can be calculated as

$$E_{n,m}^{\text{off}}(t) = p_n(t) \tau_{n,m}^{\text{trans}}(t). \tag{18}$$

C. UAV Energy Consumption Model

1) Computation Energy Consumption: Similarly, the computation energy consumption of the UAV in time slot t can be expressed as

$$E_u^{\text{comp}}(t) = k_u(F_U)^3 \tau_{n,m}^{\text{comp}}(t), m = M + 1. \tag{19}$$

2) Propulsion Energy Consumption: From [24] and [37], the propulsion energy consumption of rotary-wing UAV depends on its flight velocity and can be given by

$$\begin{aligned} E_u^{\text{propul}}(t) = \tau \Bigg(& P_0 \left(1 + \frac{3\|v_u(t)\|^2}{U_{\text{tip}}^2} \right) \\ & + P_i \left(\sqrt{1 + \frac{\|v_u(t)\|^4}{4v_0^4}} - \frac{\|v_u(t)\|^2}{2v_0{}^2} \right)^{\frac{1}{2}} \\ & + \frac{1}{2} d_0 \rho_0 s_0 A_0 \|v_u(t)\|^3 \Bigg) \end{aligned} \tag{20}$$

where $v_u(t) = [(\|\mathbf{w}_u(t+1) - \mathbf{w}_u(t)\|)/\tau]$ represents the UAV velocity. P_0 and P_i are the blade profile power and induced power in hovering status, respectively. U_{tip} is the tip speed of the rotor blade, v_0 is the mean rotor-induced velocity in hovering status, d_0 is the fuselage drag ratio, ρ_0 is the air density, s_0 is the rotor solidity, and A_0 is the rotor disc area. When the UAV is hovering, i.e., when the flight speed is 0, the hover energy consumption can be obtained as $E_u^h = \tau(P_0 + P_i)$.

Therefore, the total energy consumption of the UAV in time slot t is given by $E_u(t) = E_u^{\text{comp}}(t) + E_u^{\text{propul}}(t)$.

IV. Problem Formulation

In this section, we first introduce a system payoff function to measure the system cost for computing tasks. We then formulate our multislot dynamic joint trajectory control and offloading allocation optimization problem in the proposed UAV-assisted VEC network to minimize the total system cost.

A. Payoff Function

The main factors to determine the total cost of the system is the AoI cost, energy consumption, and rental price, which will be derived in detail below.

1) AoI Cost: AoI is a destination node-centric metric for measuring the freshness of information in a network [38]. In this UAV-assisted VEC network, we denote $\Delta_n(t)$ as our AoI of vehicle n in time slot t, which is defined as the elapsed time between current time t and generation time of the latest results received by vehicle n, i.e.,

$$\Delta_n(t) = t - g_n(t) \tag{21}$$

where $g_n(t)$ is the task generation time of the latest computing results received by vehicle n before time slot t.

Fig. 2 shows the evolution of $\Delta_n(t)$ for vehicle n under the FCFS principle. For task $u \in \mathcal{U}_n(t)$, the generation time is denoted as $t_{n,u}$, and the computation completion time is denoted as $t'_{n,u}$. Let $T_{n,u} = t'_{n,u} - t_{n,u}$ and $Y_{n,u} = t_{n,u+1} - t_{n,u}$ denote the execution time of task u and the interval generation time. Thus, $g_n(t)$ can be described as $g_n(\tau) = t_{n,\hat{U}_n(\tau)}$, where $\hat{U}_n(\tau) = \max\{u | t'_{n,u} \leq \tau\}$ is the index of the latest received task results.

As shown in Fig. 2, for each vehicle, the corresponding AoI is accumulated linearly over time until a task is computed, and when the AoI drops to a smaller value that reflects the time between the generation and reception of that task. Specifically,

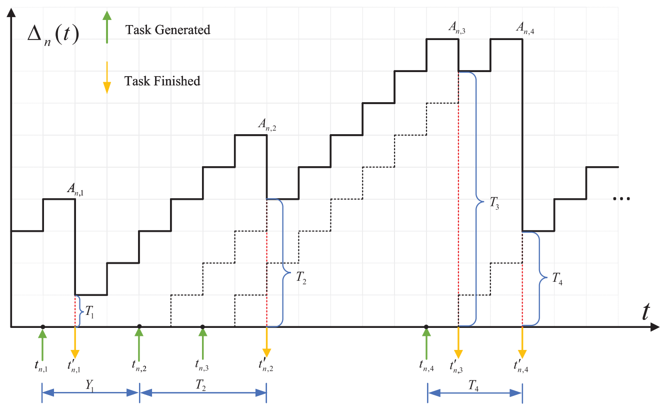


Fig. 2. AoI evolution in UAV-assisted VEC.

when a task is finished, the AoI is reset to the processing latency for this task; otherwise, the AoI grows at a fixed rate (i.e., 1). The evolution of $\Delta_n(t)$ can be updated as

$$\Delta_n(t+1) = \begin{cases} T_{n,u}+1, & \text{if} \quad t = t'_{n,u} \\ \Delta_n(t)+1, & \text{otherwise.} \end{cases} \tag{22}$$

In addition, we take a PAoI as the maximum age of the vehicle before it receives offloading results [39]. The PAoI of task u is given by

$$A_{n,u} = \begin{cases} \Delta_n(0) + T_{n,1}, & \text{if} \quad u = 1 \\ Y_{n,u-1} + T_{n,u}, & \text{if} \quad u > 1 \end{cases} \tag{23}$$

where $\Delta_n(0)$ is the initial age of vehicle n.

For the proposed computation model, since the partial offloading scheme is used and the task is computed in parallel on the ENs and vehicles, the total execution latency to complete task u can be expressed by $T_{n,u} = \max\{\tau^{\text{local}}_{n,u}, \tau^{\text{off}}_{n,u}\}$. Assuming $\Delta_n(0) = 0$, the PAoI for vehicle n is obtained by

$$A^{\text{total}}_n(t) = \max\left\{\tau^{\text{local}}_n(t), \tau^{\text{off}}_n(t)\right\} + Y_n(t) \tag{24}$$

where $Y_n(t)$ is the time interval for the task to arrive at vehicle n.

2) Energy Consumption: For the partial offloading scheme, the total energy consumption to execute tasks can be expressed as the sum of two components, local computation energy consumption, and computation offloading energy consumption precisely, i.e.,

$$E^{\text{total}}_n(t) = E^{\text{local}}_n(t) + \alpha_{n,m}(t)E^{\text{off}}_{n,m}(t). \tag{25}$$

3) Rental Price: We need to consider the cost of renting computing resources from the EN server. Since the RSUs and the UAVs have different computation capabilities, they offer different prices for their services. A unit price (in pence per CPU cycle) of computing resources rented by the RSUs and the UAV is denoted by $P_m = P_R, m \in \mathcal{M}$ or $P_m = P_U, m = M+1$. Thus, the total amount price paid by vehicle n to EN m is

$$P^{\text{total}}_n(t) = P_m o_n(t) X_c D_n(t). \tag{26}$$

Therefore, the payoff function of vehicle n can be modeled as a linear function of the above three costs as

$$C^{\text{total}}_n(t) = \gamma_A A^{\text{total}}_n(t) + \gamma_E E^{\text{total}}_n(t) + \gamma_P P^{\text{total}}_n(t) \tag{27}$$

where γ_A, γ_E, and γ_P denote the weight coefficients of the AoI cost, energy consumption, and rental price, respectively, and $\gamma_A + \gamma_E + \gamma_P = 1$.

B. Problem Formulation

Our objective aims to optimize UAV trajectory, user association, and offloading allocation in order to minimize long-term costs for N vehicles.

