



ARTICLE

# Value-of-Information Driven Distributed Offloading Decision for UAVs: A Behavior Prediction-Based Deep Reinforcement Learning Approach

Zengqi Ma and Wendong Zhao\*

College of Communications Engineering, Army Engineering University of PLA, Nanjing, China

\*Corresponding Author: Wendong Zhao. Email: zhaowendongtx@163.com

Received: 27 June 2026; Accepted: 26 August 2026; Published: 22 October 2026

**ABSTRACT:** In large-scale monitoring scenarios with multiple unmanned aerial vehicles (UAVs) and offloading nodes, concurrent offloading of collected data to the same node within a short time window will cause excessive node load, prolonged processing delay, and degraded information timeliness. To address this issue, this paper considers the impact of users' personalized demands on information timeliness and introduces the concept of Value of Information (VoI). A user demand-oriented dynamic VoI model is established to accurately describe and evaluate information timeliness. The multi-UAV offloading node selection problem is formulated as a semi-Markov decision process (SMDP), and a Behavior Prediction-based Distributed Offloading Decision (BP-DOD) algorithm is proposed. By predicting UAV offloading behaviors and constructing conflict costs, the algorithm explicitly characterizes the competition for computing resources caused by the mismatch between the generation order and execution order of different UAV decisions, thereby significantly improving convergence stability. Furthermore, Dueling DQN, Double DQN and reward shaping techniques are integrated to further improve algorithm performance. Simulation results show that compared with benchmark algorithms, the proposed algorithm can increase the average VoI by up to 0.777 and reduce the information invalidation rate by up to 86.6 percentage points, and exhibits favorable adaptability across different scales of UAVs and edge nodes.

**KEYWORDS:** Unmanned aerial vehicle; value of information; multi-access edge computing; user demand; deep reinforcement learning

## 1 Introduction

Benefiting from three-dimensional mobility, rapid deployment capability and line-of-sight communication characteristics, unmanned aerial vehicles (UAVs) offer distinct advantages in improving information acquisition efficiency of users [1], and have been widely applied in intelligent transportation [2], precision agriculture [3], disaster monitoring [4] and other fields. However, in time-sensitive scenarios with high real-time response requirements and wide regional coverage, such as battlefield information monitoring and emergency rescue, if UAVs transport raw data back to end users after completing information collection, the timeliness of information may fail to meet user demands. Multi-access Edge Computing (MEC) technology [5] deploys computing resources at edge nodes close to information sources, enabling UAVs to offload information nearby for processing and then forward the results to users, which better ensures information timeliness while improving transmission efficiency [6].

Due to the limited processing capacity of edge nodes, concurrent offloading of multiple UAVs to the same node within a short time interval will lead to increased node load and processing delay, thus

impairing information timeliness [7]. To address this issue, researchers have proposed various offloading algorithms. Early studies addressed this problem from the perspectives of convex optimization and game theory. Ma et al. [8] conduct game-theoretic analysis on the offloading problem of multi-user multi-access points to minimize delay and energy consumption. Cao et al. [9] study the scenario where UAVs fly between a given start point and a destination, offloading computation tasks to multiple base stations along the flight path. The optimization objective is to minimize task completion time by jointly optimizing trajectory and offloading scheduling. Dai et al. [10] determine the optimal offloading node by comprehensively considering computation, transmission and delay overhead for multiple UAVs performing sensing and computation tasks.

In recent years, deep reinforcement learning (DRL) has been widely applied to offloading node selection problem due to its strong adaptability to complex dynamic environments. Zhang et al. [11] comprehensively consider quality of service and delay, model them as a utility function, and maximize long-term cumulative utility by jointly selecting offloading computation, data volume, and target edge node. Sun et al. [12] investigate joint service caching and computation offloading in mobile edge-cloud computing networks, with the goal of minimizing the average service delay. Vidya and Gopalakrishnan [13] focus on delay-sensitive tasks and dynamic edge load characteristics, and minimize energy consumption and transmission delay. Tang and Wong [14] model delay and task dropping as costs, and minimize the long-term expected cost by observing the dynamic load of edge nodes. Rahmaty et al. [15] model task completion rate, task delay and energy consumption as quality of experience, and maximize quality of experience by optimizing offloading decisions. Wang et al. [16] propose a deep reinforcement learning-based computation task offloading algorithm, which allows UAVs to determine offloading decisions according to the information of arriving tasks. Deng et al. [17] propose an offloading strategy based on game theory and reinforcement learning for UAV swarms performing emergency fault inspection, to select edge servers with the objective of minimizing energy consumption and delay. Sacco et al. [18] propose a distributed task offloading method based on multi-agent reinforcement learning, which leverages information from peer devices to solve the computation offloading optimization problem in real time, so as to minimize task completion time and energy consumption in UAV networks.

However, the above studies generally assume that offloading devices can obtain all kinds of real-time information required for current offloading and determine the offloading node accordingly. In the scenario where multiple UAVs offload information to edge nodes, different UAVs have different offloading behaviors such as offloading timing and data volume. It is difficult for UAVs to know each other's offloading behaviors or perceive the real-time status of nodes, rendering existing offloading algorithms insufficient to fully address the node selection problem in such scenarios.

Regarding the quantification of information timeliness, most existing studies [19–24] focus on reducing the end-to-end delay from information collection to delivery. In particular, studies in [23,24] further investigate this problem from the perspective of Age of Information (AoI) [25]. Liu et al. [23] study the scenario where UAVs fly among sensor nodes and collect status updates, and minimize AoI by jointly optimizing UAV trajectory and sensor node scheduling. Liu et al. [24] propose a clustering-based data collection strategy, which jointly optimizes collection point locations, node association, upload sequence and flight trajectory to minimize the maximum and average AoI, respectively.

Nevertheless, all the above studies quantify information timeliness solely from the temporal dimension, while information timeliness is not only determined by time, but also closely related to users' personalized demands [26]. For example, if users need data from a specific historical moment, information timeliness does not change with delay; if users need real-time status data, the shorter the transmission and processing delay, the higher the timeliness, and the information may become completely invalid after the deadline. Delay

alone cannot accurately reflect such demand-oriented timeliness differences. By introducing the concept of Value of Information (VoI) [27], information timeliness can be described and evaluated more precisely.

Currently, a few studies have applied VoI to data collection scenarios. Abbas et al. [28] propose a VoI-aware data collection scheme, which deploys the minimum number of UAVs to collect the most valuable data and minimize energy consumption under delay and coverage constraints. Jing et al. [29] study multi-UAV cooperative data collection and maximize total VoI through path planning. However, the above works only discuss the data collection problem of UAVs, and have not extended the VoI metric to the full workflow of data offloading, edge processing, and end-user delivery. For large-scale time-sensitive monitoring scenarios, edge processing and forwarding are essential to preserve information timeliness. A systematic comparison of the above related works is summarized in Table 1, covering optimization objectives, timeliness quantification metric, consideration of user demands, and adopted methodologies.

