Joint Data Caching and Computation Offloading in UAV-Assisted Internet of Vehicles via Federated Deep Reinforcement Learning

Jiwei Huang, *Senior Member, IEEE*, Man Zhang, Jiangyuan Wan, Ying Chen, *Senior Member, IEEE*, and Ning Zhang, *Senior Member, IEEE*

***Abstract*—Due to the dense buildings around the macro base stations (MBSes) and the hotspot requests within particular area (e.g., traffic intersections), it is a challenging task for Quality of Service (QoS) guarantee in Internet of Vehicle (IoV). To address these challenges, unmanned aerial vehicles (UAVs) can be integrated into mobile edge computing (MEC) for IoV by leveraging their advantages of mobile flexibility, low price, and line-of-sight (LoS) communication links. In this paper, we establish a joint UAV-assisted IoV scenario, where both UAVs and MBSes can provide computation and data caching services for smart vehicles. Then, we formulate a joint optimization problem for dynamic data caching and computation offloading, aiming to minimize the average task processing delay and maximize the UAV cache hit ratio. By applying deep reinforcement learning (DRL) techniques, we design an intelligent data caching and computation offloading (IDCCO) algorithm to deal with large-scale and continuous state and action spaces. Furthermore, in order to accelerate the convergence speed of DRL model training while protecting the privacy of original user data in IoV, we propose a distributed training mechanism based on Federated Learning (FL), where the DRL model training is performed locally on UAV and global parameter aggregation is performed on MBS. Finally, extensive experiments are conducted, and the experimental results demonstrate the superiority of our approach over several comparative algorithms in shortening the training time, reducing the task processing delay, and maximizing the cache hit ratio.**

***Index Terms*—Computational offloading, data caching, federated deep reinforcement learning, Internet of Vehicle, mobile edge computing, unmanned aerial vehicles.**

Manuscript received 9 October 2023; revised 11 March 2024 and 14 May 2024; accepted 18 June 2024. Date of publication 18 July 2024; date of current version 7 November 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFD2001000 and Grant 2022YFD2001002, in part by Beijing Natural Science Foundation under Grant L232050, in part by the Project of Cultivation for young top-motch Talents of Beijing Municipal Institutions under Grant BPHR202203225, and in part by Young Elite Scientists Sponsorship Program by BAST under Grant BYESS2023031. The review of this article was coordinated by Dr. Lin X. Cai. *(Corresponding author: Ying Chen.)*

Jiwei Huang, Man Zhang, and Jiangyuan Wan are with the Beijing Key Laboratory of Petroleum Data Mining, China University of Petroleum, Beijing 102249, China (e-mail: huangjw@cup.edu.cn; manzhang0902@163.com; cup_wjy1225@163.com).

Ying Chen is with the Computer School, Beijing Information Science and Technology University, Beijing 100101, China (e-mail: chenying@bistu.edu.cn).

Ning Zhang is with the University of Windsor, Windsor, ON N9B 3P4, Canada (e-mail: ning.zhang@uwindsor.ca).

Digital Object Identifier 10.1109/TVT.2024.3429507

I. Introduction

With the rapid evolution of Internet of Vehicle (IoV), a multitude of vehicular applications have emerged to provide diverse services, such as intelligent navigation, in-vehicle games, virtual reality and entertainment videos [1]. Most of these applications have high demand on the computing and storage capacity of vehicles, but the current computing capacity and storage capacity of vehicles are limited. To address this limitation, mobile edge computing (MEC) emerges as a viable solution, which integrates task collection, computing and storage services at the edge of the networks by deploying edge servers (ESes) in the base stations (BSes) [2], [3]. Despite the potential benefits of MEC in empowering IoV [4], certain challenges persist when ESes are statically deployed on macro base stations (MBSes). For example, in urban areas, the communication links between the MBS and vehicles may be interfered by dense buildings, resulting in unstable communication and task processing. In addition, at traffic intersections with heavy traffic flow, the burst of huge simultaneous task arrivals makes it difficult for ESes to process the tasks in a timely manner. Therefore, a more flexible strategy of deploying the edge servers is urgently required.

In recent years, unmanned aerial vehicles (UAVs) have been widely used in MEC due to their advantages of mobile flexibility, low price, and line-of-sight (LoS) communication links [5], [6], [7]. In the IoV, UAVs equipped with ES can be flexibly deployed in request hotspots (e.g., traffic intersections) to assist the MBS in providing computing and storage services for vehicles. Unlike edge servers on the ground, the communication links between UAVs and vehicles are usually LoS links, which are not interfered by buildings. With the assistance of UAVs, computational tasks from vehicles can be flexibly offloaded to either the MBSes or UAVs according to the current network state. An optimal strategy of dynamic computation offloading is critical to the end-to-end performance of UAV-assisted MEC in IoV. Meanwhile, existing in-vehicle applications are data-intensive and large amounts of data (e.g., codes, databases, trained AI models, etc.) has to be stored in ES. Due to the limited storage capacity of its embedded system, UAV can only store part of the data [8], and thus it has to dynamically choose the most popular data for caching in UAV-assisted IoV.

For data-intensive tasks, existing data caching algorithms can be divided into two categories, including traditional methods and learning-based approaches. Traditional methods are based on convex optimization or probabilistic modeling. However, as various attributes (such as content popularity and user mobility) in IoV are dynamic, these traditional strategies are difficult to adapt to the dynamic environment. Moreover, obtaining the necessary global information in reality poses significant challenges. To address these limitations, recent research works apply advanced machine learning (ML) technologies including deep reinforcement learning (DRL) [9], multilayer perceptrons (MP) and convolutional neural networks (CNN) [10] to dynamically predict the data popularity for data caching. However, most of them employ centralized learning algorithms, leading to the following issues. As the number of users increases, data transmission and model training will consume excessive communication and computation resources [11]. In addition, the increase of training data also makes the training of centralized learning model more and more difficult. Finally, the transmission of user data raises concerns about potential personal privacy leakage. Therefore, it is essential to devise a solution to obtain the optimal global data caching strategy with high efficiency and low cost in dynamic scenarios while protecting user privacy for IoV.

To address the above problems, we propose an intelligent data caching and computation offloading algorithm that combines DRL and federated learning (FL), namely Fed-IDCCO. Basically, the procedure of our approach is to iteratively execute the follow three steps. Firstly, heterogeneous task information of vehicles is sent to UAVs, meanwhile, each UAV collects task information within its coverage area and trains DRL models individually based on the task information and the current environmental network state. Secondly, after each round of training, the updated parameters of each UAV's DRL model are transmitted to the MBS. Upon receiving the data from UAVs, the MBS aggregates all the received DRL model parameters using a federated averaging scheme. Thirdly, the UAVs download the aggregated parameters from MBS to update their local DRL models, with which the data caching and computation offloading can be optimized and the DRL model can be trained in the subsequent iterations.

The contribution of this paper is mainly three-fold as follows.

1) We present a UAV-assisted vehicular edge computing network, where vehicles on the road generate heterogeneous data-intensive tasks, UAVs hover over traffic intersections and the MBS covers the entire area. Both UAVs and the MBS can provide data caching and computation offloading services for vehicles. Furthermore, we formulate a joint optimization problem for data caching and computation offloading with the objectives of minimizing the task average processing delay and maximize the UAV cache hit ratio.
2) In addressing the dynamic nature of the IoV, we account for various factors such as the time-varying computational resources available of ES, vehicle mobility, and regional prevalent data dynamics. To solve the large-scale dynamic scenario problem, we propose a DRL-based algorithm, which can effectively solve the Markov Decision Process (MDP) space explosion problem caused by the dynamic scenario and obtain the optimal data caching and computation offloading strategy.
3) Recognizing the significance of user privacy and the need to expedite model training convergence, we propose a distributed training mechanism based on FL. Experimental results show that our Fed-IDCCO algorithm can effectively reduce the average task processing delay and maximize the UAV cache hit ratio in dynamic network scenarios compared with several baseline algorithms, while accelerating the convergence speed of the DRL model compared with the centralized approaches.

The remainder of this paper is organized as follows. Section II describes the system model and formulates an optimization problem for data caching and computation offloading. Section III designs the IDCCO algorithm, and proposes a distributed framework for IDCCO using FL architecture. Section IV shows the experimental results. Section V discusses the related work. Finally, Section VI concludes this paper.

II. System Model and Problem Formulation

In this section, we firstly present the UAV-assisted IoV system, followed by the detailed system model, including caching model and offloading model. Then, we formulate an optimization problem of joint data caching and computation offloading for UAV-assisted IoV. For convenience, the key notations used are summarized in Table I.

A. Network Architecture

As shown in Fig. 1, we consider an edge-computing vehicle network scenario consisting of one MBS, U UAVs and N vehicles. Please note that, for IoV systems with multiple MBSes, we can divide the system into several sub-systems in each of which there is only one MBS, and then solve the caching and offloading problem simultaneously. The sets of UAVs and vehicles are denoted by $\mathcal{U} = \{1, 2, \ldots, U\}$ and $\mathcal{N} = \{1, 2, \ldots, N\}$, respectively. We assume that the coverage areas of UAVs do not overlap, and each UAV approximately covers the same number of vehicles, i.e., $|\mathcal{N}_1| = |\mathcal{N}_2| = \ldots = |\mathcal{N}_u| = N/U$. Both the MBS and UAVs have caching and computing capabilities to provide services to vehicles. The MBS statically stores all the data requested by the vehicle tasks, while UAVs can only cache part of the data due to their limited storage space. UAVs need to download new data from the MBS to replace the outdated data for updating cache. We divide time into a set of discrete time slots $\mathcal{T} = \{1, 2, \ldots, t, \ldots\}$, and the length of each time slot is set to l_t. We assume that each vehicle n ($n \in \mathcal{N}$) in system continuously generates a data-intensive task in each time slot t ($t \in \mathcal{T}$).