We define two sets $\mathbf{A}(t) = \{\alpha_{n,m}(t)\}_{n\in\mathcal{N},m\in\mathcal{M}'}$ and $\mathbf{O}(t) = \{o_n(t)\}_{n\in\mathcal{N}}$. Then, the total cost minimization problem can be formulated as

$$\textbf{OP1}: \min_{\mathbf{w}_u(t),\mathbf{A}(t),\mathbf{O}(t)} \sum_{t\in\mathcal{T}}\sum_{n\in\mathcal{N}} C^{\text{total}}_n(t) \tag{28}$$

$$\text{s.t.} \ \|\mathbf{w}_n(t+1) - \mathbf{w}_n(t)\| = v_n(t)\tau \quad \forall n \in \mathcal{N}\ \forall t \in \mathcal{T} \tag{28a}$$

$$\|\mathbf{w}_u(t+1) - \mathbf{w}_u(t)\| \le v_{\max}\tau \quad \forall t \in \mathcal{T} \tag{28b}$$

$$v_u(t) \in [0, v_{\max}], \theta_u(t) \in [0, 2\pi) \quad \forall t \in \mathcal{T} \tag{28c}$$

$$\alpha_{n,m}(t) \in \{0,1\} \quad \forall m \in \mathcal{M}' \quad \forall n \in \mathcal{N}\ \forall t \in \mathcal{T} \tag{28d}$$

$$\sum_{m=1}^{M+1} \alpha_{n,m}(t) = 1 \quad \forall n \in \mathcal{N}\ \forall t \in \mathcal{T} \tag{28e}$$

$$\sum_{n\in\mathcal{N}} \alpha_{n,m}(t) \le N^{\max}_M \quad \forall m \in \mathcal{M}\ \forall t \in \mathcal{T} \tag{28f}$$

$$\sum_{n\in\mathcal{N}} \alpha_{n,m}(t) \le N^{\max}_U, m = M+1 \quad \forall t \in \mathcal{T} \tag{28g}$$

$$o_n(t) \in [0,1] \quad \forall n \in \mathcal{N}\ \forall t \in \mathcal{T} \tag{28h}$$

$$\max\left\{\tau^{\text{local}}_n(t), \tau^{\text{off}}_n(t)\right\} \le T^{\max}_n(t) \quad \forall n \in \mathcal{N}\ \forall t \in \mathcal{T} \tag{28i}$$

$$\sum_{t\in\mathcal{T}} E_u(t) \le E^{\max}_U \tag{28j}$$

where (28a) describes the trajectory of a vehicle traveling at a random speed $v_n(t)$. Constraints (28b) and (28c) guarantee that the UAV flight speed is under the designed maximum flight speed. Constraints (28d)–(28g) describe the user association variable. Under these constraints, each vehicle can only select one EN per time slot to offload, and the number of vehicles served by each EN cannot exceed a maximum number of associations. Constraint (28h) guarantees that every task can be processed locally or offloaded to EN. Constraint (28i) is the maximum latency constraint. Constraint (28j) limits the UAV's energy consumption to no more than its available energy consumption, where $E^{\max}_U$ represents the maximum energy stored by UAV.

Note that (28) is a nonsmooth and nonconvex function due to the nondifferentiable function $\max\{\cdot\}$ in the objective function, and $\mathbf{A}(t)$ is the set with binary discrete variables, which makes the feasible set of the optimization problem nonconvex set. Since **OP1** is a mixed-integer nonlinear programming (MINLP) NP-hard problem, it is hard to obtain a closed-form optimal solution by traditional methods.

Normally, the wireless communication environment of a UAV-assisted VEC network is unknown and highly dynamic due to random generation of onboard tasks and inherent fast-moving nature of vehicles and UAV. Thus, we choose a DRL approach to find an optimal flight and offloading solution for the complex nonconvex problem with multidimensional optimization variables coupled in uncertain large-scale networks.

V. DRL-Based Algorithm Design

In this section, a DRL-TCOA algorithm is proposed to solve **OP1**.

A. MDP Formulation

MDP is a standardized framework for solving sequential decision problems [40]. Because of the dynamic evolutionary nature of buffer queues in vehicles and ENs, the continuity of the UAV flight trajectories, the different states of channel gains, the vehicle locations, and the computing tasks at each time slot in the considered networks satisfy the Markov property, we transform **OP1** into a model-free MDP in consideration of the changing unknown environment where the state transition probabilities are difficult to obtain.

A model-free MDP normally consists of a 3-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R})$ where $\mathcal{S}$ denotes the system state space, $\mathcal{A}$ denotes the action space, and $\mathcal{R}$ denotes the reward function, respectively. Precisely, in time slot t, in our system the DRL agent collects global information through the SDN controller as the current environment state is denoted as $s_t \in \mathcal{S}$. The agent selects an action a_t from $\mathcal{A}$ according to a policy $\pi(a_t|s_t)$, interacts with the environment, and receives an immediate reward $r_t(s_t, a_t) \in \mathcal{R}$. Then, the environment state transits to next state $s_{t+1} \in \mathcal{S}$. Based on the obtained rewards, the agent continuously improves its policy and learns iteratively through constant interaction with the environment until an optimal policy is obtained. For **OP1**, the state space, action space, and immediate reward function of the system are defined as follows.

1) State Space $\mathcal{S}$: At the beginning of each time slot, the SDN controller forwards the global dynamic information collected periodically to the agent from the UAV-assisted VEC network, which can be considered as the state of the environment observed by the agent. We define the state at time slot t as

$$\mathbf{s_t} = \left\{\mathbf{W}(t), \mathbf{U}(t), \mathbf{Y}(t), \mathbf{Q}(t), \mathbf{H}(t), E_U^{\text{res}}(t)\right\} \tag{29}$$

where $\mathbf{W}(t)$, $\mathbf{U}(t)$, $\mathbf{Y}(t)$, $\mathbf{Q}(t)$, and $\mathbf{H}(t)$ are defined as follows.

1) $\mathbf{W}(t) = [\mathbf{w}_1(t), \mathbf{w}_2(t), \ldots, \mathbf{w}_N(t), \mathbf{w}_u(t)]$ is the set of real-time coordinates of all the vehicles and UAV in time slot t. Since the altitude of the UAV remains constant during its flight, we only need to consider their horizontal coordinates, thus $\mathbf{W}(t)$ is a vector of $2(N+1)$.
2) $\mathbf{U}(t) = [U_1(t), U_2(t), \ldots, U_N(t)]$ is the information of all the tasks generated in time slot t, where $U_n(t) = (D_n(t), T_n^{\max}(t))$ and $\mathbf{U}(t)$ is a vector of $2N$. If vehicle n does not generate a task in time slot t, $U_n(t) = 0$.
3) $\mathbf{Y}(t) = [Y_1(t), Y_2(t), \ldots, Y_N(t)]$ is an N-dimensional vector containing the time interval for all the vehicles to generate tasks.
4) $\mathbf{Q}(t) = [B_1(t), B_2(t), \ldots, B_N(t), Q_1(t), Q_2(t), \ldots, Q_M(t), Q_{M+1}(t)]$ is an $N+M+1$-dimensional vector containing the backlog of computation queue for all vehicles and ENs.
5) $\mathbf{H}(t) = [h_{1,1}(t), \ldots, h_{n,m}(t), \ldots, h_{N,M+1}(t)]$ is an $N(M+1)$-dimensional vector containing the channel gain between the vehicles and ENs.
6) $E_u^{\text{res}}(t)$ is the residual energy of UAV in time slot t, updated according to $E_u^{\text{res}}(t+1) = \max\{E_u^{\text{res}}(t) - E_u(t), 0\}$, and $E_u^{\text{res}}(0) = E_U^{\max}$.

2) Action Space $\mathcal{A}$: In time slot t, the DRN agent makes a joint action a_t based on the observed system state s_t containing the following three actions.

1) *UAV Trajectory $\mathbf{w}_u(t)$:* As mentioned above, the UAV controls its flight trajectory by adjusting its speed and angle, i.e., $\{v_u(t), \theta_u(t)\}$.
2) *User Association $\mathbf{A}(t)$:* The adopted DRL algorithm is based on a continuous action space. Since $\alpha_{n,m}(t) \in \{0, 1\}$ is discrete variable, we transform $\alpha_{n,m}(t)$ into a continuous variable $\alpha_n(t) \in [0, 1]$, named the offloading allocation variable which satisfies

$$\alpha_{n,m}(t) = \begin{cases} 1, & \text{if } \mathbb{1}_{\{d_{n,m} \le \frac{L}{M}\}}\alpha_n(t) \in (\frac{1}{3}, \frac{2}{3}], m \in \mathcal{M} \\ & \text{or } \alpha_n(t) \in \left(\frac{2}{3}, 1\right), m = M+1 \\ 0, & \text{otherwise} \end{cases} \tag{30}$$

where $\mathbb{1}_{\{\cdot\}}$ is the indicator function which is equal to 1 when the condition $\{\cdot\}$ is satisfied and 0 otherwise. The transformed variable $\alpha_n(t)$ ensures the action space continuity of the DRL algorithm while satisfying the constraint that each vehicle is only associated with one EN in each time slot. When $\alpha_n(t) \in [0, (1/3)]$, the corresponding vehicle chooses to process tasks locally, and $\{\alpha_{n,m}(t)\}_{m \in \mathcal{M}'}$ is equal to 0; when $\alpha_n(t) \in ((1/3), (2/3)]$, the corresponding vehicle chooses to offload tasks to RSU m on the current road L_m, and $\alpha_{n,m}(t) = 1, m \in \mathcal{M}$; and $\alpha_n(t) \in ((2/3), 1]$ means the vehicle chooses to offload to the UAV, $\alpha_{n,m}(t) = 1, m = M+1$.
3) *Task Offloading Decision $\mathbf{O}(t)$:* The offloading ratio $o_n(t)$ is selected from the continuous space $[0, 1]$. To reduce the dimensionality of the action space, we define

$$o_n(t) = \begin{cases} 3\alpha_n(t) - \lfloor 3\alpha_n(t) \rfloor, & \text{if } \alpha_n(t) \in \left(\frac{1}{3}, \frac{2}{3}\right) \cup \left(\frac{2}{3}, 1\right) \\ 1, & \text{if } \alpha_n(t) \in \left\{\frac{2}{3}, 1\right\} \\ 0, & \text{otherwise.} \end{cases} \tag{31}$$