**Table 1:** Comparison of related works.

| Paper       | Optimization Objective                       | Information Timeliness |                       | Method                   |
|-------------|----------------------------------------------|------------------------|-----------------------|--------------------------|
|             |                                              | User Demand            | Quantification Metric |                          |
| [8]         | Delay and energy consumption                 | ×                      | Delay                 | Game theory              |
| [9]         | Task completion time                         | ×                      | Delay                 | Convex optimization      |
| [10]        | Computation, transmission and delay overhead | ×                      | Delay                 | Game theory              |
| [11]        | Quality of service and delay                 | ×                      | Delay                 | Offline RL (MDP)         |
| [12]        | Service delay                                | ×                      | Delay                 | DQN and SAC (MDP)        |
| [13]        | Energy consumption and delay                 | ×                      | Delay                 | Adaptive DRL (MDP)       |
| [14]        | Delay and task dropping                      | ×                      | Delay                 | DRL (MDP)                |
| [15]        | Task completion rate, delay and energy       | ×                      | Delay                 | DRL (MDP)                |
| [16]        | Delay and energy consumption                 | ×                      | Delay                 | DQN (MDP)                |
| [17]        | Energy consumption and delay                 | ×                      | Delay                 | Game theory and RL (MDP) |
| [18]        | Task completion time and energy              | ×                      | Delay                 | MARL (MDP)               |
| <b>Ours</b> | <b>VoI</b>                                   | ✓                      | <b>VoI</b>            | <b>DRL (SMDP)</b>        |

To address the above limitations, this paper further investigates the offloading node selection problem after UAV information collection, considering the impact of users' personalized demands on information timeliness. Due to diverse UAV offloading behaviors, complex interaction with nodes, and difficulty in obtaining global offloading information in real time, it is difficult for traditional optimization methods to yield effective solutions. Deep reinforcement learning integrates reinforcement learning with deep neural networks, which can not only handle high-dimensional state spaces, but also adapt flexibly to complex dynamic system environments and learn policies that optimize long-term system performance. On this basis, a distributed offloading decision algorithm based on UAV behavior prediction is proposed, which enables

nodes to determine appropriate offloading targets for the next collection task without requiring real-time global state information. The main contributions are as follows:

- A user-demand-oriented information value model is constructed. Considering the differences in information timeliness caused by users' personalized demands for different target areas, the concept of Value of Information (VoI) is introduced to accurately describe and evaluate information timeliness. Meanwhile, the offloading problem is formulated as a semi-Markov decision process (SMDP) to accommodate irregular decision intervals.
- A Behavior Prediction-based Distributed Offloading Decision (BP-DOD) algorithm is proposed. The algorithm employs LSTM to predict UAV offloading behaviors and combines the predictions with pending decisions to construct forward and backward conflict costs. This mechanism explicitly characterizes the competition for computing resources caused by the mismatch between the generation order and execution order of different UAV decisions, thereby significantly improving convergence stability.
- Double DQN and Dueling DQN techniques are adopted to reduce Q-value overestimation and improve learning efficiency. With the help of reward shaping technology, nodes obtain richer reward signals during learning, which accelerates policy convergence.
- Simulation results show that the proposed algorithm can effectively improve the average VoI of information and reduce the information invalidation rate, and exhibits favorable adaptability across different scales of UAVs and edge nodes.

For the convenience of reading, the key notations used throughout this paper are summarized in [Table 2](#).

**Table 2:** Summary of key notations.

| Notation                 | Description                                                                                     |
|--------------------------|-------------------------------------------------------------------------------------------------|
| $\mathcal{M}$            | Set of UAVs                                                                                     |
| $\mathcal{N}$            | Set of edge nodes                                                                               |
| $\mathcal{T}$            | Set of time slots                                                                               |
| $k_m(j)$                 | The $j$ -th information collected by UAV $m$                                                    |
| $t_m^{\text{col}}(j)$    | Time slot when UAV $m$ completes collecting $k_m(j)$                                            |
| $\lambda_m(j)$           | Data size of information $k_m(j)$                                                               |
| $I_m^{\text{col}}(j)$    | Collection duration of information $k_m(j)$                                                     |
| $\lambda_{m,n}(t)$       | Data size offloaded by UAV $m$ to the dedicated queue of node $n$ at the start of time slot $t$ |
| $q_{m,n}(t)$             | Length of the dedicated queue for UAV $m$ at node $n$ by the end of time slot $t$               |
| $B_n(t),  B_n(t) $       | Set of active queues at edge node $n$ in time slot $t$ and its cardinality                      |
| $t_m^{\text{dlv}}(j)$    | Time slot when information $k_m(j)$ is fully received by the user                               |
| $t_m^{\text{off}}(j)$    | Time slot when information $k_m(j)$ starts offloading                                           |
| $I_m^{\text{off}}(j-1)$  | Interval time slots between the $j$ -th offloading and the $(j-1)$ -th offloading               |
| $H_m(j)$                 | Historical offloading record of length $W$ maintained for UAV $m$ after the $j$ -th offloading  |
| $s_m(j), a_m(j), r_m(j)$ | State, action, and reward corresponding to the $j$ -th decision of UAV $m$                      |

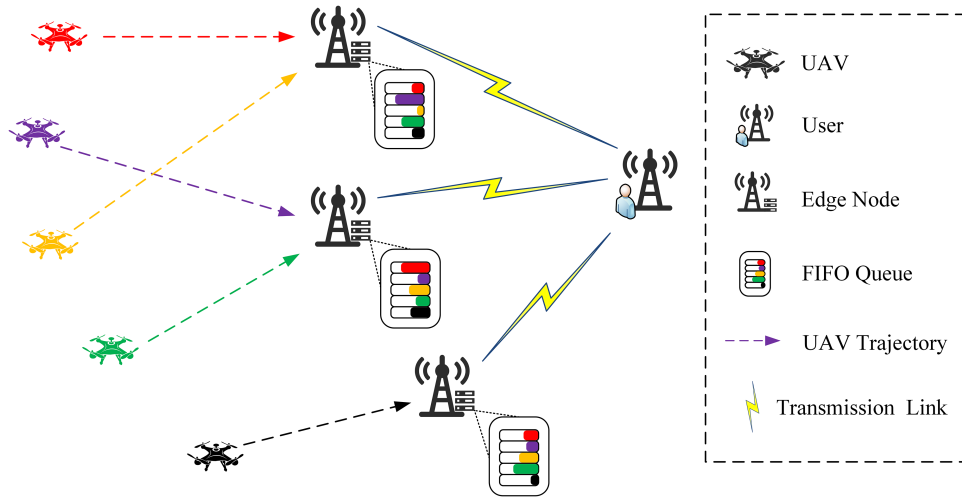
(Continued)

**Table 2 (continued)**

| Notation                                           | Description                                                                                                                         |
|----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| $G_m$                                              | Long-term reward of UAV $m$ over the entire task                                                                                    |
| $t_j$                                              | Cumulative interval time slots between the $(j+1)$ -th offloading and the first offloading                                          |
| $\Omega$                                           | Pending decision set                                                                                                                |
| $\xi(m')$                                          | Pending decision information of UAV $m'$ , including target offloading node, predicted offloading time slot and predicted data size |
| $\Gamma_m^{\text{ea}}(n), \Gamma_m^{\text{la}}(n)$ | Forward conflict cost and backward conflict cost                                                                                    |
| $e_m(j)$                                           | A complete experience sample obtained after information $k_m(j+1)$ is fully delivered                                               |

## 2 System Model

As shown in Fig. 1, this paper considers an offloading node selection scenario after multi-UAV information collection. A set of UAVs denoted by  $\mathcal{M} = \{1, 2, \dots, M\}$  undertakes periodic information collection tasks over the target area. After each collection, UAV  $m \in \mathcal{M}$  flies from its target location  $\omega_m = (x_m, y_m)$  to an edge node in set  $\mathcal{N} = \{1, 2, \dots, N\}$  for offloading. Each edge node  $n \in \mathcal{N}$  processes the offloaded information and sends the result to user  $u$ .