The specific disposal process of each vehicle task is as follows: (1) The UAV calculates the popularity of each pre-cached data in the coverage area. Each vehicle generates data-intensive tasks with different computational/data resource requirements at each time slot t, and the vehicle sends the resource requirement information of this task to the UAV; (2) The DRL model in the UAV makes computational offloading and data caching decisions based on the data popularity, task information, environmental

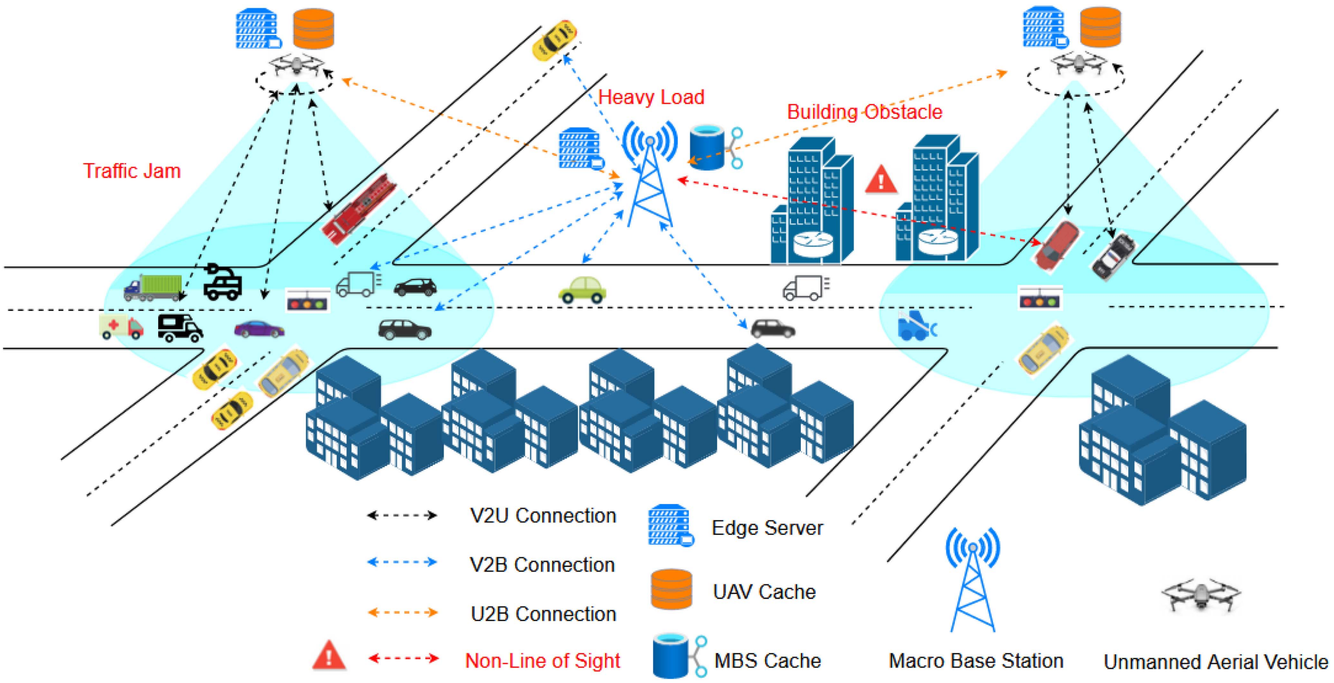

Fig. 1. System architecture.

network state, etc. and sends them back to the vehicle; and (3) The UAV chooses to cache the appropriate types of data while determining the computational resources allocated to each vehicle. After the vehicle receives the offloading strategy, the tasks will be sent to UAV or the MBS for processing through wireless link.

B. Caching Model

To provide computing and caching services for various vehicle applications, such as HD videos, augmented/virtual reality and intelligent navigation, UAVs should pre-cache the data needed for computing, including trained ML models, program code bases, and databases. We assume that there are F types of data which are needed for computation. The data set is denoted by $\mathcal{F} = \{1, 2, \ldots, F\}$. The set of cache states of f types of data in UAV $u \in \mathcal{U}$ is denoted by $X_u^t = \{x_{u,1}^t, x_{u,2}^t, \ldots, x_{u,f}^t\}$. The popularity of data f at UAV u is denoted by p_f^u, which is a key metric for evaluating the frequency of data requests, following the Mandelbrot-Zipf (MZipf) distribution [12], [13]. Then, p_f^u is given by

$$p_f^u = \frac{I_u(f)^{-z_u}}{\sum_{u\in\mathcal{U}} I_u(f)^{-z_u}}, \tag{1}$$

where $I_u(f)$ indicates the rank of data f in descending order of its popularity within the coverage area of UAV u, and z_u is a skewness factor in the range of [0.6, 1.2] [14]. The larger z_u is, the fewer popular data in UAV u is requested by vehicles. By taking popularity into account, caching policy can more accurately predict which data is most likely to be requested in the future.

At time slot t, the binary decision variable of whether UAV u caches data f is expressed as

$$x_{u,f}^t = \begin{cases} 1 & , \text{data } f \text{ is cached by UAV } u, \\ 0 & , \text{otherwise.} \end{cases} \tag{2}$$

Due to the limited memory capacity of UAVs, only part of hotspot data can be cached in UAVs. The size of data f is denoted as l_f. The data caching decision $x_{u,f}^t$ of UAV u is limited by the size of the UAV memory size L_u at time slot t. Such constrains can be expressed as follows

$$\sum_{f=1}^{F} l_f x_{u,f}^t \leq L_u. \tag{3}$$

UAVs need to be flexible in deciding which data should be cached based on changing network conditions. Thus, our first optimization objective is to maximize the UAV cache hit ratio. At time slot t, the cache hit ratio H_u^t within the coverage area of UAV u can be expressed as

$$H_u^t = \frac{\sum_{n\in\mathcal{N}_u} h_n^t}{N/U}, \forall u \in \mathcal{U}, \tag{4}$$

where h_n^t serves as an indicator function of whether a request for vehicle n hits or not. Provided that the contents of UAV cache have been determined, we assume that the data type of the request is f. If the data f has been cached in the UAV at this point, i.e., $x_{u,f}^t = 1$, then it is considered as a hit and there is $h_n^t = 1$, otherwise $h_n^t = 0$.

TABLE I
MAIN NOTATIONS

Notation	Explanation
U	the number of UAVs
N	the number of vehicles
F	the number of types of data
p_f^u	the popularity of the data f at UAV u
z_u	Mzipf skewness factor
$x_{u,f}^t$	the cache state of f type of data in UAV u
l_f	the size of data f
H_u^t	the cache hit ratio within the coverage area of UAV u
s_n^t	the size of input task
c_n^t	the required number of CPU cycles of task
k_n^t	the index of the requested data
f_u^t, f_m^t	the total computation resource of UAV u and MBS m at time slot t
a_n^t	the offloading indicator for vehicle n at time slot t
$\gamma_{u,n}^t$	the proportion of computation resources of UAV u allocated to task n at time slot t
B_u, B_m	the bandwidth of the UAV u and the MBS m
p_n, p_m	the transmission power of vehicle n and MBS m
$D_{n,u}(t), D_{n,m}(t), D_{m,u}(t)$	the distance between vehicle n, UAV u and MBS m at time slot t
$R_{n,u}(t), R_{n,m}(t), R_{m,u}(t)$	the data transmission rate between vehicle n, UAV u and MBS m at time slot t
$T_n^{UAV}(t)$, $T_n^{MBS}(t)$	the overall processing delay if the task of vehicle n is offloaded to UAV or MBS at time slot t

C. *Offloading Model*

We assume that each vehicle n will generate a data-intensive task at each time slot t, which is represented by

$$\pi_n^t = \left(s_n^t, c_n^t, k_n^t\right), k_n^t \in \mathcal{F}, \tag{5}$$

where s_n^t is the total size of task data (input parameters and program code), c_n^t is the total CPU cycle required to process the task, and k_n^t is the index of data required for task processing.

The task offloading decision process is divided into three steps: (1) Vehicle n generates a task and sends the attribute triplet π_n^t to the DRL agent of UAV u. (2) The DRL agent makes offloading decision a_n^t according to the resources required by task π_n^t, UAV cache state X_u^t and network environment state including vehicle locations and current available resources. (3) UAV u sends the offloading decision a_n^t back to vehicle n. Due to small amount of data during the communication processes in steps (1) and (3), the communication delays can be ignored. The binary task offloading variable of task π_n^t at time slot t can be represented by

$$a_n^t = \begin{cases} 1, & \text{task is offloaded to UAV } u, \\ 0, & \text{task is offloaded to MBS } m. \end{cases} \tag{6}$$

If the task of vehicle n is scheduled to UAV u, vehicle n must be within the coverage of UAV u.