Therefore, the action a_t can be defined only by the trajectory control variables and the offloading allocation variables expressed as

$$\mathbf{a_t} = \{v_u(t), \theta_u(t), \alpha_1(t), \ldots, \alpha_n(t), \ldots, \alpha_N(t)\}. \tag{32}$$

3) Immediate Reward Function $\mathcal{R}$: In order to minimize the total system cost, we define an immediate reward function such that it is negatively related to the cost function. In addition, to satisfy (28e), (28f), and (28h) in **OP1**, we introduce two penalty terms. Then, the immediate reward function is defined as

$$r_t = -\sum_{n \in \mathcal{N}} C_n^{\text{total}}(t) - \phi_1(t) - \sum_{n \in \mathcal{N}} \phi_{2n}(t) - \phi_3(t) \tag{33}$$

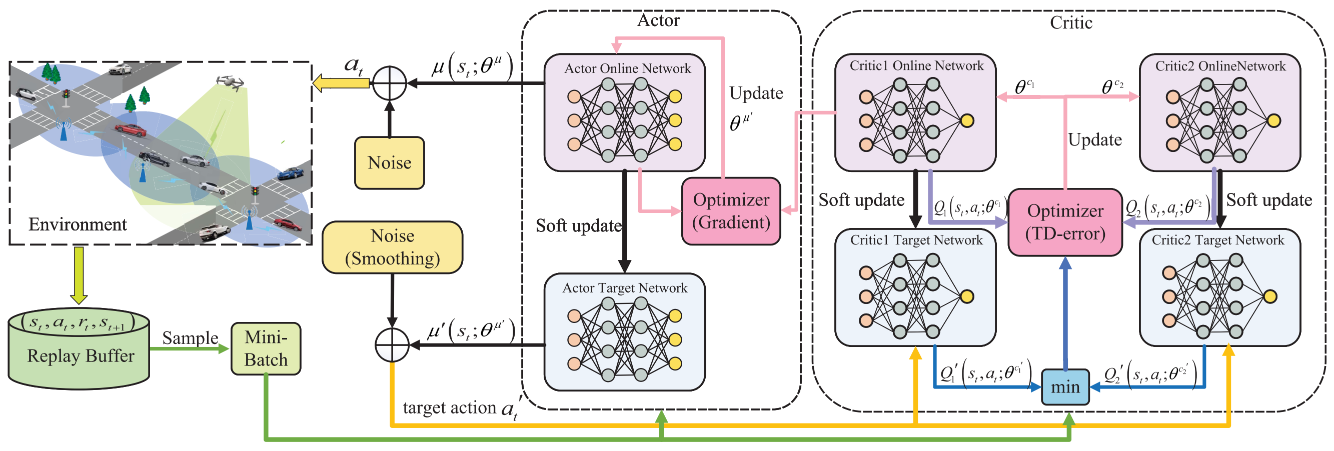


Fig. 3. Framework of the TD3-based DRL-TCOA algorithm.

where

$$\phi_1(t) = \begin{cases} c_1, & \text{if (28f) or (28g) does not satisfy} \\ 0, & \text{otherwise} \end{cases} \tag{34}$$

$$\phi_{2n}(t) = \begin{cases} c_2, & \max\{\tau_n^{\text{local}}(t), \tau_n^{\text{off}}(t)\} \geq T_n^{\max}(t) \\ 0, & \text{otherwise} \end{cases} \tag{35}$$

$$\phi_3(t) = \begin{cases} c_3, & E_u^{\text{res}}(t) < 0 \\ 0, & \text{otherwise} \end{cases} \tag{36}$$

where $c_1, c_2, c_3 > 0$ are the penalty constants.

The objective of the MDP is to find the optimal policy $\pi : \mathcal{S} \to \mathcal{A}$ for maximizing the expected long-term discounted cumulative reward, i.e.,

$$\max_{\pi} \lim_{k\to\infty} \mathbb{E}_\pi \left[\sum_{t=0}^{k} \zeta^t r_t \right] \tag{37}$$

where $\mathbb{E}_\pi(\cdot)$ denotes the expected value obtained following the policy π. $\zeta \in [0, 1]$ is a discount factor to capture the effect of expected future rewards on current state. When ζ is close to 0, it means that the agent only focuses on immediate benefits and does not consider future gains; when ζ is close to 1, it means that the long-term rewards are equally important.

Therefore, based on the above MDP model, we can reformulate **OP1** as

$$\textbf{OP2:} \quad \max_{v_u(t), \{\alpha_n(t)\}_{n\in\mathcal{N}}} \frac{1}{T} \sum_{t=1}^{T} \zeta^t r_t. \tag{38}$$

OP2 is an unconstrained model-free MDP problem that can be solved using our proposed DRL-based algorithm described in the following.

B. TD3-Based DRL-TCOA

In the proposed DRL-TCOA algorithm, we take the TD3 algorithm [41] to train our agent for optimal UAV trajectory control and offloading allocation, since it can quickly, stably, and accurately solve sequential decision problems with high-dimensional state space, continuous action space environment.

The overall framework of DRL-TCOA algorithm is depicted in Fig. 3. This algorithm consists of a set of actor networks μ and two sets of critic networks Q_1, Q_2, each has an online network and its corresponding target network. The online and target networks have the same structure but different parameters, defined as θ^μ, $\theta^{\mu'}$, θ^{c_1}, θ^{c_2}, $\theta^{c_1'}$, and $\theta^{c_2'}$, respectively.

At the beginning of each episode, we randomly initialize the locations of the vehicles and UAV, the queue backlogs of the vehicles and ENs, the onboard UAV energy, and the channel states. Then, we initialize the constructed six subnetworks and the replay buffer $\mathcal{B}$. In each time slot t, every vehicle, RSU, and UAV share their states with the agent based on the global information collected by the SDN controller, then the online actor network outputs the control action, i.e.,

$$a_t = \mu\left(s_t; \theta^\mu\right) + \epsilon \tag{39}$$

where $\epsilon \sim \mathcal{N}(0, \sigma)$ is the Gaussian noise to encourage exploration. According to the received command, the corresponding UAV flies to position $\mathbf{w}_u(t)$, the vehicle offloads the task to the corresponding EN, then the system state transits from s_t to next state s_{t+1} and receives the corresponding immediate reward r_t given by (33).

The experience (s_t, a_t, r_t, s_{t+1}) is stored in $\mathcal{B}$ to provide the training samples for the actor and critic network updates. When the stored transitions in $\mathcal{B}$ reach a certain number, a mini-batch of tuples is randomly selected from $\mathcal{B}$ to update the neural network parameters. Specifically, the twin online critic networks update the network parameters by minimizing the following loss function:

$$L\left(\theta^{c_j}\right) = |\mathcal{B}|^{-1} \sum_{i\in\mathcal{B}} \left(y_i - Q_j\left(s_i, a_i; \theta^{c_j}\right)\right), j \in \{1, 2\} \tag{40}$$

where $y_i = r_i + \zeta \min_{j=1,2} Q_j'(s_{i+1}, a_{i+1}'; \theta^{c_j'})$ is our Q-target, Clipped Double Q-learning technology is utilized to reduce potential overestimation bias by choosing two smaller Q-values. It is worth noting that $a_{t+1}' = \mu'(s_{t+1}; \theta^{\mu'}) + \epsilon', \epsilon' \sim \mathit{clip}(\mathcal{N}(0, \sigma), -c, c)$, where $\mathit{clip}(\cdot)$ is the upper and lower bound clipping function that trims the Gaussian random noise ϵ' between $(-c, c)$.