**Figure 1:** Offloading node selection scenario for multi-UAV information collection.

The entire task execution process is discretized into a sequence of time slots  $\mathcal{T} = \{1, 2, \dots, T\}$ , each with a duration of  $\delta$ . The coordinates of edge node  $n$  and user  $u$  are  $\omega_n = (x_n, y_n)$  and  $\omega_u = (x_u, y_u)$ , respectively. The following assumptions are adopted: each piece of information is indivisible and can only be offloaded to a single node; all UAVs fly at a constant speed  $V$  and maintain a fixed altitude; state updates only occur at the boundaries of time slots.

## 2.1 Value of Information Model

UAV  $m$  completes its  $j$ -th ( $j \in \mathbb{N}^+$ ) information collection at the target area at the beginning of time slot  $t_m^{\text{col}}(j) \in \mathcal{T}$ , obtains information  $k_m(j)$ , and then flies to edge node  $n_m(j)$  for offloading. The data size of information  $k_m(j)$  is given by  $\lambda_m(j) = \beta I_m^{\text{col}}(j)$ , where  $\beta$  is the data volume coefficient and  $I_m^{\text{col}}(j)$  is the collection duration. Let  $\tau_m(j)$  denote the number of time slots from the completion of information collection to its full reception by the user. The value of information is defined as

$$\text{VoI}(k_m(j)) = \begin{cases} e^{-\varphi_m \tau_m(j)}, & 0 \leq \tau_m(j) \leq \tau_m^{\max}, \\ 0, & \tau_m(j) > \tau_m^{\max}. \end{cases} \quad (1)$$

where  $\varphi_m$  is the attenuation coefficient, which reflects the user's sensitivity to information timeliness;  $\tau_m^{\max}$  is the delivery deadline (measured in time slots), and information is regarded as invalid if  $\tau_m(j) > \tau_m^{\max}$ .  $\varphi_m$  and  $\tau_m^{\max}$  jointly characterize the user's personalized demands. When the user requires real-time situational awareness, the value of information decays rapidly with delay, corresponding to a larger  $\varphi_m$  and a smaller  $\tau_m^{\max}$ . When the user focuses on data at a specific historical moment, the value of information is insensitive to delay, corresponding to a smaller  $\varphi_m$  and a larger  $\tau_m^{\max}$ . By adjusting  $\varphi_m$  and  $\tau_m^{\max}$ , the model can flexibly reflect the impact of different user demands on information timeliness.

## 2.2 Edge Node Processing Model

Each edge node maintains a dedicated First In, First Out (FIFO) queue for each UAV. It is assumed that once a UAV arrives at an edge node in a given time slot, its carried information starts offloading immediately at the beginning of the next time slot.

Let  $\lambda_{m,n}(t)$  denote the data volume offloaded by UAV  $m$  to the dedicated queue of node  $n$  at the start of time slot  $t$ , and  $q_{m,n}(t)$  (in Mbit) denote the length of the corresponding queue at the end of time slot  $t$ . A queue is defined as active if it is non-empty in a specific time slot. Let  $B_n(t)$  denote the set of active queues at edge node  $n$  in time slot  $t$ , and  $|B_n(t)|$  denote the number of active queues. Each node shares its current number of active queues with other nodes at the beginning of every time slot. The minimum number of bits required to share the number of active queues among all nodes is  $\lceil \log_2(M+1) \rceil$ .

$$B_n(t) = \{m \mid m \in \mathcal{M}, \lambda_{m,n}(t) > 0 \text{ or } q_{m,n}(t-1) > 0\}. \quad (2)$$

In each time slot  $t$ , the total processing capacity  $f_n$  of node  $n$  is equally allocated among all active queues. Accordingly, the queue length  $q_{m,n}(t)$  is updated as

$$q_{m,n}(t) = \left[ q_{m,n}(t-1) + \lambda_{m,n}(t) - \frac{f_n \cdot \delta}{\rho \cdot |B_n(t)|} \right]^+, \quad (3)$$

where the operator  $[\cdot]^+ = \max\{0, \cdot\}$ , and  $\rho$  is the information processing density representing the number of computing cycles required to process per unit data. For information waiting sequentially in the same queue, after the previous piece of information is fully processed within a time slot, the next piece starts processing at the beginning of the next time slot. The ratio of the processed result data size to the original data size is denoted as  $\zeta$ .

### 2.3 Transmission Model

Line-of-sight channels are assumed for both UAV-to-node and node-to-user links. The corresponding transmission rates are given by

$$r_{m,n} = B_{m,n} \log_2 \left( 1 + \frac{g_{m,n} p_m}{N_0 B_{m,n}} \right), \quad (4)$$

$$r_{n,u} = B_{n,u} \log_2 \left( 1 + \frac{g_{n,u} p_n}{N_0 B_{n,u}} \right). \quad (5)$$

where  $B_{m,n}$  and  $B_{n,u}$  are the communication bandwidths of the UAV  $m$ -node  $n$  link and the node  $n$ -user  $u$  link, respectively;  $g_{m,n}$  and  $g_{n,u}$  are the corresponding channel gains;  $N_0$  is the noise power spectral density;  $p_m$  and  $p_n$  are the transmission powers of UAV  $m$  and node  $n$ , respectively.

For information  $k_m(j)$  acquired at the beginning of time slot  $t_m^{\text{col}}(j)$ , its offloading start time slot can be expressed as

$$t_m^{\text{off}}(j) = t_m^{\text{col}}(j) + \left\lceil \frac{\|\omega_m - \omega_{n_m(j)}\|}{V\delta} \right\rceil + \left\lceil \frac{\lambda_m(j)}{r_{m,n_m(j)}\delta} \right\rceil, \quad (6)$$

where  $\lceil \cdot \rceil$  denotes the ceiling function. The processing result of  $k_m(j)$  will be delivered to the user after  $\lceil \lambda_m(j)/(r_{n,u}\delta) \rceil$  additional time slots. Let  $t_m^{\text{dlv}}(j)$  denote the time slot when  $k_m(j)$  is fully received by the user. Then the total end-to-end delay of information  $k_m(j)$  is

$$\tau_m(j) = t_m^{\text{dlv}}(j) - t_m^{\text{col}}(j). \quad (7)$$

### 3 Offloading Problem Formulation

In the considered scenario, since UAVs cannot know each other's offloading behaviors or perceive node status in real time, they are no longer suitable as decision-makers and only act as execution entities. Specifically, whenever a UAV offloads information to an edge node, the node observes the current state and plans an offloading decision for it, so as to maximize the VoI of information received by users. However, since UAVs carry and transmit information, if the node plans the offloading decision based on the current state after the UAV arrives at the node, the VoI of information will be further attenuated during transfer between nodes. Therefore, the node needs to plan the offloading decision for the information currently carried by the UAV in advance, so that the UAV can fly directly to the target node after collecting the information. In other words, the offloading decision planned by the node currently does not apply to the information currently carried by the UAV, but serves as the target offloading node for its next collected information. This means that the decision will not be executed immediately, and takes effect only after the UAV completes the next collection and arrives at the target node. For this reason, the Semi-Markov Decision Process (SMDP) is adopted to model the problem [30].