1) Computation Model:

a) UAV computing: When a_n^t is 1, task π_n^t will be offloaded to UAV u. At each time slot t, UAV u needs to process multiple tasks from vehicles within its coverage area simultaneously. However, due to limited computation resources of UAVs, UAVs need to efficiently allocate part of their computation resources for each task. We use $\gamma_{u,n}^t$ to represent the proportion of computation resources of UAV u allocated to task π_n^t. Then, the computation delay of task π_n^t processed by UAV u is expressed as

$$T_{n,u}^{comp}(t) = \frac{c_n^t}{\gamma_{u,n}^t f_u^t}, \tag{7}$$

where f_u^t is the total computation resource of UAV u at time slot t, and $\gamma_{u,n}^t$ is a continuous variable ranging from 0 to 1.

b) MBS computing: When a_n^t is 0, task π_n^t will be offloaded to MBS m. We assume that the MBS processes tasks in a serial manner. The computation delay of task π_n^t processed by MBS m is expressed as

$$T_{n,m}^{comp}(t) = \frac{c_n^t}{f_m^t}, \tag{8}$$

where f_m^t is the total computation resource of MBS m at time slot t.

The processing delay of a vehicle's task generally consists of four parts: (1) communication delay of offloading the task to UAVs or the MBS; (2) computation delay of processing the task; (3) data downloading delay due to UAV cache missing; and (4) down-link delay for returning the computation result. However, (4) is usually ignored, because most applications return results that are much smaller than the input data. In addition, the down-link rate is typically several times higher than the upload link rate.

2) Communication Model: If the communication link between two equipments is unobstructed, we define the state of the communication link as Line of Sight (LoS); otherwise, it is defined as Non-Line of Sight (NLoS) [15]. In our work, we assume that the communication links between a vehicle to a UAV (V2U) and MBS to UAVs (M2U) are LoS. Due to large number of buildings around the MBS, the channel state between a vehicle and MBS is set to NLoS.

a) Vehicle to UAV: In this paper, in order to address the request overloading at the traffic intersections, UAVs are deployed above the intersections and hover at a fixed height H. Thus, the position of UAV u can be considered fixed which is denoted as (X_u, Y_u, H). The position of vehicle n is denoted as (X_n^t, Y_n^t). According to Euclidean formula, we obtain the horizontal distance between vehicle n and UAV u at time slot t as follows

$$D_{n,u}(t) = \sqrt{(X_u - X_n^t)^2 + (Y_u - Y_n^t)^2 + H^2}. \tag{9}$$

The data transmission rate of offloading the task from vehicle n to UAV u can be calculated by

$$R_{n,u}(t) = B_{n,u}\log_2\left(1 + \frac{p_n g_0 G_0}{\sigma^2 (D_{n,u}(t))^{\beta_1}}\right), \tag{10}$$

where $B_{n,u}$ is the spectrum bandwidth of vehicle to UAV (V2U) communication channel, p_n is the transmission power of vehicle n, g_0 is the channel power gain at a distance of 1 m, $G_0 \approx 2.2846$ [16], β_1 is the path loss exponent, and σ^2 is the white Gaussian noise. In the above formula, we assume that all vehicles access UAV u through Orthogonal Frequency Division Multiple Access (OFDMA) with spectrum resource allocation, and different V2U links do not interfere with each other. The coverage areas between UAVs do not overlap, so there will be no interference between UAVs. According to (10), the transmission delay from vehicle n to UAV u is given by

$$T_{n,u}^{comm}(t) = \frac{s_n^t}{R_{n,u}(t)}. \tag{11}$$

b) Vehicle to MBS: Similar to (10), the transmission rate from vehicle n to the MBS can be calculated by

$$R_{n,m}(t) = B_{n,m}\log_2(1 + \frac{p_n g_0 G_0}{\sigma^2 \beta_2 (D_{n,m}(t))^{\beta_1}}), \tag{12}$$

where $B_{n,m}$ is the spectrum bandwidth of vehicle to MBS (V2B) communication channel, $D_{n,m}(t)$ is the distance between vehicle n and the MBS m at time slot t, and β_2 is an additional attenuation factor caused by NLoS links. We assume that the V2B channels also use OFDMA, so there is no interference among V2B channels. According to (12), the transmission delay from vehicle n to the MBS is expressed by

$$T_{n,m}^{comm}(t) = \frac{s_n^t}{R_{n,m}(t)}. \tag{13}$$

c) MBS to UAV: Due to the memory limit of UAVs, a UAV can only cache part of all data needed for computation, so UAV u needs to download data f from the MBS requested by π_n^t when cache missed. Similar to (10), the transmission rate from the MBS to UAV u is expressed as

$$R_{m,u}(t) = B_{m,u}\log_2\left(1 + \frac{p_n g_0 G_0}{\sigma^2 (D_{m,u}(t))^{\beta_1}}\right), \tag{14}$$

where $B_{m,u}$ is the spectrum bandwidth of MBS to UAV (B2U) communication channel, and $D_{m,u}$ is the fixed distance between MBS m and UAV u. According to (12), the downloading delay from the MBS to UAV u for data f is expressed by

$$T_{m,u,f}^{down}(t) = \frac{l_f}{R_{m,u}(t)}. \tag{15}$$

Based on (7), (11) and (15), if task π_n^t of vehicle n is offloaded to UAVs at time slot t, its overall processing delay is given by

$$T_n^{UAV}(t) = T_{n,u}^{comm}(t) + T_{n,u}^{comp}(t) + \left(1 - x_{u,f}^t\right) T_{m,u,f}^{down}(t),$$
$$x_{u,f}^t \in \{0,1\}, \forall n \in \mathcal{N}, \forall u \in \mathcal{U}, \forall f \in \mathcal{F}. \tag{16}$$

Based on (8) and (13), if task π_n^t of vehicle n is offloaded to the MBS at time slot t, its overall processing delay is given by

$$T_n^{MBS}(t) = T_{n,m}^{comm}(t) + T_{n,m}^{comp}(t), \forall n \in \mathcal{N}. \tag{17}$$

To sum up, at time slot t, the total delay of processing task π_n^t from vehicle n is expressed by

$$T_n(t) = a_n^t T_n^{UAV}(t) + \left(1 - a_n^t\right) T_n^{MBS}(t), \forall n \in \mathcal{N}. \tag{18}$$

D. Problem Formulation

We aim to minimize the average processing delay of all tasks and maximize the cache hit ratio of all UAVs, simultaneously unifying the scale by setting a weight factor λ. We jointly optimize data caching decisions $X = \{x_{u,f}^t\}_{u\in\mathcal{U}, f\in\mathcal{F}}$, vehicle task offloading decisions $A = \{a_n^t\}_{n\in\mathcal{N}}$, and UAV computing resource allocation $\Gamma = \{\gamma_{u,n}^t\}_{n\in\mathcal{N}, u\in\mathcal{U}}$. Then, the Joint Data Caching and Computation Offloading (JDCCO) problem can be mathematically formulated as

$$\min_{x_{u,f}^t, a_n^t, \gamma_{u,n}^t} \frac{1}{T}\sum_{t=1}^{T}\left[\sum_{u=1}^{U} \lambda(1 - H_u^t) + \sum_{n=1}^{N_u} T_n(t)\right], \tag{19}$$

$$\text{s.t. C1}: \sum_{\mathrm{n}\in\mathcal{N}_\mathrm{u}} \gamma_{\mathrm{u,n}}^{\mathrm{t}} = 1, 0 \le \gamma_{\mathrm{u,n}}^{\mathrm{t}} \le 1, \tag{20}$$

$$C2: \sum_{f=1}^{F} l_f x_{u,f}^t \le L_u, \tag{21}$$

$$C3: T_n(t) \le l_t, \tag{22}$$

$$C4: a_n^t \in \{0,1\}, x_{u,f}^t \in \{0,1\},$$
$$\forall n \in \mathcal{N}_u, \forall u \in \mathcal{U}, \forall f \in \mathcal{F} \tag{23}$$

C1 ensures that the computation resource slicing proportion of UAV u should be in range 0 to 1. C2 illustrates that the total size of data cached by UAV cannot exceed its memory capacity. C3 shows that the average processing delay of task π_n^t is bounded by the slot length l_t. Finally, C4 corresponds to the ranges of involved variables.

Due to the non-convex objective function, constraints and mixed integer variables, JDCCO problem is a non-convex mixed integer nonlinear programming problem. This type of optimization problem is difficult to solve in reasonable time using traditional mathematical methods (e.g., convex optimization, etc.), and DRL is well suited for this type of problem. In the next section, we will present how we can use DRL to solve this problem.

III. Federated DRL-Enabled Data Caching and Computation Offloading Scheme

In this section, we propose a federated intelligent data caching and computation offloading strategy based on federated deep reinforcement learning (FDRL). The FDRL agent can make the optimal caching and computation offloading strategy based on the task information of vehicles, the popularity of the requested data in each UAV's coverage area, and the current network environment.

A. MDP Model

In order to solve the JDCCO problem practically, we firstly have to formulate the original dynamic problem as a Markov Decision Process (MDP). The MDP model is generally represented by a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$. Here, $\mathcal{S}$ represents the system states, and $\mathcal{A}$ indicates the system actions. $\mathcal{T} = p(s_{t+1}|s_t, a_t)$ is the state transition probability, but $\mathcal{T}$ is difficult to obtain in practice. $\mathcal{R}$ is the reward of executing an action based on a state. The MDP model of JDCCO problem can be expressed as follows.