TD3 uses a delayed update mechanism, i.e., the actor network is updated after the kth update of the critic networks to ensure the stability of the policy updates. Moreover, the

Algorithm 1 TD3-Based DRL-TCOA Algorithm

```
Input: UAV-assisted VEC system environment
Output: Optimal trajectory control and offloading allocation
1: Initialize actor network $\mu$ and critic networks $Q_1$ and $Q_2$ with random parameters $\theta^{\mu}$, $\theta^{c_1}$, $\theta^{c_2}$.
2: Initialize target networks $\theta^{\mu'} \leftarrow \theta^{\mu}$, $\theta^{c'_1} \leftarrow \theta^{c_1}$, $\theta^{c'_2} \leftarrow \theta^{c_2}$.
3: Initialize replay buffer $\mathcal{B}$, mini-batch size $|N_b|$, discount factor $\zeta$, learning rates $\xi^{\mu}$, $\xi^{c_1}$, $\xi^{c_2}$, updated rate $\psi$, maximum episodes $N_{episodes}$ and maximum steps $N_{steps}$.
4: for episode $= 1$ to $N_{episodes}$ do
5:   Randomly initialize system environment and observe the initial state $s_0$.
6:   for $t = 1$ to $N_{steps}$ do
7:     Select action with exploration noise:
       $a_t = \mu(s_t; \theta^{\mu}) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$
8:     Obtain immediate reward $r_t$ and next state $s_{t+1}$.
9:     Store experience tuple $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{B}$.
10:    if $\mathcal{B}$ is full then
11:      Randomly sample a mini-batch of $N_b$ tuples $(s_i, a_i, r_i, s_{i+1})$ from $\mathcal{B}$.
12:      Compute target action: $a'_{t+1} = \mu'(s_{t+1}; \theta^{\mu'}) + \epsilon', \epsilon' \sim clip(\mathcal{N}(0, \sigma), -c, c)$.
13:      Update target $Q$ − value:$y_i = r_i + \zeta \min_{j=1,2} Q'_j(s_{i+1}, a'_{i+1}; \theta^{c'_j})$.
14:      Calculate the TD-error by (40), and update critic online networks $\theta^{c_1}$ and $\theta^{c_2}$).
15:      if mod $(t, 2) = 0$ then
16:        Update actor online network by (41).
17:        Soft update target networks by (42) and (43).
18:      end if
19:    end if
20:  end for
21: end for
```

online actor network updates the parameters by deterministic policy gradients given by

$$\nabla_{\theta^{\mu}} J(\theta^{\mu}) = |\mathcal{B}|^{-1} \sum_{i \in \mathcal{B}} \nabla_a Q_1(s_i, a_i; \theta^{c_1}) \nabla_{\theta^{\mu}} \mu(s_i; \theta^{\mu}). \quad (41)$$

In addition, for the target networks, a soft update is used to gradually track the learning online networks. The parameters of the target networks $\theta^{\mu'}$, $\theta^{c'_1}$, and $\theta^{c'_2}$ are updated by

$$\theta^{\mu'} \leftarrow \psi \theta^{\mu} + (1 - \psi) \theta^{\mu'} \quad (42)$$

$$\theta^{c'_j} \leftarrow \psi \theta^{c_j} + (1 - \psi) \theta^{c'_j}, j \in \{1, 2\} \quad (43)$$

where $\psi \in (0, 1]$ is the updated rate of the target networks.

Algorithm 1 summarizes the training process of the TD3-based joint trajectory control and offloading allocation algorithm, the agent continuously updates the network parameters according to (40)–(43) until the trajectory control and offloading allocation algorithm converges.

C. Algorithm Complexity Analysis

Typically, for a DNN with fully connected layers, the total computational complexity depends on its network size. It is assumed that the actor network and two critic network have layers L_a, $L_{c,1}$, and $L_{c,2}$, respectively. In the lth layer, the actor network and two critic networks have z_l^a, $z_l^{c,1}$, and $z_l^{c,2}$ neurons, respectively. Therefore, the total complexity of the proposed Algorithm 1 is about $\mathcal{O}(\sum_{l=0}^{L_a-1} z_l^a z_{l+1}^a + \sum_{l=0}^{L_{c,1}-1} z_l^{c,1} z_{l+1}^{c,1} + \sum_{l=0}^{L_{c,2}-1} z_l^{c,2} z_{l+1}^{c,2})$.

TABLE II
HYPERPARAMETERS OF USED TD3

Parameters	Value
Discount factor, ζ	0.999
Learning rate of actor network, ξ^{μ}	0.005
Learning rate of critic network, ξ^{c}	0.005
Updated rate of target network, ψ	0.005
Mini-batch size, $\lvert N_b \rvert$	32
Delay update frequency of actor networks, k	2
Variance of explore noise, policy noise, σ	0.1

VI. NUMERICAL RESULTS

In this section, we provide numerical simulation results to evaluate the performance of the proposed DRL-TCOA algorithm in the UAV-assisted VEC network.

A. Simulation Parameter Settings

Our simulations are conducted on a computer configured with an Intel i7-12700k CPU and an NVIDIA RTX 3080 GPU with 12 GB of video memory, and all the neural networks are trained in Python 3.9.12 and Tensorflow-gpu 2.9.1. The proposed TD3-based DRL-TCOA algorithm contains six identical neural networks, each consisting of two fully connected hidden layers with [600, 400] neurons. The hidden layers of networks use the rectified linear unit (ReLU) as the activation function, and the output layers of the actor networks use the Tanh in order to restrict the range of output action values to $[-1, 1]$. The specific hyperparameters of the TD3 networks are summarized in Table II.

To ensure the diversity of traffic feature data, we conducted experiments in different simulated road scenarios, i.e., the three maps shown in Fig. 4. Map 1 shows a 600-m two-way lane with three RSUs uniformly deployed, where each RSU has a coverage radius of 100 m. Map 2 shows a crossroad with an area of 400 m $\times$ 400 m and two RSUs are deployed at $(100, 100)$ and $(300, 300)$. Map 3 shows an area consisting of one 600-m east–west lane and two 400-m north–south lanes with three RSUs deployed at $(100, 100)$, $(300, 300)$, and $(500, 100)$. All the vehicles are randomly distributed on this roadway and travel at a random speed of 10–20 m/s. The duration of each time slot is set to 0.05 s. At the beginning of each time slot, the vehicles randomly generate computation-intensive tasks with task arrival rates set to 0.2, 0.4, 0.6, 0.8, and 1.0, respectively. The precise parameters of the UAV-assisted VEC network are given in Table III.

B. Convergence Performance

To verify the convergence of the proposed DRL-TCOA algorithm, we conduct experiments with the total number of vehicles of 5, 10, and 15, respectively, as shown in

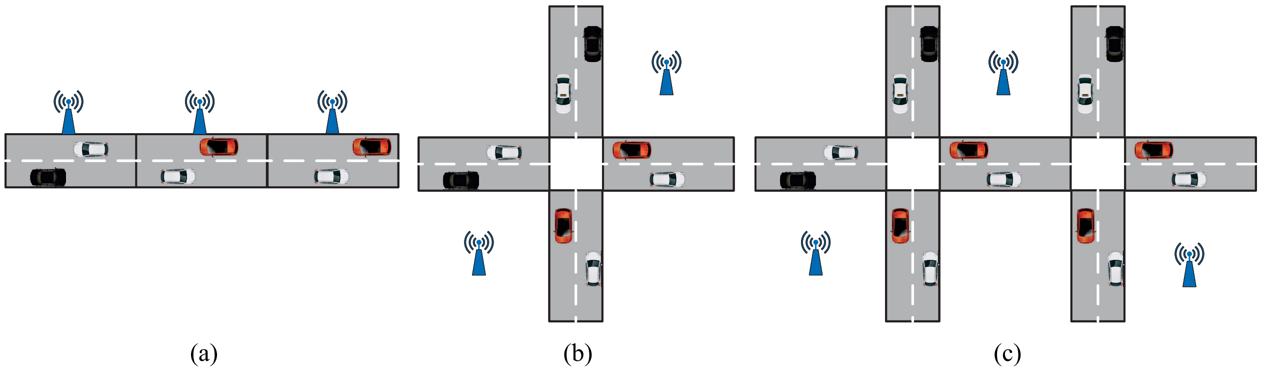


Fig. 4. Simulated scenarios. (a) Map1: 200 m×3. (b) Map2: 400 m×400 m. (c) Map3: 600 m×400 m.