Generally, a Markov Decision Process (MDP) can be represented by a 5-tuple  $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$ , where  $\mathcal{S}$  is the state space,  $\mathcal{A}$  is the action space,  $P(s' | s, a)$  is the state transition probability that depends only on the previous state  $s \in \mathcal{S}$  and action  $a \in \mathcal{A}$ ,  $r$  is the reward received after taking action  $a$  in state  $s$ , and  $\gamma$  is the discount factor that balances the influence of immediate and future rewards. On this basis, the Semi-Markov Decision Process (SMDP) introduces the action duration distribution  $F(d | s, a)$ , where  $d$  denotes the duration of executing action  $a$  in state  $s$ , and can be formulated as a 6-tuple  $(\mathcal{S}, \mathcal{A}, P, r, F, \gamma)$ . Unlike MDP, the state transition probability of SMDP depends on both the previous state-action pair and the action duration, denoted as  $P(s', d | s, a)$ . Therefore, SMDP does not require the system to satisfy the Markov

property at every time instant as in MDP; instead, the Markov property holds only at state transition instants. Meanwhile, to account for the impact of action duration on rewards, the discount factor  $\gamma$  in SMDP acts as a per-time-slot discount factor, and the discount weight between two states is given by  $\gamma^d$ .

In the considered scenario, the duration between different decisions of each UAV is not fixed, making MDP unsuitable for problem formulation. SMDP allows variable decision execution intervals and is well suited for offloading selection problems with irregular decision intervals.

### 3.1 State

Node  $n_m(j)$  receives information  $k_m(j)$  at the beginning of time slot  $t_m^{\text{off}}(j)$ , and then shares the data volume  $\lambda_m(j)$  of information  $k_m(j)$  and the interval time slots  $I_m^{\text{off}}(j-1)$  elapsed since the last offloading among all nodes. Each node can locally store the offloading records of UAV  $m$  at all nodes. Let  $H_m(j)$  denote the historical offloading record of length  $W$  maintained by each node for UAV  $m$  after the  $j$ -th offloading:

$$H_m(j) = [x_m(j-W+1), x_m(j-W+2), \dots, x_m(j)], \quad (8)$$

where each offloading record  $x_m(i)$  is defined as

$$x_m(i) = (n_m(i), I_m^{\text{off}}(i-1), \lambda_m(i)). \quad (9)$$

Here,  $n_m(i)$  denotes the offloading node,  $I_m^{\text{off}}(i-1)$  denotes the interval time slots between the current offloading and the last offloading, and  $\lambda_m(i)$  denotes the offloaded data volume. The minimum number of bits required to share a single offloading record  $x_m(i)$  among nodes is  $\lceil \log_2 N \rceil + \lceil \log_2 T \rceil + \lceil \log_2 \beta T \rceil$ .

The VoI of information received by users is affected by multiple factors. First, due to diverse user demands, information collected by different UAVs has different attenuation coefficients and delivery deadlines. Second, the varying load levels and processing capacities of edge nodes also lead to different degrees of VoI degradation. Third, edge nodes need to perceive UAV behavior patterns from historical offloading records to effectively plan the target offloading node for the next information collection task. Therefore, at the beginning of time slot  $t_m^{\text{off}}(j)$ , node  $n_m(j)$  identifies the identity of the UAV and observes the attributes of information  $k_m(j)$ , node status, and historical offloading records. Specifically, the state observed by node  $n_m(j)$  is defined as

$$s_m(j) = \left( m, \varphi_m, \tau_m^{\max}, q_{m,n_m(j)}(t_m^{\text{off}}(j)-1), \{ |B_n(t_m^{\text{off}}(j)-1)| \}_{n \in \mathcal{N}}, \{ f_n \}_{n \in \mathcal{N}}, H_m(j) \right) \quad (10)$$

where  $m$  denotes the identity of the UAV;  $q_{m,n_m(j)}(t_m^{\text{off}}(j)-1)$  represents the length of the dedicated queue maintained by node  $n_m(j)$  for UAV  $m$ ;  $\{ |B_n(t_m^{\text{off}}(j)-1)| \}_{n \in \mathcal{N}}$  and  $\{ f_n \}_{n \in \mathcal{N}}$  indicate the number of active queues and the processing capacity of each edge node, respectively. In addition, the processing capacity  $\{ f_n \}_{n \in \mathcal{N}}$  of all nodes can be determined upon node deployment and does not need to be shared among nodes.

### 3.2 Action and Duration

Let action  $a_m(j) \in \mathcal{N}$  denote the offloading decision generated by node  $n_m(j)$  for UAV  $m$ . This action does not take effect for information  $k_m(j)$ , but serves as the target offloading node for the next collected information  $k_m(j+1)$ . The first offloading of a UAV defaults to the node closest to its target area.

Action  $a_m(j)$  will not be executed immediately after generation, and takes effect only when information  $k_m(j+1)$  is offloaded to the corresponding node  $a_m(j)$ . The duration of action  $a_m(j)$  can be expressed as

$$I_m^{\text{off}}(j) = t_m^{\text{off}}(j+1) - t_m^{\text{off}}(j). \quad (11)$$

### 3.3 Reward

After node  $n_m(j)$  generates action  $a_m(j)$ , UAV  $m$  takes it as the target offloading node for information  $k_m(j+1)$ . Since the goal is to maximize the VoI of information received by users, the reward  $r_m(j)$  of this action can be determined by the VoI of information  $k_m(j+1)$ .

Nevertheless, if reward  $r_m(j)$  is only determined by the VoI of information  $k_m(j+1)$ , it is difficult for nodes to pay effective attention to information with slow VoI attenuation. When such information accumulates in nodes, it will in turn affect the processing of other information. For this reason, reward shaping technology [31] is adopted to adjust the reward. Reward shaping is a method to guide agent learning by introducing additional task-related learnable reward signals, which can help agents better understand task goals, accelerate learning and improve learning effect. Accordingly, reward  $r_m(j)$  is constructed as

$$r_m(j) = \text{VoI}(k_m(j+1)) - \frac{\tau_m(j+1)}{\tau_m^{\max}}. \quad (12)$$

### 3.4 Optimization Objective

The long-term reward of UAV  $m$  during the entire task is defined as

$$G_m = \sum_{j \geq 1} \gamma^{t_j} r_m(j), \quad (13)$$

where  $t_j = \sum_{l=1}^j I_m^{\text{off}}(l) = t_m^{\text{off}}(j+1) - t_m^{\text{off}}(1)$  denotes the cumulative interval time slots between the  $(j+1)$ -th offloading and the first offloading of UAV  $m$ ;  $\gamma \in (0, 1]$  is the per-time-slot discount factor.