1) State Space $\mathcal{S}$:

$$\begin{aligned} s_u(t) = \{&s_1^t, s_2^t, \ldots, s_n^t, \\ &c_1^t, c_2^t, \ldots, c_n^t, k_1^t, k_2^t, \ldots, k_n^t, \\ &X_1^t, X_2^t, \ldots, X_n^t, Y_1^t, Y_2^t, \ldots, Y_n^t, \\ &f_u^t, f_m^t, p_1^u, p_2^u, \ldots, p_f^u\}, \\ &\forall n \in \mathcal{N}_u, \forall f \in \mathcal{F}, \end{aligned} \tag{24}$$

where s_n^t is the total size of task π_n^t, c_n^t is the total CPU cycle required to process task π_n^t, k_n^t is the index of data required by task π_n^t, X_n^t and Y_n^t are two-dimensional coordinate position of vehicle n, f_u^t and f_m^t are the available computing resources of UAV u and MBS m, respectively, p_f^u means the content popularity of data f under UAV u.

2) Action Space $\mathcal{A}$:

$$\begin{aligned} a_u(t) = \{&x_{u,1}^t, x_{u,2}^t, \ldots, x_{u,f}^t, \\ &a_1^t, a_2^t, \ldots, a_n^t, \\ &\gamma_{u,1}^t, \gamma_{u,2}^t, \ldots, \gamma_{u,n}^t\}, \\ &\forall n \in \mathcal{N}_u, \forall f \in \mathcal{F}, \end{aligned} \tag{25}$$

where $x_{u,f}^t$ is the binary decision variable of whether UAV u caches data f, a_n^t is the binary task offloading variable of task π_n^t, and $\gamma_{u,n}^t$ is the proportion of computation resources of UAV u allocated to task π_n^t.

3) Reward $\mathcal{R}$: When the current state of UAV u is $s_u(t)$, the DRL agent in UAV u will get a reward $r_u(t)$ after performing action $a_u(t)$. Based on (19), the DRL agent needs to minimize the task processing delay and maximize the UAV cache hit ratio. In addition, we need to add a large positive numerical penalty $P_u(t)$ to the reward function when environmental constraints (21) and (22) are not met. Therefore, we design the reward function consisting of utility function and QoS penalty as follows.

$$r_u(t) = \lambda(1 - H_u^t) + \sum_{n=1}^{N_u} T_n(t) + P_u(t). \tag{26}$$

In our work, DRL agents are deployed on UAVs. According to (24) and (25), the available computing resources in the state space $s_u(t)$ and the resource allocation ratios in the action space $a_u(t)$ are continuous variables, and it is hard to decompose these continuous into discrete ones. The Twin Delayed Deep Deterministic policy gradient (TD3) algorithm is suitable for handling problems with large-scale and continuous state and action space. Therefore, we take advantage of the TD3 algorithm to solve our problem, and the agents can make real-time decisions effectively.

Each DRL agent of UAV u maintains two types of neural networks, including main networks and target networks. For each type of the neural networks, there are an actor network and two critic networks [17]. The parameters of actor network and critic network from main network are denoted by ϕ, θ_1, θ_2. The parameters of actor network and critic networks from target networks are denoted by ϕ', ${\theta_1}'$, ${\theta_2}'$. The actor network aims to maximize the cumulative expected rewards by gradient ascent, which can choose either critic network to calculate the Q value. While the critic networks are updated by minimizing the error between the current value of Q and the target value of Q.

First, the DRL agent of UAV u observes the current system state $s_u(t)$. The state $s_u(t)$ is input into the actor network of main networks, and then we can obtain the action $a_u(t)$ from the output layer of actor network, where the strategy π_ϕ is used to obtain action $a(t)$, i.e.,

$$a_u(t) = \pi_\phi(s_u(t)) + \epsilon, \epsilon \sim \mathcal{N}(0, \tau), \tag{27}$$

where ϵ is the noise which can increase exploration of DRL model. ϵ follows normal distribution with mean value equal to 0 and variance equal to τ, after DRL agent executes action $a_u(t)$ obtained from (27), the current system state is updated from $s_u(t)$ to $s_u(t+1)$. Then, based on state $s_u(t)$ and action $a_u(t)$, DRL agent can get the reward $r_u(t)$ for executing action $a_u(t)$ in state $s_u(t)$. Finally, DRL agent stores the quadruple $\{s_u(t), a_u(t), s_u(t+1), r_u(t)\}$ into replay memory for speeding up model convergence. The parameter updating process of actor network uses deterministic policy gradient updating, which is expressed as

$$\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s), \tag{28}$$

where Q_{θ_1} is the Q value of the first critic network in main network. Furthermore, the parameters of main critic networks are updated by following formula:

$$\theta_i = \arg\min_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2, \tag{29}$$

where $Q_{\theta_i}(s, a)$ is the current value of Q, y is the target value of Q calculated by target critic networks. The framework of our DRL-based approach is illustrated by Fig. 2. Formally, Algorithm 1 shows the detailed procedures.

B. Federated-DRL Algorithm for Intelligent Data Caching and Computation Offloading

As described in the above subsection, DRL can dynamically and efficiently get the optimal data caching and compute the offloading strategy. However, most existing DRL-based approaches have the following limitations.

(1) The convergence process is slow because of the large amount of data used for centralized training.

(2) If the user data is uploaded to the MBS for centralized training, the transmission of large-scale data between MBS and

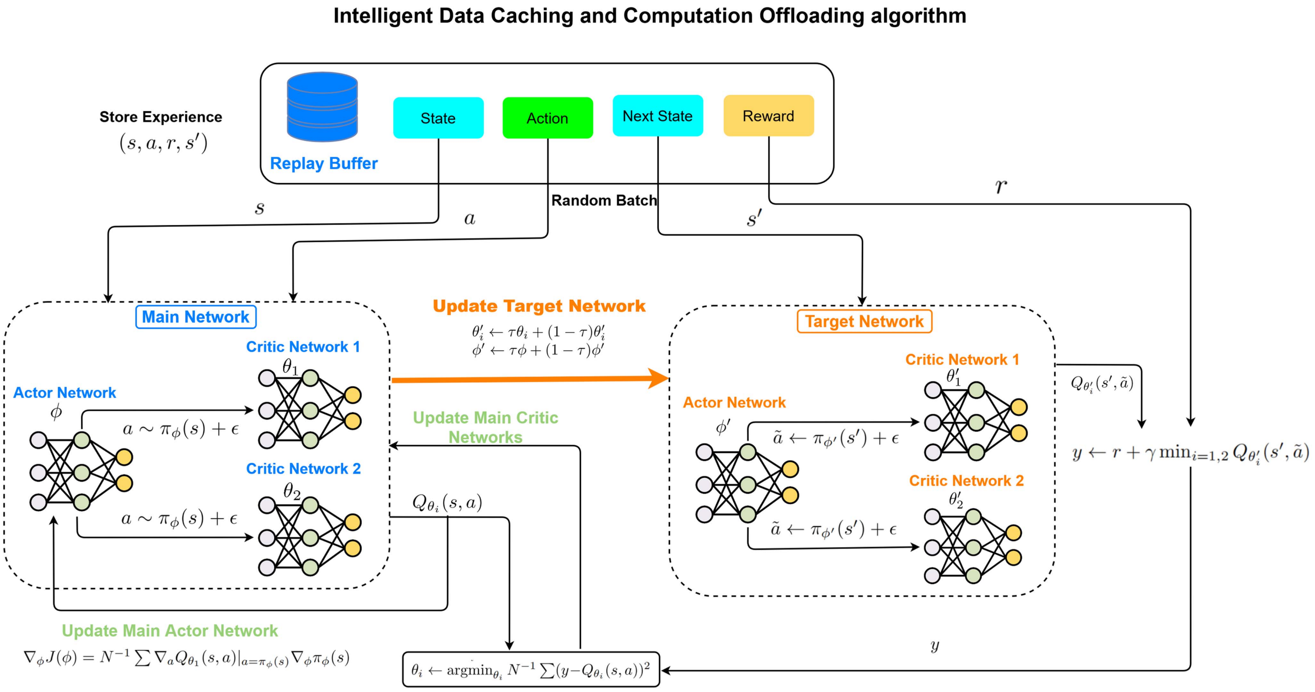
Intelligent Data Caching and Computation Offloading algorithm
Store Experience
(s, a, r, s')
Replay Buffer
State
Action
Next State
Reward
Random Batch
s
a
s'
r
Main Network
Actor Network
ϕ
Critic Network 1
θ_1
Critic Network 2
θ_2
$a \sim \pi_\phi(s) + \epsilon$
$a \sim \pi_\phi(s) + \epsilon$
Update Target Network
$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$
$\phi' \leftarrow \tau\phi + (1-\tau)\phi'$
Update Main Critic Networks
$Q_{\theta_i}(s, a)$
Update Main Actor Network
$\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s,a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
$\theta_i \leftarrow \mathrm{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s,a))^2$
Target Network
Actor Network
ϕ'
Critic Network 1
θ'_1
Critic Network 2
θ'_2
$\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon$
$\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon$
$Q_{\theta'_i}(s', \tilde{a})$
$y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
y

Fig. 2. Framework of IDCCO algorithm.