TABLE III
VEC NETWORK PARAMETERS

Parameters	Description	Value
$D_n(t)$	Data size	[500, 800]KB
X_c	Number of CPU cycles	1000cycles/bit
T_n^{max}	Maximum tolerant latency of task	0.5s
a, b	A2G channel coefficient	10, 0.4
$\eta_{Rayleigh}$	Rayleigh fading coefficient	0.707
f	Carrier frequency	1950 MHz
C	Speed of light	$3 \times 10^8 m/s$
η_{LoS}, η_{NLoS}	Additional path loss	1, 20
W_R, W_U	Total uplink bandwidth	20, 50MHz
$p_n(t)$	Transmission power of vehicles	10mW
N_0	AWGN power spectral density	-174dBm/Hz
F_n, F_R, F_U	Computation capacity	0.5, 2, 5GHz
k_n	Computation energy efficiency coefficient	10^{-27}
p_R, p_U	Rental unit price	3×10^{-8}, 5×10^{-8}/cycles
P_0,P_i	Blade profile power and induced power in hovering status	79.8563, 88.6279w
U_{tip}	Tip speed of rotor blade	120m/s
v_0	Mean rotor-induced velocity in hovering status	4.03m/s
d_0, ρ_0	Fuselage drag ratio, air density	$0.6m^2, 1.225kg/m^2$
s_0, A_0	Rotor solidity, rotor disc area	$0.05, 0.503m^2$

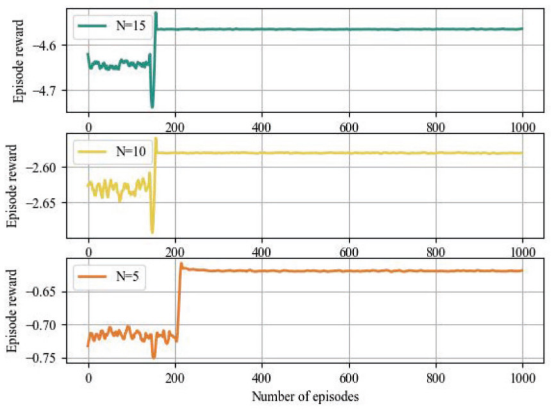
Fig. 5. We set the number of training episodes to 1000, and each episode contains 100 steps. Fig. 5(a) shows the average episode reward. We observe that in the initial training stage, the rewards fluctuate drastically around lower values, because the agent does not have enough knowledge about the environment in this stage, and it randomly takes actions to make choices. As the training proceeds, the rewards rise rapidly when enough data samples are accumulated and the DRL networks obtain a relatively high and stable reward at about 200 episodes. In addition, as the number of vehicles N increases, the system cost increases accordingly, but the convergence speed is not significantly affected. Fig. 5(b) demonstrates the loss curves during the training period by comparing them with those DQN and DDPG algorithms of $N = 10$ to show the convergence of DRL-TCOA algorithm in different N. From the experimental results, it can be seen that the DRL-TCOA algorithm converges quickly in about 200 episodes under different numbers of vehicle users. The DDPG algorithm converges slightly lower in about 400 episodes, while the convergence speed of the DQN algorithm is very low in about 2000 episodes, due to the fact that this algorithm learns inefficiently the environment in high-dimensional action space. In Fig. 5(c), we investigate the effect of four different combinations of learning rates within a range of 0.0001–0.01 on the DRL-TCOA algorithm, which is an important hyperparameter that affects the convergence of a learning algorithm. It shows that too high a learning rate may lead to an unstable training process and even unacceptable divergence. On the contrary, a too small learning rate may result in slower convergence and a local optimum. We observe that the convergence speed accelerates with increasing learning rate in this range, and a relatively high reward is obtained when $\xi^{\mu} = 0.005$ and $\xi^{c} = 0.005$. Therefore, we set the learning rates to 0.005 in the subsequent experiments.

C. Performance Evaluation

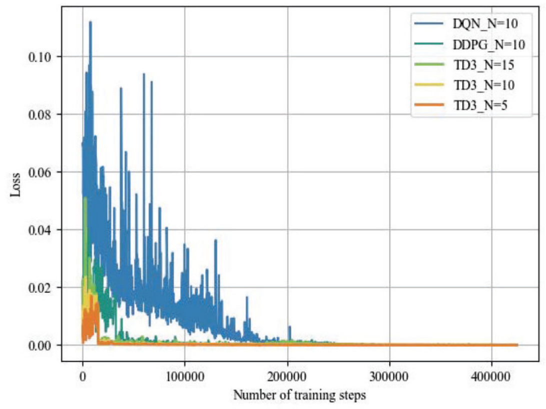
In this section, we evaluate the performance of the proposed DRL-TCOA algorithm under initial conditions with different dimensions and compare it with the following four baseline strategies to verify its effectiveness in obtaining optimal trajectory control and offloading allocation.

1) *DDPG-Based Joint Trajectory Control and Offloading Allocation (DDPG-TCOA)*: This is an alternative scheme based on the DDPG algorithm to replace the TD3 algorithm for joint optimization.
2) *DQN-Based Binary Offloading Allocation (DQN-BOA):* In each time slot t, the agent chooses a flight direction for the UAV from $\{-1, 0, 1\}$ and N offloading decisions for the vehicles from $\{0, 1, 2\}$. Thus, the dimension of the action space is 3×3^N.
3) *Local Execution (LE):* All the computing tasks are executed locally.
4) *Full Offloading (FO):* All the computing tasks are fully offloaded to the RSUs for execution.
5) *Random Offloading (RO):* Computing tasks are randomly selected to be executed locally, offloaded to the RSUs, or to the UAV.

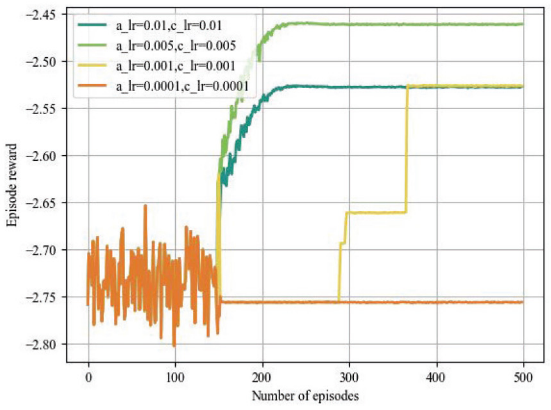
First, we conduct experiments in the three maps illustrated in Fig. 4, where the numbers of vehicles are set to $N = 10$. Fig. 6 presents the total system cost comparison of different strategies. We can observe that the system cost incurred by our proposed DRL-TCOA algorithm is always smaller than



(a)



(b)



(c)

Fig. 5. Convergence performances of the DRL-TCOA algorithm. (a) Episode reward. (b) Loss with different N. (c) Episode reward versus learning rate.

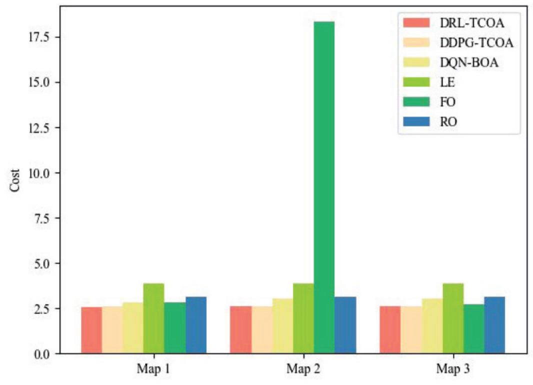


Fig. 6. Cost with different strategies and maps.

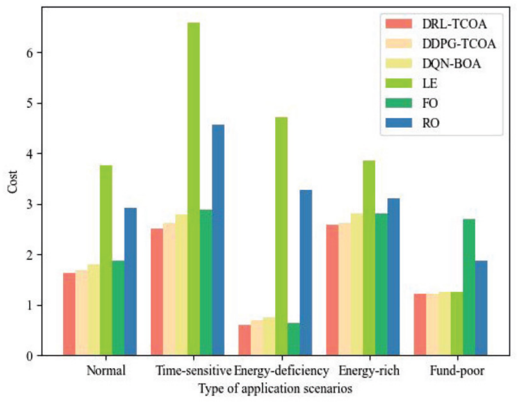


Fig. 7. Cost with different strategies and tasks.

the other strategies in three different maps, which verifies the superiority and generalizability of our algorithm. It is also clear from Fig. 6 that the costs of the other strategies do not change much across the three maps, except for FO strategy in Map 2, where the costs increase dramatically. This is because in Map 2, if all ten vehicle users offload their tasks to the RSUs, it will far exceed the computation capacity of two RSUs, the computation latency will increase dramatically, and the system cost will increase dramatically. The other maps are not affected by this, and in subsequent experiments we will not select Map 2 as the experimental scenario in order to scrutinize the effects of other parameters on the system performance.