The offloading policy  $\pi_n[a_m(j) | s_m(j)]$  is defined as the probability distribution of taking action  $a_m(j)$  by node  $n$  given state  $s_m(j)$ . The optimization objective of this paper is to find the optimal policy  $\pi_n^*$  for each node  $n$  to maximize the long-term reward of the entire system, thereby maximizing the VoI of information received by users.

$$\pi_n^* = \arg \max_{\pi_n} \mathbb{E} \left[ \sum_{m \in \mathcal{M}} G_m | \pi_n \right], \quad (14)$$

where the expectation operator  $\mathbb{E}[\cdot]$  is used to handle the randomness of the system and the uncertainty of long-term rewards.

## 4 The Proposed BP-DOD Algorithm

Most existing deep reinforcement learning architectures are inherently built upon the MDP formulation, where states are observed instantly and actions take effect immediately. In contrast, the problem in this paper is formulated as an SMDP: the action generated by a node after observing the current state does not apply to the information currently carried by the UAV, but specifies the target offloading node for its next collected information. Therefore, what a node truly needs to evaluate during decision-making is which node is more suitable for processing the information when the UAV performs its next offloading, rather than which node has the lowest load at the current moment. Intuitively, offloading decisions can be made by predicting the node load at the next offloading instant. However, there exists a mutual interplay between node load and offloading decisions: node load not only affects decision-making, but the decisions themselves also reshape the future load distribution. Since different UAVs have different action execution times, the generation order of decisions may differ from their actual execution order. Therefore, when making an offloading decision, the effects of other UAV offloadings occurring both before and after the target UAV arrives should be jointly considered in evaluating information processing at the nodes.

To address this issue, this paper proposes a distributed offloading decision algorithm based on deep reinforcement learning (DRL). The algorithm is designed on the basis of the DQN architecture [32], which can learn through interaction with the environment without prior system modeling information and is capable of handling large-scale state spaces. Meanwhile, the DQN structure is concise and easy to extend. We modify the internal structure of DQN so that it not only adapts to the SMDP formulation of the considered scenario, but also explicitly characterizes the potential impact of offloading decisions made by other UAVs when the current UAV performs its next information offloading. In the algorithm, the Q-value represents the expected long-term reward corresponding to each state-action pair. Each node learns the mapping from states to Q-values of all actions via a neural network, and selects the action that maximizes the Q-value during decision-making, so as to maximize the long-term VoI of information received by users.

#### 4.1 Network Architecture

The overall neural network architecture of the edge node is illustrated in Fig. 2. State information is first fed into the network through the input layer: historical offloading records are used to predict future UAV offloading behaviors, while the remaining state features are directly sent to the fully connected layer. The prediction layer predicts the interval time slots and data volume of the UAV's next offloading to each node based on historical offloading records. The feature construction layer further calculates the conflict cost of each node according to the prediction results. The fully connected (FC) layer learns the mapping from state features to action Q-values. To improve learning efficiency, the Dueling DQN technique is introduced in the advantage-value layer. Finally, the Q-value of each action is obtained at the output layer. Let  $\theta_n$  denote the neural network parameter vector of node  $n$ . The detailed design of each layer is described as follows.

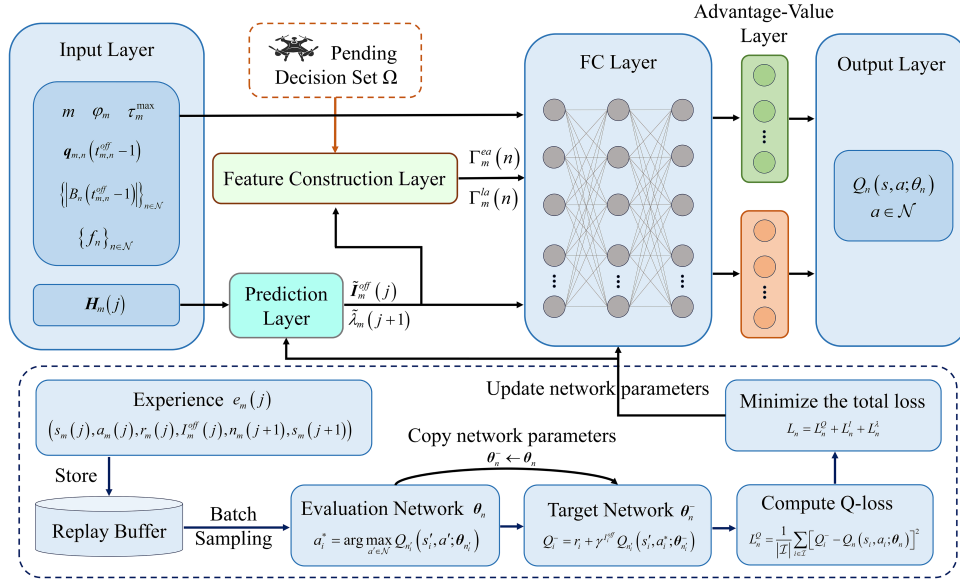


**Figure 2:** Neural network architecture of the edge node.

##### 4.1.1 Input Layer

The input layer receives state  $s_m(j)$  and delivers it to subsequent layers. The historical offloading record  $H_m(j)$  in the state is first passed to the prediction layer for learning UAV offloading patterns, while the rest of the state information is directly sent to the fully connected layer.

#### 4.1.2 Prediction Layer

The prediction layer is designed based on LSTM, which predicts the interval time slots and data volume of the UAV's next offloading to each node using the historical offloading records stored by the node. After node  $n_m(j)$  receives information  $k_m(j)$ , the offloading record is updated from  $H_m(j-1)$  to  $H_m(j)$ . Then, based on the offloading patterns in  $H_m(j)$ , the prediction layer estimates the interval time slots  $\tilde{I}_m^{\text{off}}(n, j)$  between the next offloading of UAV  $m$  to each node and the current time slot  $t_m^{\text{off}}(j)$ :

$$\tilde{I}_m^{\text{off}}(j) = (\tilde{I}_m^{\text{off}}(n, j))_{n \in \mathcal{N}}. \quad (15)$$

Let  $\tilde{\lambda}_m(j+1)$  denote the predicted data volume of the next offloading. Then the output of the prediction layer can be expressed as

$$\tilde{\mathbf{x}}_m(j+1) = (\tilde{I}_m^{\text{off}}(j), \tilde{\lambda}_m(j+1)). \quad (16)$$

#### 4.1.3 Feature Construction Layer

The offloading decision generated by the node does not take effect immediately, but is executed after the UAV completes the next information collection and arrives at the target node. For each UAV  $m'$  that has received an offloading decision but not yet executed it, the relevant decision information is defined as

$$\xi(m') = (m', a_{m'}(j'), \tilde{t}_{m'}^{\text{off}}(j'), \tilde{\lambda}_{m'}(j')), \quad (17)$$

where  $a_{m'}(j')$  denotes the target offloading node of UAV  $m'$ ,  $\tilde{t}_{m'}^{\text{off}}(j')$  denotes the predicted offloading time slot, and  $\tilde{\lambda}_{m'}(j')$  denotes the predicted offloading data volume. After generating an offloading decision, the node shares  $\xi(m')$  with other nodes, and all nodes maintain an identical pending decision set  $\Omega = \{\xi(m')\}$ . The minimum number of bits required to share each pending decision  $\xi(m')$  among nodes is  $\lceil \log_2 M \rceil + \lceil \log_2 N \rceil + \lceil \log_2 T \rceil + \lceil \log_2 \beta T \rceil$ .