Algorithm 1: Intelligent Data Caching and Computation Offloading Algorithm (IDCCO).

Input: Training episode number N; training step number T; replay memory size D; batch size S; target smoothing coefficient η; exploration noise ϵ; learning rates of actor networks and critic networks l_a, l_c; two actor networks' weights ϕ, ϕ'; four critic networks' weights $\theta_1, \theta_2, {\theta_1}', {\theta_2}'$

Output: a_n^t: decisions for tasks offloading of vehicles in $\mathcal{N}_u$; $x_{u,f}^t$: decisions for data caching of UAV u; $\gamma_{u,n}^t$: decisions for computing resources allocating of UAV u.

```
for $episode \leftarrow 1$ to $N$ do
  Initialize environment parameters, get initial state $s_0$
  for $step \leftarrow 1$ to $T$ do
    Obtain action $a_u(t)$ using (27)
    Execute action $a_u(t)$
    Obtain the next state $s_u(t+1)$ and reward $r_u(t)$
    Store transition $\langle s_u(t), a_u(t), s_u(t+1), r_u(t)\rangle$ into replay memory $D$
    if step $\geq S$ then
      Randomly sample N experience from replay memory
      Update $\phi$ with (28)
      Update $\theta_1$ and $\theta_2$ with (29)
      Update target networks with $\eta = 0.005$:
      $\phi' \leftarrow \eta\phi + (1-\eta)\phi'$
      ${\theta_i}' \leftarrow \eta\theta_i + (1-\eta){\theta_i}'$
    end if
  end for
end for
```

vehicles is likely to bring high costs and is also limited by bandwidth. Furthermore, the central server has limited computation power and is difficult to meet the training needs of large amounts of data [18].

(3) Uploading users' data to the central server may lead to users' privacy security issues [19], [20].

Therefore, the centralized training either on UAVs or the MBS is not pratical in IoV. To address the above problems, we put forward an intelligent data caching and computation offloading algorithm based on FL, namely Fed-IDCCO. In our approach, the DRL agents can be trained in a distributed manner, each of which can maintain its data locally without revealing any user private information to the MBS on the Internet. We set the data in each UAV to be independent and identically distributed, which also improves the training speed.

The model training process is illustrated in Fig. 3. First, during each training process round i, UAVs download the global DRL network parameters $W(i)$ from the MBS. Then, each UAV locally trains the DRL agent by local data within the coverage area and upload the updated local DRL model weights $W_u(i), u \in \mathcal{U}$ to the MBS. Finally, the MBS performs federated average aggregation on the received parameters to obtain an updated global model $W(i+1)$. The specific parameter aggregation process is shown by (30) as follows.

$$W(i+1) = \frac{1}{U} \sum_{u=1}^{U} W_u(i). \tag{30}$$

MBS sends the updated global model to each local model to continue training, and repeats the above process until each

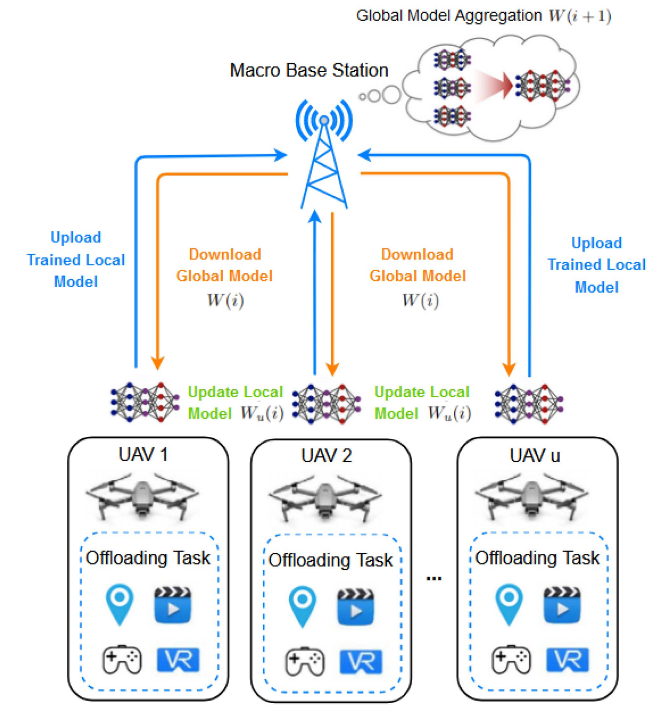
Global Model Aggregation $W(i+1)$
Macro Base Station
Upload Trained Local Model
Download Global Model $W(i)$
Download Global Model $W(i)$
Upload Trained Local Model
Update Local Model $W_u(i)$
Update Local Model $W_u(i)$
UAV 1
UAV 2
UAV u
Offloading Task
Offloading Task
Offloading Task
...

Fig. 3. FL-based model training.

```
Algorithm 2: Federated IDCCO Algorithm.
Input:
  Number of iterations $N_f$
  MBS side: Global DRL model weights $W(0)$ at the beginning of decision-making period
  UAV side:Local DRL model weights $W_u(0), \forall u \in \mathcal{U}$ at the beginning of decision-making period
Output: Trained DRL model weights of DRL models in UAVs and the MBS
for iteration $i \leftarrow 1$ to $N_f$ do
    for each $UAV \leftarrow 1$ to $U$ do
        Download global DRL model weights $W(i)$ from the MBS
        set $W_u(i) = W(i)$
        Update local DRL model weights $W_u(i)$ according to Algorithm 1
        Upload the trained weights $W_u(i)$ to the MBS
    end for
    MBS receives all trained weights $W_u(i)$ ($\forall u \in \mathcal{U}$)
    MBS executes federated aggregation based on (30)
end for
```

agent obtains the optimal and stable reward, which ensures the convergence of the global model. Meanwhile, the independent and identically distributed data makes the aggregation of models simpler, which helps to improve the convergence speed and performance of the Fed-IDCCO algorithm. The detailed procedures are shown in Algorithm 2.

In the following, we analyze the computational complexity of the two proposed algorithms. For Algorithm 1, in each episode, there are N_u vehicles in the coverage area of UAV u and each vehicle generates a task. Also there are F types of data that each task can request. The actors and critics in the DRL model are four-layer fully connected neural networks containing two hidden layers denoted as (L_1, L_2). According to (24) and (25), the dimension of the state space is $5N_u + F + 2$, and the dimension of the action space is $2N_u + F$. Therefore, the complexity of obtaining an action is $O((5N_u + F + 2)L_1 + L_1L_2 + (2N_u + F)L_2)$. In our experiments, the size of the hidden layers is set to be fixed, so the complexity of obtaining an action can be expressed as $O(N_u + F)$. Similarly, the computational complexity of the network training process is $O(N_u + F)$. Assuming that T steps are trained in a episode, the time complexity of Algorithm 1 with N episodes can be obtained as $O(NT(N_u + F))$.

In Algorithm 2, each UAV updates the local DRL model using Algorithm 1. In the remaining steps, model uploading, downloading, and parameter aggregation are done in constant-level time. We set the number of UAVs to U and the number of iterations to N_f. Finally, the time complexity of Algorithm 2 is $O(NTUN_f(N_u + F))$.

IV. Performance Evaluation

In this section, we conduct simulation experiments to evaluate the performance of the proposed scheme. We use real-world data collected from taxis in the city of Shanghai to construct the simulation scenarios, and compare the performance of our approach with other baseline algorithms. Experimental results will be discussed in detail.

A. Experimental Settings

We simulate an IoV scenario in the city of Shanghai in China. A dataset collected from taxis is used in our experiments to simulate the movement of the vehicles [21]. The dataset includes more than 7 million pieces of trajectory data collected from 4,316 taxis in Shanghai, where each piece of data contains information such as vehicle ID, time, latitude, longitude, and speed. According to the dataset, we generate the user requests with different data processing requirements at different locations.

In the experiments, we consider an MBS covering two intersections (hotspots), with a UAV deployed above each intersection to handle bursty requests during rush hours. When the MBS covers multiple intersections, our model can be extended accordingly by modifying the parameters of the number of UAVs. The number of vehicles in the coverage area of each UAV ranges from [20,70]. We assume the number of types of data ranges from [10,40]. Other specific key parameters are shown in Table II. Our experiments are conducted on NVIDIA GTX 1060 GPU with 6 GB memory. The experiment uses Python 3.6 and PyTorch 1.9 to build neural networks and train DRL models.

The actors and critics in the DRL model are four-layer fully connected neural networks containing one input layer, two hidden layers and one output layer. The two hidden layers are composed of 200 and 300 neurons respectively. The activation function between the two hidden layers is the rectifying linear unit (ReLU). We use adam optimizer to update neural network parameters. Other specific key parameters of DRL model are shown in Table III.