The effectiveness of the DRL-TCOA algorithm is verified with the system average cost in different application scenarios as shown in Fig. 7. For the normal scenario, the three subcosts are weighted equally, i.e., $\lambda_A = 0.33, \lambda_E = 0.33$, and $\lambda_P = 0.33$. Compared with LE, FO, and RO strategies, the DRL-TCOA algorithm reduces the total system cost by 56.6%, 12.4%, and 44.0%, respectively. For the time-sensitive scenario, the real-time information needs to be updated frequently, requiring us to pay more attention to the AoI cost, thus we set $\lambda_A = 0.8, \lambda_E = 0.1$, and $\lambda_P = 0.1$. The DRL-TCOA algorithm reduces the system cost to 62% compared with that of LE and the cost differences become wider compared with those of rest strategies due to the fact that the computation capability of the edge servers is greater than that of the vehicles themselves. For the energy-deficiency scenario, we focus on the system energy consumption to demonstrate that the system operates properly, and we set $\lambda_A = 0.1, \lambda_E = 0.8$, and $\lambda_P = 0.1$. Since the energy consumption of the local computation is larger than the transmission energy consumption, the cost difference between the five compared algorithms widens and the corresponding cost reduction of the DRL-TCOA algorithm reaches 87.2% compared with that of the LE strategy. Similarly, for the fund-poor scenario, we set $\lambda_A = 0.1, \lambda_E = 0.1$, and $\lambda_P = 0.8$. In this case, the DRL-TCOA algorithm and the LE strategy produce less cost than other strategies due to the cost

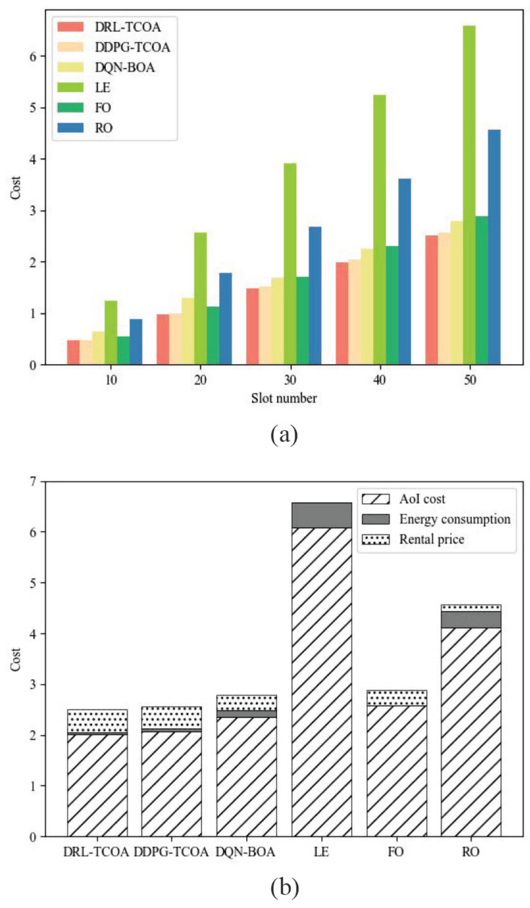


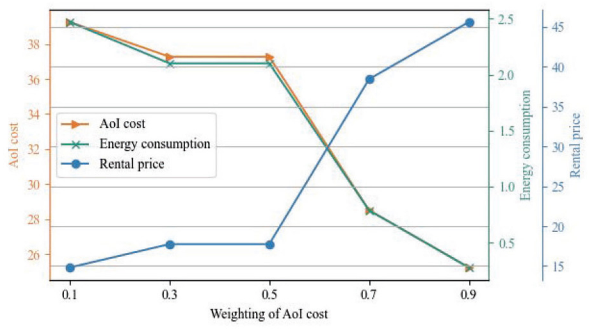
Fig. 8. Cost under different strategies and cost component. (a) Cost in different slot number. (b) Percentage of different cost component.

minimization of the DRL-TCOA algorithm and no-pay rent for the local computing in the LE strategy, and the cost of the RO strategy remains between the lowest and highest. In particular, for the normal traveling vehicle, the onboard energy supply is generally sufficient without much consideration of the energy cost, however reducing the AoI cost and rental price is more important. Thus, for this energy-rich scenario, we set $\lambda_A = 0.45$, $\lambda_E = 0.1$, and $\lambda_P = 0.45$, where the cost differences between strategies are not as large as in the other scenarios due to the tradeoffs between the AoI cost and the considered rental price. In this case, but the DRL-TCOA algorithm still reduces the costs by 33.0%, 7.8%, and 17.2% compared to those of LE, FO, and RO strategies. We can see that our proposed DRL-TCOA algorithm performs the best in different scenarios, which verifies that our algorithm outperforms the other traditional computation offloading strategies.

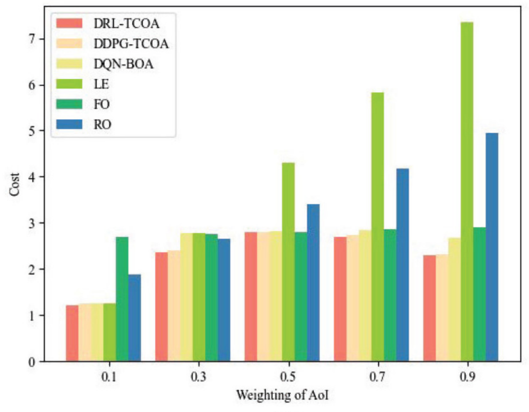
Fig. 8 illustrates further the cost optimization in different strategies for the time-sensitive scenario. From Fig. 8(a), it can be seen that the system cost increases with the number of time slots. The average system cost obtained by the DRL-TCOA algorithm is always lower than those of the other strategies. The less better performance of the DDPG-TCOA strategy compared to the DRL-TCOA algorithm is due to the fact that DRL-TCOA has two critic networks, which can choose the smaller of two Q-values to eliminate the overestimation bias. Compared with the DRL-TCOA algorithm and the DDPG-TCOA algorithm, the DQN-BOA strategy performs slightly worse, because the action space of the DRL-TCOA algorithm is continuous, which makes vehicles be offloaded to ENs at any ratio, and the UAV choose an optimal flight speed within the speed constraint. While the DQN-BOA strategy only selects a finite number of actions optimized in discrete intervals. In addition, it also shows that the average cost of the DRL-TCOA algorithm decreases significantly as the number of time slots increases. Because this algorithm uses the long-term cumulative cost as the optimization goal, rather than focusing only on timely reporting, which means that the agent can learn to predict future state and make its optimal decisions. Fig. 8(b) presents three cost components of the system cost under different strategies with $T = 50$. The results show that for the time-sensitive tasks, the DRL-TCOA algorithm not only optimizes the total system cost but also reduces the AoI cost by 66.9%, 22.0%, and 50.9% compared to LE, FO, and RO strategies, respectively. Besides, since the transmission energy consumption is much smaller than the local computation energy consumption, the FO strategy has extremely low energy consumption and LE strategy has the highest energy consumption, while opposite for the rental price. The cost for the rental component of the DRL-TCOA algorithm is higher than that of other strategies due to the fact that renting a UAV server is more expensive than RSU servers.

1) Impact of Weights: Fig. 9 illustrates the tradeoff among the AoI cost, energy consumption, rental price, and the system cost with different weights on AoI. From the previous section, we know that $\lambda_A + \lambda_E + \lambda_P = 1$. We fix λ_E to 0.1, and we change λ_A from 0.1 to 0.9 with an increment of 0.2, which means λ_P decreases accordingly. As shown in Fig. 9(a), with the increase of λ_A, the AoI cost decreases while the rental price increases. This is because the smaller λ_A corresponds to the fund-poor scenario in Fig. 7 where the agent is mainly concerned with reducing the rental price. However, larger λ_A corresponds to the time-sensitive scenario where the agent focuses on reducing the AoI cost. It is interesting to note that the trends for AoI cost and energy consumption are nearly identical, due to the approximately positive correlation between the lower AoI and transmission energy consumption of offloading computation compared to those of the local computing. In addition, it illustrates the complex relationships between the AoI, energy consumption, and rental prices that are difficult to optimize simultaneously.

In order to evaluate the effect of the weighting factor on the effectiveness of the DRL-TCOA algorithm, we compare the average system cost of different strategies under different λ_A. In Fig. 9(b), we can clearly see that the system cost of the DRL-TCOA algorithm is lower than those of the compared baseline strategies regardless of λ_A. Moreover, the system costs of the DRL-TCOA algorithm, the DDPG-TCOA strategy, and the DQN-BOA strategy are relatively low when λ_A is very large or small, which implies that there should be a tradeoff in the three cost components. The system cost of the LE strategy increases approximately linearly with the increase of



(a)



(b)

Fig. 9. AoI weight impact on cost. (a) Tradeoff among AoI cost, energy consumption, and rental price. (b) Cost under different strategies and weight on AoI.

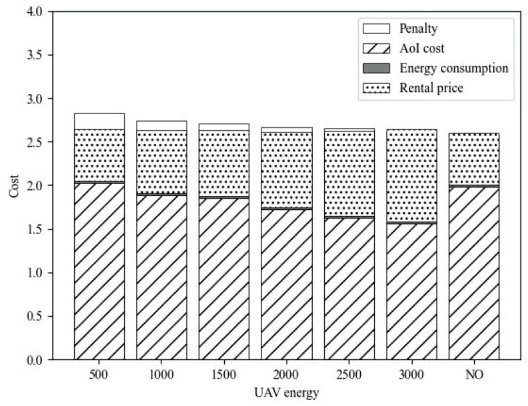


Fig. 10. Impact of UAV energy on cost.

λ_A, because there is no rent to pay for the local computation. The total system cost is simply a weighted sum of the AoI cost and the computing energy consumption, which means the increasing weights will augment the cost. In contrast, the cost of the FO strategy does not fluctuate much with varying λ_A, while that of the RO strategy is in between. In the later experiments, we will investigate energy-rich scenarios to simulate normally functioning vehicular networks, i.e., $\lambda_A = 0.45$, $\lambda_E = 0.1$, and $\lambda_P = 0.45$.