When UAV  $m$  chooses to offload to node  $n$ , the UAVs in  $\Omega$  may arrive at node  $n$  earlier than, simultaneously with, or later than UAV  $m$ . The closer the arrival times are, the greater the impact on UAV  $m$ . Accordingly, the forward conflict cost  $\Gamma_m^{\text{ea}}(n)$  and backward conflict cost  $\Gamma_m^{\text{la}}(n)$  are defined to quantify the impact of UAVs in  $\Omega$ :

$$\Gamma_m^{\text{ea}}(n) = \sum_{\Omega} \mathbb{I}(a_{m'}(j') = n) \mathbb{I}(\tilde{t}_{m'}^{\text{off}}(j') \leq \tilde{t}_m^{\text{off}}(n, j+1)) \tilde{\lambda}_{m'}(j') \cdot \exp\left(\frac{\tilde{t}_{m'}^{\text{off}}(j') - \tilde{t}_m^{\text{off}}(n, j+1)}{\kappa}\right), \quad (18)$$

$$\Gamma_m^{\text{la}}(n) = \sum_{\Omega} \mathbb{I}(a_{m'}(j') = n) \mathbb{I}(\tilde{t}_{m'}^{\text{off}}(j') > \tilde{t}_m^{\text{off}}(n, j+1)) \tilde{\lambda}_{m'}(j') \cdot \exp\left(\frac{\tilde{t}_m^{\text{off}}(n, j+1) - \tilde{t}_{m'}^{\text{off}}(j')}{\kappa}\right), \quad (19)$$

where  $\tilde{t}_m^{\text{off}}(n, j+1) = t_m^{\text{off}}(j) + \tilde{I}_m^{\text{off}}(n, j)$ ;  $\mathbb{I}(\cdot)$  is the indicator function, which equals 1 if the condition inside holds and 0 otherwise;  $\kappa$  is the scaling coefficient used to adjust the exponential decay rate.

The feature construction layer calculates the forward and backward conflict costs of each node based on the prediction layer output  $\tilde{\mathbf{x}}_m(j+1)$ . Its output can be expressed as

$$\Psi_m = [(\Gamma_m^{\text{ea}}(n), \Gamma_m^{\text{la}}(n))]_{n \in \mathcal{N}}. \quad (20)$$

#### 4.1.4 Fully Connected Layers

Three fully connected layers are responsible for learning the mapping from state features to action Q-values. Each fully connected layer consists of a set of neurons with the ReLU activation function, and each neuron is fully connected to all neurons in the preceding and subsequent layers.

#### 4.1.5 Advantage-Value Layer and Output Layer

The advantage-value layer and the output layer jointly implement the Dueling DQN technique [33] and output the Q-value of each action. Its core idea is to separately learn the state value (the part of Q-value determined by the state itself) and the action advantage value (the part of Q-value determined by action selection), and then combine them to compute the final Q-value, so as to improve estimation accuracy.

The advantage-value layer contains two fully connected subnetworks: the advantage network and the value network. For node  $n$ , the advantage network learns the advantage value  $A_n(s_m(j), a; \theta_n)$  for each action  $a \in \mathcal{N}$ , and the value network learns the state value  $V_n(s_m(j); \theta_n)$ . Based on the outputs of the advantage-value layer, the Q-value of action  $a \in \mathcal{N}$  under state  $s_m(j)$  at node  $n$  is computed as

$$Q_n(s_m(j), a; \theta_n) = V_n(s_m(j); \theta_n) + \left( A_n(s_m(j), a; \theta_n) - \frac{1}{|\mathcal{M}|} \sum_{a' \in \mathcal{N}} A_n(s_m(j), a'; \theta_n) \right), \quad (21)$$

where  $|\mathcal{M}|$  denotes the total number of candidate actions.

#### 4.2 Algorithm Training Process

Each node maintains an evaluation network and a target network separately to implement the Double DQN technique [34]. The evaluation network is used for offloading decision-making and parameter update, while the target network is dedicated to target Q-value calculation. The two networks share an identical architecture, with parameters denoted as  $\theta_n$  and  $\theta_n^-$ , respectively.

An  $\varepsilon$ -greedy policy is adopted during training to balance exploration and exploitation. The action selected by node  $n_m(j)$  for UAV  $m$  is given by

$$a_m(j) = \begin{cases} \text{a randomly selected node,} & \text{with probability } \varepsilon, \\ \arg \max_{a \in \mathcal{N}} Q_n(s_m(j), a; \theta_n), & \text{with probability } 1 - \varepsilon. \end{cases} \quad (22)$$

As training proceeds,  $\varepsilon$  is gradually decreased, so that the algorithm shifts from extensive exploration in the early stage to exploitation of the learned offloading policy. In addition, each node  $n$  maintains an experience replay buffer  $\mathcal{D}_n$ . A complete experience sample is obtained by the node once a piece of information is fully delivered:

$$e_m(j) = (s_m(j), a_m(j), r_m(j), I_m^{\text{off}}(j), n_m(j+1), s_m(j+1)). \quad (23)$$

When sufficient experiences are accumulated in the replay buffer, a mini-batch of samples  $\mathcal{I}$  is randomly drawn from  $\mathcal{D}_n$  for training. Random experience replay alleviates the temporal correlation among consecutive decision samples and improves the utilization efficiency of historical experiences. For the  $i$ -th sample in the mini-batch,  $s_i$ ,  $a_i$ ,  $r_i$ ,  $I_i^{\text{off}}$ ,  $n'_i$  and  $s'_i$  denote the current state, action, reward, action duration, next offloading node and next state, respectively.