TABLE II
EXPERIMENTAL SETTINGS OF UAV-ASSISTED IoV SIMULATIONS

Definitions	Notations	Value
number of UAV	U	2
number of types of data	F	$10 \sim 40$
length of one time slot	l_t	1s
size of data f	l_f	$0.2L_u \sim 0.4L_u$
Mzipf skewness factor	z_u	0.5 [14]
input task size	s_n^t	5MB∼10MB
required number of CPU cycles of task	c_n^t	$10^8 \sim 10^9$
fixed altitude of UAVs	H	100m [22]
bandwidth of UAV	B_u	10MHz
transmission power of vehicle	p_n	1W [23]
channel power gain at a distance of 1m	g_0	1.42×10^{-4} [16]
gaussian white noise	σ^2	−100dBm
path loss exponent	β_1	3
additional attenuation factor caused by NLoS links	β_2	0.01 [15]
bandwidth of the MBS	B_m	40MHz
transmission power of MBS	p_m	10W

TABLE III
IDCCO PARAMETERS SETTING

Definitions	Notations	Value
learning rate of actor networks	l_a	0.001
learning rate of critic networks	l_c	0.0001
target smoothing coefficient	η	0.005
reply buffer size	D	5000
batch size	S	64
exploration noise	ϵ	0.03

In our work, we first compare the convergence performance of Fed-IDCCO with centerlized IDCCO. Then we compare the performance of Fed-IDCCO with five baseline algorithms on average task processing delay and UAV cache hit ratio. The baseline algorithms are as follows.

- *Offload to UAV:* All vehicle tasks are processed by UAVs.
- *Offload to MBS:* All vehicle tasks are processed by the MBS.
- *Least Recently Used (LRU):* The data that has not been used recently will be replaced.
- *Least Frequently Used (LFU):* The least recently used data will be replaced.
- *First In First Out (FIFO):* The first data to enter the cache will be replaced.
- *Random:* Randomly select data and put them into the cache every episode.
- *Optimal (OPT):* The UAVs know the data to be requested at the next time slot $t+1$ in advance for caching, which has the highest cache hit ratio.

B. Experimental Results

First, we demonstrate the loss function between Fed-IDCCO algorithm and centralized IDCCO algorithm, which is denoted as the average of the reward function of all agents over all time slots, as shown in Fig. 4. After 50 episodes, Fed-IDCCO algorithm and centralized IDCCO algorithm converged to the stable value, but Fed-IDCCO algorithm reached the stable value faster than centralized IDCCO algorithm. This shows that the Fed-IDCCO algorithm has better performance in terms of stability and fast convergence speed, which may be attributed to the less training data on each UAV. In addition, the parameter aggregation of FL also speeds up the training process of DRL model on each UAV.

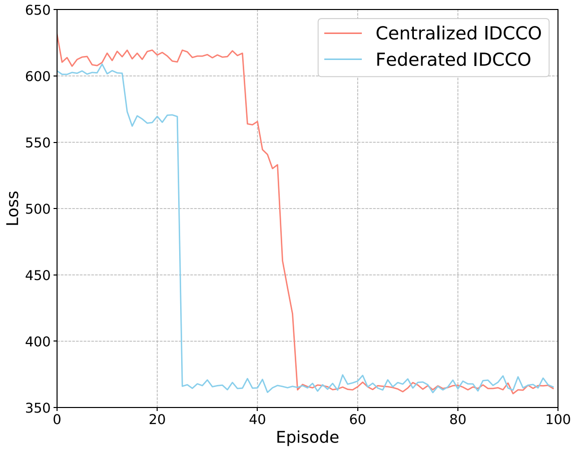

Fig. 4. Loss function between Fed-IDCCO and centralized IDCCO.

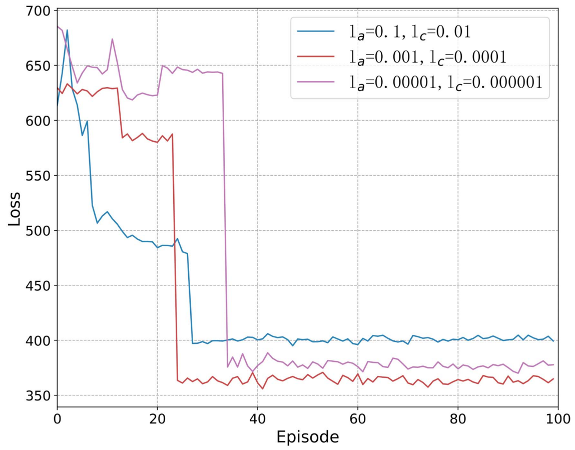

Fig. 5. Loss function under different learning rates.

Fig. 5 shows the effect of different learning rates on the convergence of the algorithm in actor and critic networks. The Fed-IDCCO algorithm converges fastest and has the lowest system loss when $l_a = 0.001$ and $l_c = 0.0001$, eventually dropping to around 360. When $l_a = 0.00001$ and $l_c = 0.000001$, the loss function of the algorithm is slow to change, indicating that the convergence is slow and the computational cost will be relatively large. With the settings of $l_a = 0.1$ and $l_c = 0.01$, the learning rate is too large, causing the model parameters to skip the minimum of the loss function during the update process, resulting in a failure to converge to the optimal solution.

Fig. 6 shows the cache hit ratios under different UAV cache capacities, where the UAV cache capacity ranges from 50 to

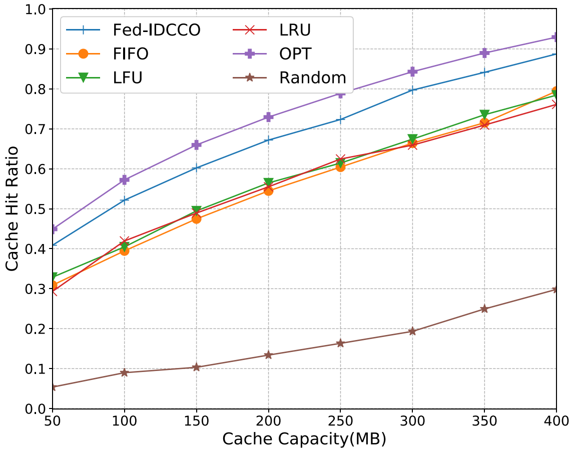

Fig. 6. Cache hit ratio with different cache capacity.

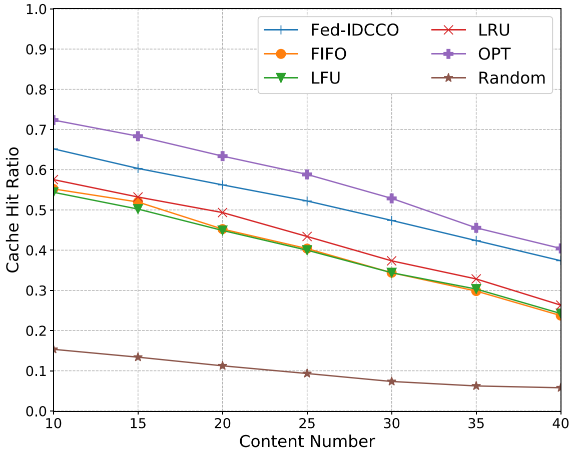

Fig. 8. Cache hit ratio with different numbers of data.

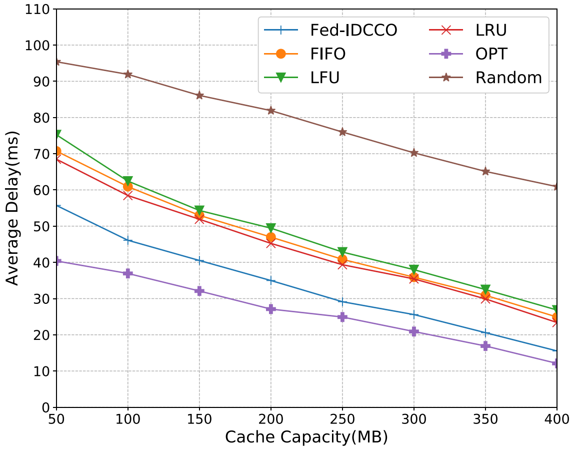

Fig. 7. Average delay with different cache capacity.

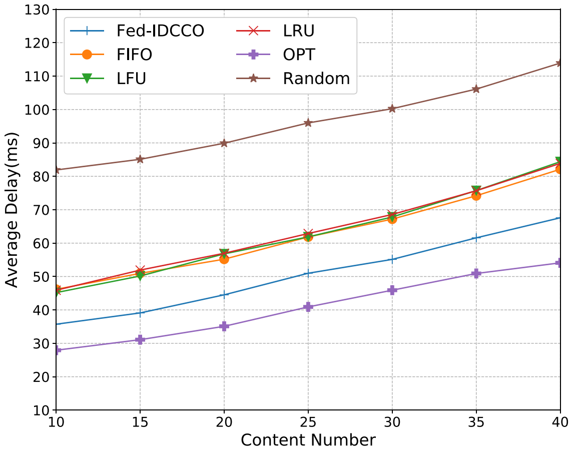

Fig. 9. Average delay with different numbers of data.

400. The Fed-IDCCO algorithm improves the hit rate by an average of 19.6%, 20.3%, 18.1%, and 3 times compared to LRU, FIFO, LFU, and Random, second only to OPT algorithm. With the increase of cache capacity, the cache hit ratios of various algorithms increase, because the larger UAV cache capacity means that more popular data can be cached, resulting in less data replacement process in the UAVs' storages.