2) Impact of UAV Energy: Fig. 10 illustrates the impact of the UAV maximum available energy on system cost.

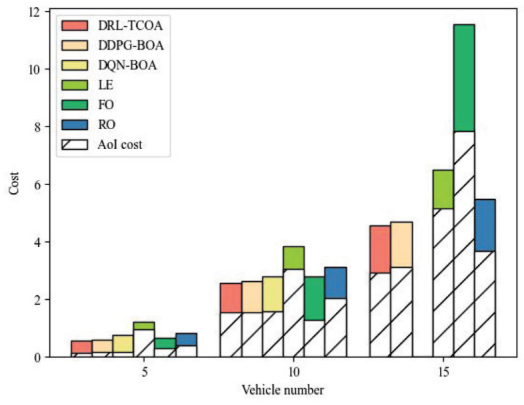
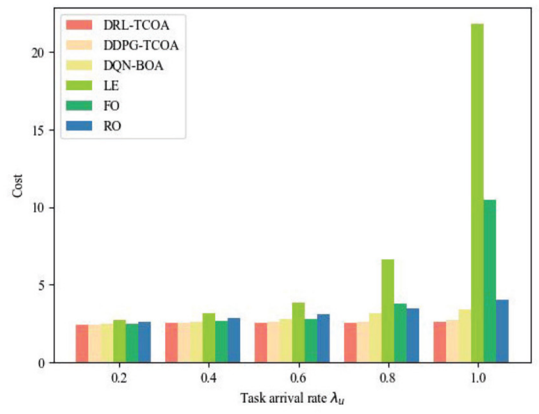


Fig. 11. Impact of vehicle number N on cost.

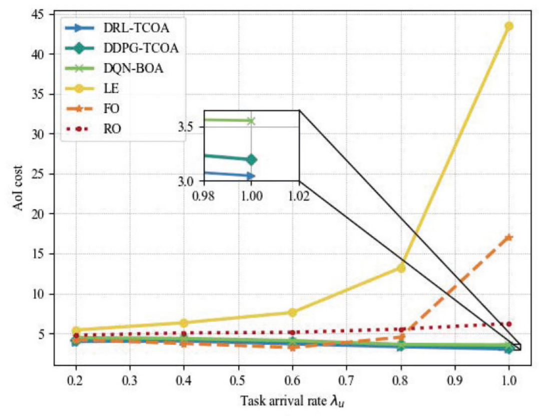
According to (20), we can calculate that the UAV maximum propulsion energy consumption is 3209.8 J and the minimum hovering energy consumption is 421.2 J. Thus, we observe that the UAV maximum available energy varies between $[500, 3000]$ J and compare it to the case where there is no energy limit. As the UAV available energy increases, the system cost decreases slightly. This is due to the fact that when the UAV's available energy increases, the UAV flight capacity increases and is able to provide computing services to more vehicle users. At the same time, the small decrease in system cost indicates that our DRL-TCOA algorithm is more tunable and adaptable and can achieve good performance under different constraints. In addition, we can see that the AoI cost and penalty gradually decrease with the increase of available energy, while the rental price gradually increases, which indicates that the UAV service capability is also gradually increasing. When the initial energy of the UAV is 3000 J, the penalty is almost 0, indicating that the UAV service capability is almost unaffected by the energy constraint at this point.

3) Impact of Vehicle Numbers: Fig. 11 shows the impact of vehicle numbers on the system average cost. We observe that the total system cost increases with the increases of these numbers, an increase because of the increase of the tasks to be processed. It still shows that the DQN-BOA strategy fails to obtain any results when the vehicle number is 15 due to the exponential proliferation of the action space dimension, whereas the DRL-TCOA algorithm and the DDPG-TCOA strategy working in continuous action space can well solve our optimization problem. Moreover, the total system cost and the AoI cost of the FO strategy exceed that of the LE strategy when the vehicle number is too large. This is because when the corresponding task number is large enough or above the computation capacity of the RSU, the queuing time on the RSU side may be longer than the local processing time.

4) Impact of Task Arrival Rate: The total system cost and the AoI cost at different task arrival rates λ_u are shown in Fig. 12. As the task arrival rate increases, the system cost produced by the DRL-TCOA algorithm and the DDPG-TCOA strategy varies little and the AoI cost decreases. This is because the task arrival rate increases, and the update frequency of tasks accelerates, which shortens the system waiting time.

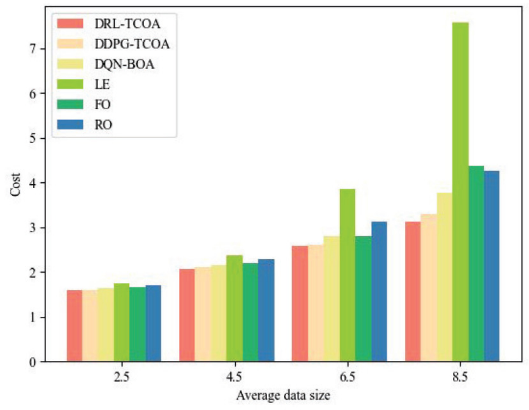


(a)

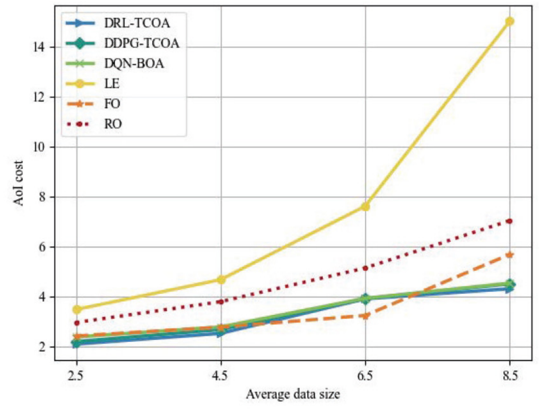


(b)

Fig. 12. Impact of task arrival rate λ_u on cost. (a) Cost under different strategies with λ_u. (b) AoI cost under different strategies with λ_u.



(a)



(b)

Fig. 13. Impact of average data size on cost. (a) Cost under different strategies. (b) AoI cost under different strategies.

At the same time, the increase in tasks also leads to an increase in the computation time, energy consumption, and rental price. It also proves that the DRL-TCOA algorithm is able to balance the three costs. On the other hand, the total system cost and the AoI cost of the LE strategy increase significantly because the fast updated computing tasks are beyond the local computation capacity, and the queuing time increases dramatically. Similarly, when λ_u is large (e.g., 1), the number of tasks exceeds the computation capacity of the RSU, and the cost of the FO strategy increases dramatically.

5) Impact of Task Data Size: Fig. 13 illustrates the impact of different sizes of computing tasks on the total system cost and the AoI cost. The average data size of the computing tasks generated by each time slot of the vehicle gradually increases from 250 to 850 kB. Obviously, the total system cost and the AoI cost increase with the increase in task data size. Besides, the system cost of the DRL-TCOA algorithm, the DDPG-TCOA strategy, and the DQN-BOA strategy decreases significantly as the data size increases. However, the DRL-TCOA algorithm always has the best performance. When the average data size is small (e.g., 250 kB), the DRL-TCOA algorithm reduces the total system cost by 9.46%, 4.36%, and 6.99% compared to LE, FO, and RO strategies, respectively. These ratios increase to 58.76%, 28.55%, and 26.64% when the data size increases to 850 kB. This result further demonstrates that our algorithm can be applied to computing networks with larger data sizes with greater adaptability.

VII. Conclusion

In order to provide and ensure real-time and efficient processing of vehicle tasks, this article investigates a joint optimization problem of trajectory control and computation offloading in a proposed UAV-assisted VEC network. Considering the stochastic nature of task arrivals and the highly dynamic changes in the network environment, we propose a TD3-based DRL-TCOA algorithm to solve a nonconvex complicated optimization problem. Compared with the traditional DQN algorithm and the other three baseline strategies, this algorithm converges fast during training process and well realizes the optimization purpose. The considered optimization performance presented in the simulation results is better than those of the other strategies from different perspectives. In addition, our proposed algorithm can effectively reduce the total system cost and improve the overall performance of the system in different scenarios by adjusting the weights on AoI cost, energy consumption, and rental price.

For our future work, we will start from the following aspects. We will consider the possibility of packet loss during task offloading when vehicles travel at high speeds and possibly collaborative offloading between vehicles or RSUs. We will try to combine new optimization metrics, for example, transmission rate or error with promising technologies, such as NOMA and RIS to improve transmission and computation efficiency of UAV-assisted VEC systems with high reliability.

References

[1] W. Tong, A. Hussain, W. X. Bo, and S. Maharjan, "Artificial intelligence for vehicle-to-everything: A survey," *IEEE Access*, vol. 7, pp. 10823–10843, 2019.