Since the UAV may offload to a different node in the next step, the Q-value of the next state should be calculated by the network corresponding to node  $n'_i$ . To mitigate the Q-value overestimation problem caused

by using the same network for both action selection and evaluation in conventional DQN, the Double DQN method is adopted. First, the evaluation network of node  $n'_i$  is used to select the optimal action in state  $s'_i$ :

$$a_i^* = \arg \max_{a' \in \mathcal{N}} Q_{n'_i}(s'_i, a'; \theta_{n'_i}). \quad (24)$$

Then the target network of node  $n'_i$  is used to evaluate this action, and the target Q-value is obtained as

$$Q_i^- = r_i + \gamma^{I_i^{\text{off}}} Q_{n'_i}(s'_i, a_i^*; \theta_{n'_i}^-). \quad (25)$$

The Q-value loss function of node  $n$  is expressed as

$$L_n^Q = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} [Q_i^- - Q_n(s_i, a_i; \theta_n)]^2, \quad (26)$$

where  $|\mathcal{I}|$  is the batch size of samples.

In addition, the prediction layer needs to learn UAV offloading behaviors. The offloading interval prediction loss  $L_n^I$  and data volume prediction loss  $L_n^\lambda$  are calculated via mean squared error, respectively. The total loss function of node  $n$  is given by

$$L_n = L_n^Q + L_n^I + L_n^\lambda. \quad (27)$$

The node updates the parameters of the evaluation network and the prediction layer by minimizing  $L_n$ . The parameters of the evaluation network are copied to the target network every fixed number of updates:

$$\theta_n^- \leftarrow \theta_n. \quad (28)$$

The overall procedure of the proposed BP-DOD algorithm is summarized in Algorithm 1.

---

**Algorithm 1:** The proposed BP-DOD algorithm

---

**Input:** Node set  $\mathcal{N}$ , UAV set  $\mathcal{M}$ , discount factor  $\gamma$ , exploration probability  $\varepsilon$ , batch size, target network update period

**Output:** Offloading policy for each node

- 1: Initialize prediction network, evaluation network, target network, and replay buffer for each node; set  $\theta_n^- \leftarrow \theta_n$
  - 2: Initialize historical offloading records  $H_m$ , pending decision set  $\Omega$ , incomplete experience map
  - 3: **for** each training episode **do**
  - 4:   **while**  $k_m(j)$  is offloaded to node  $n_m(j)$  **do**
  - 5:     Update  $H_m$ , observe state  $s_m(j)$
  - 6:     Predict intervals  $\tilde{I}_m^{\text{off}}(j)$  and data size  $\tilde{\lambda}_m(j+1)$
  - 7:     Compute conflict costs using  $\Omega$
  - 8:     Select  $a_m(j)$  via  $\varepsilon$ -greedy policy
  - 9:     Add pending decision to  $\Omega$ , create an incomplete experience for  $k_m(j+1)$
  - 10:   **end while**
  - 11:   **while**  $k_m(j+1)$  is completely received by the user **do**
  - 12:     Compute reward  $r_m(j)$
  - 13:     Store the complete experience  $e_m(j)$  into  $\mathcal{D}_n$
  - 14:   **end while**
- 

(Continued)

**Algorithm 1 (continued)**


---

```

15: while training conditions are met do
16:   Sample mini-batch from  $\mathcal{D}_n$ 
17:   Compute target Q-values
18:   Compute  $L_n^Q, L_n^I, L_n^\lambda, L_n$ 
19:   Update  $\theta_n$  via gradient descent
20:   Periodically set  $\theta_n^- \leftarrow \theta_n$ 
21: end while
22: Decrease  $\varepsilon$  gradually
23: end for

```

---

**4.3 Complexity Analysis**

**Time complexity:** Let  $\mathcal{L}$  denote the number of multiplication operations in the neural network. For each training sample, the computation cost of forward propagation and backward propagation is both  $O(\mathcal{L})$ . Given a batch size of  $|Z|$ , the time complexity of a single training step is  $O(\mathcal{L}|Z|)$ . Let  $N^{\text{off}}$  denote the total number of decision events per episode and  $N^{\text{epi}}$  denote the number of training episodes. Then the time complexity of the entire training process is  $O(N^{\text{epi}}N^{\text{off}}\mathcal{L}|Z|)$ .

**Space complexity:** Let  $|\mathcal{D}_n|$  denote the size of the experience replay buffer. Each experience needs to store the current state, action, reward, action duration, next offloading node and next state, with a constant storage overhead. Therefore, the space complexity of a single node is  $O(|\mathcal{D}_n|)$ , and the total space complexity of all nodes is  $O(|\mathcal{N}||\mathcal{D}_n|)$ . The storage overhead of neural network parameters is relatively small and fixed, which can be ignored.

**5 Simulation Experiments and Performance Analysis**

The simulation is conducted in a  $2000 \times 2000 \text{ m}^2$  area with 25 UAVs and 2 edge nodes. The simulation parameters are set based on existing works [14,35]. We set 120 training episodes, each containing 12,000 time slots with a duration of 0.1 s per slot. The exploration factor  $\varepsilon$  linearly decays from 1 to 0.01, and the RMSProp optimizer is adopted. The length of historical offloading records  $W$  is 25, and the attenuation coefficient  $\varphi_m$  is selected from the set  $\{0.0, 0.0001, 0.0002, 0.0003, 0.0004\}$ . The user is located at (1000, 0), and the two edge nodes are located at (700, 400) and (1300, 400), respectively. To simulate diverse UAV offloading behaviors, the collection duration is set to four patterns—sinusoidal, triangular, sawtooth and random—as in [36], which also demonstrates that more complex patterns can be derived from combinations of these basic patterns. The remaining simulation parameters are listed in Table 3.

**Table 3:** Simulation parameters.

| Parameter Description                      | Value          | Parameter Description                 | Value          |
|--------------------------------------------|----------------|---------------------------------------|----------------|
| Flight speed $V$                           | 30 m/s         | Data volume coefficient $\beta$       | 0.3            |
| Communication bandwidth $B_{m,n}, B_{n,u}$ | 1 MHz, 1 MHz   | Information processing density $\rho$ | 0.397 GHz/Mbit |
| Channel power gain $g_{m,n}, g_{n,u}$      | −80 dB, −48 dB | Batch size                            | 128            |
| Noise power spectral density $N_0$         | −180 dBm/Hz    | Learning rate                         | 0.001          |
| Transmission power $p_m, p_n$              | 0.5 W, 20 W    | Discount factor $\gamma$              | 0.9995         |
| Processing capacity $f_n$                  | 2, 2.5 GHz     | Replay buffer size                    | 20,000         |
| Delivery deadline $\tau_m^{\max}$          | 800, 900, 1000 | LSTM cell                             | 64             |
| Result-to-original data ratio $\zeta$      | 0.1            | FC layer neurons                      | $128 \times 3$ |
| Scaling coefficient $\kappa$               | 200            |                                       |                |

All simulations are performed on a Windows 10 workstation equipped with a quad-core/eight-thread Intel CPU, 8 GB system memory and an NVIDIA GeForce MX250 GPU. The algorithm is implemented in Python 3.9.21 with TensorFlow 2.10.0, using PyCharm as the development platform.

### 5.1 Performance Comparison with Benchmark Algorithms

Three benchmark algorithms are selected for comparison: the random selection algorithm, the nearest node first algorithm, the DRL-based algorithm proposed in [13], and the Multi-Agent Proximal Policy Optimization (MAPPO) algorithm. The evaluation metrics include average VoI, information invalidation rate, average node residence time (the duration from information offloading to completion of processing), and average number of active queues per node. All results are statistical averages of 30 independent experiments under the same scenario.