Similar to Fig. 6, Fig. 7 shows the average task processing delays under different UAV cache capacities. The Fed-IDCCO algorithm is always better than LRU, LFU, FIFO, and Random, with an average reduction in task processing delay of 25.1%, 27.6%, 30.7%, and 60.1%, second only to OPT algorithm. The random algorithm has the highest average task processing delay. With the increase of cache capacity, the average task processing delays of all algorithms decrease, because as the UAV cache hit ratio becomes higher and higher, the UAV downloads less data from the MBS when cache misses.

Figs. 8 and 9 show the UAV cache hit ratios and average task processing delays for different numbers of data. Our proposed algorithm improves the hit rate by an average of 22%, 31.2%, 32.1% and 4.5 times, and reduces the latency by an average of 20.3%, 18.5%, 19.6% and 46.1% compared to LRU, FIFO, LFU and Random. With the increase of the number of data, the delays of all algorithms increase and the cache hit ratios decrease. Because the storages of UAVs are fixed, more and more types of data lead to the proportion of hotspot data that UAVs can cache becomes lower and lower, which indirectly leads to the increase of the process of UAVs downloading data from the MBS. In addition, the performance of our Fed-IDCCO algorithm always has the best performance. It shows that our Fed-IDCCO algorithm can flexibly adjust the offloading strategy according to the number of data to minimize the task processing delay and maximize the UAV cache hit ratio.

Figs. 10 and 11 show the network performance for different computation resources of UAVs and the MBS. The performance of Fed-IDCCO algorithm is better than UAV computing and MBS computing algorithms. Because the Fed-IDCCO algorithm can make offloading decisions flexibly based on the available computational resources of both UAVs and the MBS, it always offloads the task to the MEC server with abundant computational resource. In summary, the Fed-IDCCO algorithm is able to find the optimal offloading strategy in any complex network environment.

Fig. 12 shows the cache hit ratios for different number of vehicles within the coverage area of one UAV. The result shows that the cache hit ratio increases with the number of vehicles for

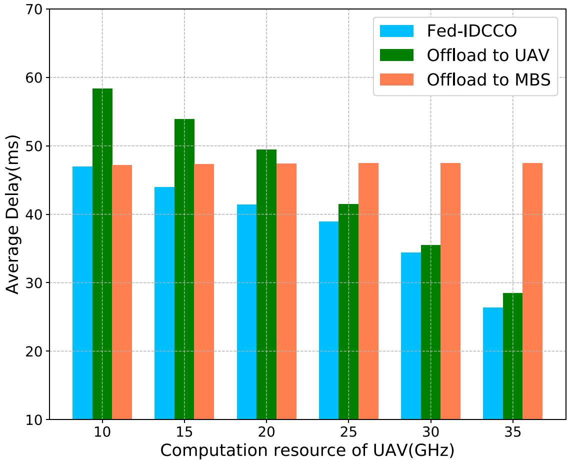

Fig. 10. Average delay with different computation resource of UAV.

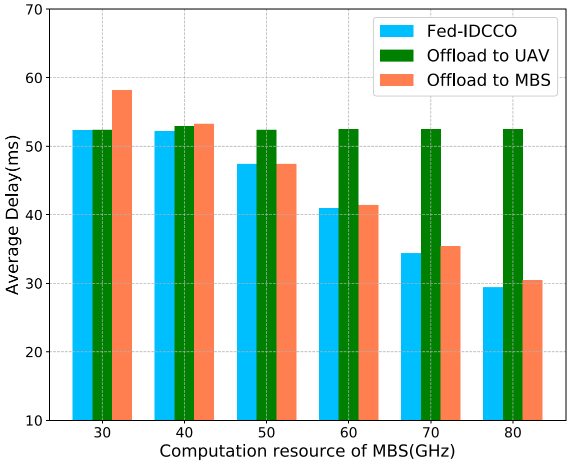

Fig. 11. Average delay with different computation resource of MBS.

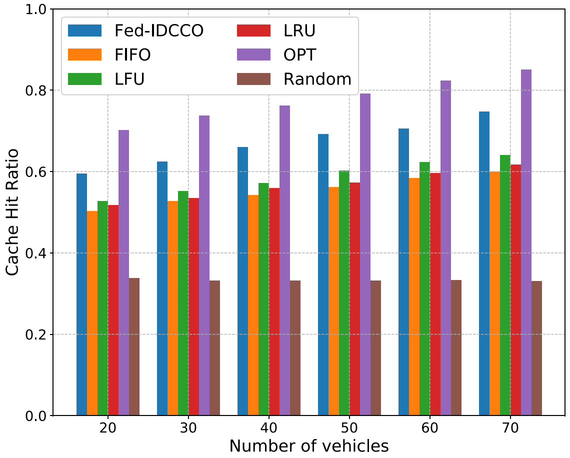

Fig. 12. Cache hit ratio with different numbers of vehicles.

all algorithms except the random algorithm. This is because as the UAVs cover more vehicles, these vehicles provide more data for the training of the DRL model, therefore, the DRL model is better able to learn the global prevalence of the data. In addition, our Fed-IDCCO algorithm has the highest cache hit ratio in all cases except for the OPT algorithm. Compared to LRU, FIFO and LFU, the cache hit rate is improved by 19.1%, 21.8% and 14.7% on average. And compared to Random algorithm, the performance of our algorithm is doubled. All these results show that the Fed-IDCCO algorithm can make the optimal caching decisions for any number of vehicles.

V. Related Work

In recent years, UAVs have been increasingly used as infrastructure service facilities in edge computing scenarios because of their low price, high flexibility, easy deployment and line-of-sight links. UAVs carrying edge servers can be deployed in hot request areas (e.g., intersections) and areas where infrastructure is lacking. UAVs can provide services such as communication, computation and storage for devices in the coverage area. There has been a lot of work related to the study of UAV-assisted edge computing.

Chen et al. [24] proposed a hybrid computing model, including UAV, edge layer, cloud type, to introduce UAV in edge computing/cloud computing to ensure high quality of service (QoS) of user tasks. Hu et al. [25] constructed an intelligent edge computing scenario for UAV-assisted mobile edge computing. The goal is to reduce the task delay by jointly optimizing the mobile trajectory and calculating the unloading ratio under the constraints of discrete variables, UAV energy consumption and moving trajectory. Du et al. [26] proposed a joint optimization problem for the computing resource allocation of edge servers, UAV charging and hovering time and heterogeneous task execution sequence, which effectively reduced the total energy consumption of UAV task processing. With the same energy consumption target, Sun et al. [27] constructed a UAV-assisted edge computing framework. They solved this problem by optimizing server CPU frequency, computing offloading strategy and UAV moving route. Hu et al. [28] studied a UAV-assisted mobile edge computation problem, which uses an alternating optimization algorithm to minimize the overall energy consumption of UAVs and terminals under constraints such as task order, bandwidth splitting, and movement trajectory. Zhang et al. [29] studied the trajectory and communication and computing resource allocation of UAVs optimized for disaster rescue scenarios, with the goal of maximizing the average total QoE. Seid et al. [30] proposed a multi-agent method to minimize long-term network computing cost in the scenario of multi-UAV clusters, and obtained the optimal task offloading and resource allocation scheme under service quality constraints. Chen et al. [31] studied the problem of energy consumption minimization of UAV deployed with MEC server, and proposed a hybrid heuristic optimization algorithm to improve the optimization efficiency of the algorithm. Ke et al. [32] proposed a joint auxiliary edge server model of UAV and macro station, in which both UAV and macro station can continuously collect renewable energy, and proposed a distributed computing offloading scheme based on deep reinforcement Learning (DCODRL). The goal is to minimize the weighted average cost of task processing.

UAVs equipped with storage resources can provide caching services for vehicles in the coverage area. UAVs caching the most frequently requested part of data from vehicles can effectively reduce the processing latency of vehicle tasks, and there has been a lot of works studying the caching problem in UAVs.

Lin et al. [33] constructs a UAV-assisted edge computing scenario, where edge servers are deployed on each UAV to serve devices on the ground, and studies an optimization problem for joint task offloading, resource allocation, content caching and UAV movement trajectory. Wang et al. [34] considers a dynamic cellular network scenario for UAVs with mobility and data requests. In order to cope with the dynamic environment, a cache placement and content delivery algorithm based on DRL is proposed. Anokye et al. [35] uses content request and random waypoint user movement model to predict the UAV's movement trajectory and caching strategy, and the algorithm uses deep reinforcement learning. Wu et al. [36] constructed a communication network capable of caching data, in which the UAV provides charging and caching servers for mobile users, and proposed a joint data caching and movement trajectory algorithm based on DRL. Wang et al. [37] proposed a novel caching strategy to cache popular data in advance on UAVs and user devices to reduce repeated data transfers in end-to-end supported UAVs networks, and proposed a cache placement optimization algorithm based on deep reinforcement learning to determine the caching strategy. Chen et al. [38] proposed a computational offloading policy for IoT devices to satisfy multiple constraints such as computational resource constraints, latency constraints, and energy consumption constraints, so as to achieve the goal of minimizing the total QoS cost of all devices.

Different from the above works, the UAVs in our scenario can provide caching service for vehicles, and the UAV caching hotspot request data can effectively reduce the processing delay of the task and improve the task processing efficiency. In addition, we fully consider the dynamic of the network environment and vehicle mobility. In our approach, we build intelligent algorithms based on FL and DRL, which not only obtain long-term optimal data caching and computation offloading strategy in dynamic network environment, but also allow DRL models to converge faster while protecting user privacy.