[2] A. Waheed et al., "A comprehensive review of computing paradigms, enabling computation offloading and task execution in vehicular networks," *IEEE Access*, vol. 10, pp. 3580–3600, 2022.

[3] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mob. Netw. Appl.*, vol. 26, no. 3, pp. 1145–1168, Jun. 2021. [Online]. Available: https://doi.org/10.1007/s11036-020-01624-1

[4] X. Huang, L. He, X. Chen, L. Wang, and F. Li, "Revenue and energy efficiency-driven delay-constrained computing task offloading and resource allocation in a vehicular edge computing network: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8852–8868, Jun. 2022.

[5] C. Li et al., "Dependency-aware vehicular task scheduling policy for tracking service VEC networks," *IEEE Trans. Intell. Veh.*, vol. 8, no. 3, pp. 2400–2414, Mar. 2023.

[6] H. Tang, H. Wu, G. Qu, and R. Li, "Double deep Q-network based dynamic framing offloading in vehicular edge computing," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 3, pp. 1297–1310, May/Jun. 2023.

[7] X. Peng et al., "Deep reinforcement learning for shared offloading strategy in vehicle edge computing," *IEEE Syst. J.*, vol. 17, no. 2, pp. 2089–2100, Jun. 2023.

[8] L. Yao, X. Xu, M. Bilal, and H. Wang, "Dynamic edge computation offloading for Internet of Vehicles with deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 11, pp. 12991–12999, Nov. 2023.

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

[10] Y. Zeng, I. Guvenc, R. Zhang, G. Geraci, and D. W. Matolak, *UAV Communications for 5G and Beyond*. Hoboken, NJ, USA: Wiley, 2020.

[11] H. Zhang, L. Song, and Z. Han, *Unmanned Aerial Vehicle Applications Over Cellular Networks for 5G and Beyond* (Wireless Networks). Cham, Switzerland: Springer, 2020.

[12] F. Zhou, R. Q. Hu, Z. Li, and Y. Wang, "Mobile edge computing in unmanned aerial vehicle networks," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 140–146, Feb. 2020.

[13] C. Wang, T. Yao, T. Fan, S. Peng, C. Xu, and S. Yu, "Modeling on resource allocation for age-sensitive mobile edge computing using federated multi-agent reinforcement learning," *IEEE Internet Things J.*, vol. 11, no. 2, pp. 3121–3131, Jan. 2024.

[14] W. Wu, P. Yang, W. Zhang, C. Zhou, and X. Shen, "Accuracy-guaranteed collaborative DNN inference in Industrial IoT via deep reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4988–4998, Jul. 2021.

[15] B. Li, R. Yang, L. Liu, J. Wang, N. Zhang, and M. Dong, "Robust computation offloading and trajectory optimization for multi-UAV-assisted MEC: A multi-agent DRL approach," *IEEE Internet Things J.*, vol. 11, no. 3, pp. 4775–4786, Feb. 2024.

[16] Z. Liu, J. Qi, Y. Shen, K. Ma, and X. Guan, "Maximizing energy efficiency in UAV-assisted NOMA-MEC networks," *IEEE Internet Things J.*, vol. 10, no. 24, pp. 22208–22222, Dec. 2023.

[17] F. Pervez, A. Sultana, C. Yang, and L. Zhao, "Energy and latency efficient joint communication and computation optimization in a multi-UAV assisted MEC network," *IEEE Trans. Wireless Commun.*, early access, Jul. 11, 2023, doi: 10.1109/TWC.2023.3291692.

[18] T. P. Truong, N.-N. Dao, and S. Cho, "HAMEC-RSMA: Enhanced aerial computing systems with rate splitting multiple access," *IEEE Access*, vol. 10, pp. 52398–52409, 2022.

[19] N. T. Hoa, D. V. Dai, L. H. Lan, N. C. Luong, D. V. Le, and D. Niyato, "Deep reinforcement learning for multi-hop offloading in UAV-assisted edge computing," *IEEE Trans. Veh. Technol.*, vol. 72, no. 12, pp. 16917–16922, Dec. 2023.

[20] L. Li, W. Guan, C. Zhao, Y. Su, and J. Huo, "Trajectory planning, phase shift design and IoT devices association in flying-RIS-assisted mobile edge computing," *IEEE Internet Things J.*, vol. 11, no. 1, pp. 147–157, Jan. 2024.

[21] L. Zhao et al., "Vehicular computation offloading for industrial mobile edge computing," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7871–7881, Nov. 2021.

[22] J. Hu, C. Chen, L. Cai, M. R. Khosravi, Q. Pei, and S. Wan, "UAV-assisted vehicular edge computing for the 6G Internet of Vehicles: Architecture, intelligence, and challenges," *IEEE Commun. Stand. Mag.*, vol. 5, no. 2, pp. 12–18, Jun. 2021.

[23] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. IEEE INFOCOM*, 2012, pp. 2731–2735.

[24] Z. Qin et al., "AoI-aware scheduling for air-ground collaborative mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 22, no. 5, pp. 2989–3005, May 2023.

[25] H. Li, J. Zhang, H. Zhao, Y. Ni, J. Xiong, and J. Wei, "Joint optimization on trajectory, computation and communication resources in information freshness sensitive MEC system," *IEEE Trans. Veh. Technol.*, early access, Nov. 6, 2023, doi: 10.1109/TVT.2023.3326808.

[26] C. Zhao, S. Xu, and J. Ren, "AoI-aware wireless resource allocation of energy-harvesting-powered MEC systems," *IEEE Internet Things J.*, vol. 10, no. 9, pp. 7835–7849, May 2023.

[27] H. Chen, X. Qin, Y. Li, and N. Ma, "Energy-aware path planning for obtaining fresh updates in UAV-IoT MEC systems," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2022, pp. 1791–1796.

[28] M. Xie, J. Gong, X. Jia, and X. Ma, "Age and energy tradeoff for multicast networks with short packet transmissions," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 6106–6119, Sep. 2021.

[29] X. Diao, X. Guan, and Y. Cai, "Joint offloading and trajectory optimization for complex status updates in UAV-assisted Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 23881–23896, Dec. 2022.

[30] Z. Ning et al., "Mobile edge computing enabled 5G health monitoring for Internet of Medical Things: A decentralized game theoretic approach," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 2, pp. 463–478, Feb. 2021.

[31] G. Luo et al., "Software-defined cooperative data sharing in edge computing assisted 5G-VANET," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 1212–1229, Mar. 2021.

[32] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Commun. Lett.*, vol. 3, no. 6, pp. 569–572, Dec. 2014.

[33] V. D. Tuong, T. P. Truong, T.-V. Nguyen, W. Noh, and S. Cho, "Partial computation offloading in NOMA-assisted mobile-edge computing systems using deep reinforcement learning," *IEEE Internet Things J.*, vol. 8, no. 17, pp. 13196–13208, Sep. 2021.

[34] Y. Liu et al., "Joint communication and computation resource scheduling of a UAV-assisted mobile edge computing system for platooning vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8435–8450, Jul. 2022.

[35] 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 Trans. Mobile Comput.*, vol. 21, no. 10, pp. 3536–3550, Oct. 2022.

[36] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

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

[38] S. K. Kaul and R. D. Yates, "Age of information: Updates with priority," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 2644–2648.

[39] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1183–1210, May 2021.

[40] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 1054–1054, Sep. 1998.

[41] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor–critic methods," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596. [Online]. Available: https://proceedings.mlr.press/v80/fujimoto18a.html

Junjie Yan received the bachelor's degree in communication engineering from the College of Communication Engineering, Jilin University, Changchun, China, in 2017, where she is currently pursuing the Ph.D. degree.

Her research interests mainly include intelligent edge computing, UAV communications, and deep reinforcement learning.

Xiaohui Zhao received the Ph.D. degree in applied mathematics and control theory from the Université de Technologie de Compiègne, Compiègne, France, in 1993.

He was a Postdoctoral Researcher with the Institute of Automatic Control, Southeast University, Nanjing, China, from 1994 to 1996 and a Senior Visiting Scholar for half a year with the Laboratoire d'Informatique, Université de Pierre et Marie Curie, Paris, France, in 2006. He is currently a Professor of Communication Engineering with Jilin University, Changchun, China. His research interests include wireless communication, cognitive radio, and adaptive signal processing.

Zan Li (Member, IEEE) received the Ph.D. degree from the University of Bern, Bern, Switzerland, in 2016.

Then, he worked with Alibaba Group, Hangzhou, China, as a Senior Algorithm Engineer. He is currently an Associate Professor with the College of Communication Engineering, Jilin University, Changchun, China. His current research interests include indoor positioning, machine learning, wireless communication, and cognitive radio.

Dr. Li won the Fritz-Kutter Award 2016 (The Best Ph.D. Dissertation in Swiss Universities) for his Ph.D. dissertation.