VI. Conclusion

In this paper, we consider a UAV-MBS-assisted mobile edge computing scenario for IoV. We study the joint data caching and computation offloading to minimize the task-average processing delay and maximize the UAV cache ratio. We design a distributed intelligent algorithm based on DRL and FL to obtain the optimal data caching and computation offloading strategy. The training process can be accelerated in a parallel manner without transmitting any user-sensitive data to the core network. Extensive experiments are constructed based on real-world dataset, and the experimental results validate the efficiency and the superiority of the proposed approach to several baseline algorithms.

For the future work, we will consider load balancing among multiple UAVs to further improve the system performance. Moreover, considering the battery of UAVs is limited, how to maximize the processing of user tasks with limited battery capacity is also a challenging problem.

References

[1] G. Karagiannis et al., "Vehicular Networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 4, pp. 584–616, Fourthquarter 2011.

[2] Y. Chen, J. Xu, Y. Wu, J. Gao, and L. Zhao, "Dynamic task offloading and resource allocation for NOMA-aided mobile edge computing: An energy efficient design," *IEEE Trans. Serv. Comput.*, early access, Mar. 12, 2024, doi: 10.1109/TSC.2024.3376240.

[3] M. Li, J. Gao, L. Zhao, and X. Shen, "Deep reinforcement learning for collaborative edge computing in vehicular networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1122–1135, Dec. 2020.

[4] Y. Chen, F. Zhao, X. Chen, and Y. Wu, "Efficient multi-vehicle task offloading for mobile edge computing in 6G networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 5, pp. 4584–4595, May 2022.

[5] W. Qi, Q. Song, L. Guo, and A. Jamalipour, "Energy-efficient resource allocation for UAV-assisted vehicular networks with spectrum sharing," *IEEE Trans. Veh. Technol.*, vol. 71, no. 7, pp. 7691–7702, Jul. 2022.

[6] J. Huang, F. Liu, and J. Zhang, "Multi-dimensional QoS evaluation and optimization of mobile edge computing for IoT: A survey," *Chin. J. Electron.*, vol. 33, no. 4, pp. 859–874, 2024.

[7] J. Lu et al., "SIC-STIA-IS: An interference management scheme for the UAV-assisted heterogeneous network," in *Proc. IEEE 2023 Int. Conf. Commun.*, 2023, pp. 672–678.

[8] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE 2018 Conf. Comput. Commun.*, 2018, pp. 207–215.

[9] Z. Ning et al., "Joint computing and caching in 5G-envisioned internet of vehicles: A deep reinforcement learning-based traffic control system," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5201–5212, Aug. 2021.

[10] A. Ndikumana, N. H. Tran, D. H. Kim, K. T. Kim, and C. S. Hong, "Deep learning based caching for self-driving cars in multi-access edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 5, pp. 2862–2877, May 2021.

[11] Y. Chen, K. Li, Y. Wu, J. Huang, and L. Zhao, "Energy efficient task offloading and resource allocation in air-ground integrated MEC systems: A distributed online approach," *IEEE Trans. Mobile Comput.*, vol. 23, no. 8, pp. 8129–8142, Aug. 2024.

[12] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9441–9455, Oct. 2020.

[13] J. Ji, K. Zhu, and L. Cai, "Trajectory and communication design for cache-enabled UAVs in cellular networks: A deep reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 6190–6204, Oct. 2023.

[14] L. Zhao, Y. Ran, H. Wang, J. Wang, and J. Luo, "Towards cooperative caching for vehicular networks with multi-level federated reinforcement learning," in *Proc. IEEE 2021 Int. Conf. Commun.*, 2021, pp. 1–6.

[15] M. Zhu, X.-Y. Liu, and A. Walid, "Deep reinforcement learning for unmanned aerial vehicle-assisted vehicular networks," 2019, *arXiv:1906.05015*.

[16] Y. Qu et al., "Service provisioning for UAV-enabled mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 11, pp. 3287–3305, Nov. 2021.

[17] X. Wang, H. Shi, Y. Li, Z. Qian, and Z. Han, "Energy efficiency resource management for D2D-NOMA enabled network: A Dinkelbach combined twin delayed deterministic policy gradient approach," *IEEE Trans. Veh. Technol.*, vol. 72, no. 9, pp. 11756–11771, Sep. 2023.

[18] J. Huang et al., "Incentive mechanism design of federated learning for recommendation systems in MEC," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 2596–2607, Feb. 2024.

[19] X. Li, L. Lu, W. Ni, A. Jamalipour, D. Zhang, and H. Du, "Federated multi-agent deep reinforcement learning for resource allocation of vehicle-to-vehicle communications," *IEEE Trans. Veh. Technol.*, vol. 71, no. 8, pp. 8810–8824, Aug. 2022.

[20] Z. Wang, Q. Hu, Z. Xiong, Y. Liu, and D. Niyato, "Resource optimization for blockchain-based federated learning in mobile edge computing," *IEEE Internet Things J.*, vol. 11, no. 9, pp. 15166–15178, May 2024.

[21] S. Liu, Y. Liu, L. M. Ni, J. Fan, and M. Li, “Towards mobility-based clustering,” in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2010, pp. 919–927.

[22] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane, and Y. Liu, “Multi-UAV-enabled load-balance mobile-edge computing for IoT networks,” *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6898–6908, Aug. 2020.

[23] H. Peng and X. Shen, “Multi-agent reinforcement learning based resource management in MEC-and UAV-assisted vehicular networks,” *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 131–141, Jan. 2021.

[24] W. Chen, B. Liu, H. Huang, S. Guo, and Z. Zheng, “When UAV swarm meets edge-cloud computing: The QoS perspective,” *IEEE Netw.*, vol. 33, no. 2, pp. 36–43, Mar./Apr. 2019.

[25] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, and G. Y. Li, “Joint offloading and trajectory design for UAV-enabled mobile edge computing systems,” *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1879–1892, Apr. 2019.

[26] Y. Du, K. Yang, K. Wang, G. Zhang, Y. Zhao, and D. Chen, “Joint resources and workflow scheduling in UAV-enabled wirelessly-powered MEC for IoT systems,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 10, pp. 10187–10200, Oct. 2019.

[27] C. Sun, W. Ni, and X. Wang, “Joint computation offloading and trajectory planning for UAV-assisted edge computing,” *IEEE Trans. Wireless Commun.*, vol. 20, no. 8, pp. 5343–5358, Aug. 2021.

[28] X. Hu, K.-K. Wong, K. Yang, and Z. Zheng, “UAV-assisted relaying and edge computing: Scheduling and trajectory optimization,” *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4738–4752, Oct. 2019.

[29] L. Zhang, B. Jabbari, and N. Ansari, “Deep reinforcement learning driven UAV-assisted edge computing,” *IEEE Internet Things J.*, vol. 9, no. 24, pp. 25449–25459, Dec. 2022.

[30] A. M. Seid, G. O. Boateng, B. Mareri, G. Sun, and W. Jiang, “Multi-agent DRL for task offloading and resource allocation in multi-UAV enabled IoT edge network,” *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4531–4547, Dec. 2021.

[31] Y. Chen, D. Pi, S. Yang, Y. Xu, J. Chen, and A. W. Mohamed, “HNIO: A hybrid nature-inspired optimization algorithm for energy minimization in UAV-assisted mobile edge computing,” *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 3, pp. 3264–3275, Sep. 2022.

[32] H. Ke, H. Wang, W. Sun, and H. Sun, “Adaptive computation offloading policy for multi-access edge computing in heterogeneous wireless networks,” *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 1, pp. 289–305, Mar. 2022.

[33] N. Lin, H. Qin, J. Shi, and L. Zhao, “Deep reinforcement learning empowered multiple UAVs-assisted caching and offloading optimization in D2D wireless networks,” in *Proc. 19th ACM Int. Conf. Comput. Front.*, 2022, pp. 150–158.

[34] Z. Wang, T. Zhang, Y. Liu, and W. Xu, “Deep reinforcement learning for caching placement and content delivery in UAV NOMA networks,” in *Proc. IEEE 2020 Int. Conf. Wireless Commun. Signal Process.*, 2020, pp. 406–411.

[35] S. Anokye, D. Ayepah-Mensah, A. M. Seid, G. O. Boateng, and G. Sun, “Deep reinforcement learning-based mobility-aware UAV content caching and placement in mobile edge networks,” *IEEE Syst. J.*, vol. 16, no. 1, pp. 275–286, Mar. 2022.

[36] C. Wu, S. Shi, S. Gu, L. Zhang, and X. Gu, “Deep reinforcement learning-based content placement and trajectory design in urban cache-enabled UAV networks,” *Wireless Commun. Mobile Comput.*, vol. 2020, pp. 1–11, 2020.

[37] D. Wang, Q. Liu, J. Tian, Y. Zhi, J. Qiao, and J. Bian, “Deep reinforcement learning for caching in D2D-enabled UAV-relaying networks,” in *Proc. IEEE/CIC 2021 Int. Conf. Commun. China*, 2021, pp. 635–640.

[38] Y. Chen, J. Hu, J. Zhao, and G. Min, “QoS-aware computation offloading in LEO satellite edge computing for IoT: A game-theoretical approach,” *Chin. J. Electron.*, vol. 33, no. 4, pp. 875–885, 2024.