Fig. 3a shows the comparison of average VoI and information invalidation rate among all algorithms. The proposed BP-DOD algorithm achieves an average VoI of 0.882, while the random selection algorithm, the nearest node first algorithm, the DRL-based algorithm and the MAPPO algorithm obtain average VoI of 0.667, 0.313, 0.105 and 0.854, respectively—representing an increase of up to 0.777 in average VoI. The gap in information invalidation rate is more significant: the invalidation rates of the four benchmarks are 26.1%, 66.1%, 88.9% and 5.1%, respectively, whereas that of the proposed BP-DOD algorithm is only 2.3%, representing a maximum reduction of 86.6 percentage points. In terms of stability, the standard deviation of average VoI of the proposed BP-DOD algorithm is 0.0066, and the standard deviation of information invalidation rate is 0.0069, slightly higher than 0.0026 and 0.0028 of the DRL-based algorithm. However, this does not mean that the DRL-based algorithm is more stable; instead, it indicates that the scenario has reached the performance limit of the DRL-based algorithm, whose performance is at an extremely low level with very little fluctuation.

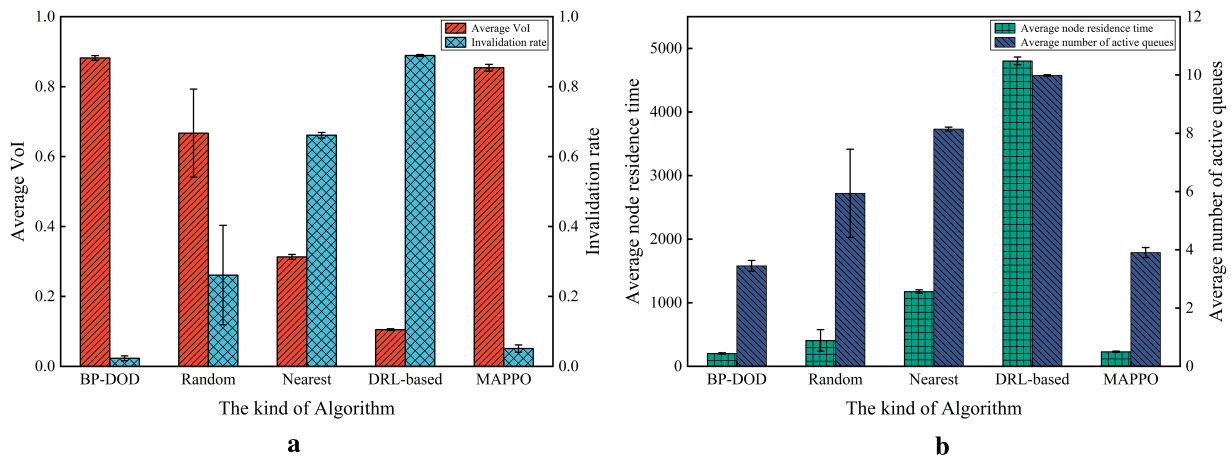


**Figure 3:** Performance comparison with benchmark algorithms. (a) Average VoI and information invalidation rate; (b) Average residence time and number of active queues.

Fig. 3b presents the comparison of average node residence time and average number of active queues. The average residence times of the random selection algorithm, the nearest node first algorithm, the DRL-based algorithm and the MAPPO algorithm are 408.23, 1176.68, 4805.42 and 230 time slots, respectively, while that of the proposed BP-DOD algorithm is only 202.54 time slots. In terms of average number of active queues, the four benchmarks have 5.938, 8.142, 9.983 and 3.905 queues on average, respectively, all higher than 3.449 of the proposed BP-DOD algorithm. For stability, the standard deviation of average node

residence time of the proposed BP-DOD algorithm is 10.8105, the lowest among all algorithms; the standard deviation of average number of active queues is 0.1851, higher than 0.0672 of the nearest node first algorithm and 0.0242 of the DRL-based algorithm. This is because the average number of active queues of the proposed BP-DOD algorithm is significantly lower than that of other algorithms, making it more susceptible to node load fluctuations.

## 5.2 Ablation Study

To verify the contributions of the LSTM, conflict cost, Double DQN and Dueling DQN, four algorithm variants are compared: the full BP-DOD algorithm, the standard DQN (without Dueling DQN and Double DQN), the BP-DOD algorithm without LSTM (with the prediction layer removed), and the BP-DOD algorithm without conflict cost (with the feature construction layer removed).

Fig. 4a and Fig. 4b show the convergence performance of average VoI and information invalidation rate for each variant, respectively. It can be observed that as the number of training episodes increases, the full BP-DOD algorithm converges the fastest with the highest stability and the best performance upper bound, followed by the standard DQN with slower convergence and slightly worse performance. The BP-DOD without LSTM achieves slightly lower performance than the standard DQN. For the variant without conflict cost, since it cannot estimate the future competition for node computing resources during offloading, it is difficult to effectively plan offloading decisions for UAVs. Therefore, it has the slowest convergence, the worst performance and the largest fluctuation.

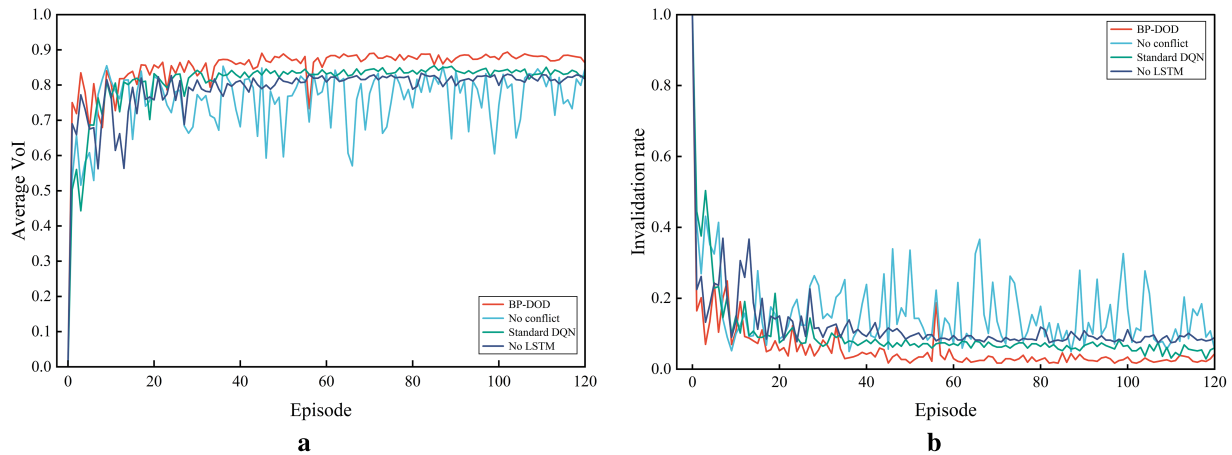


**Figure 4:** Results of ablation study. (a) Convergence of average VoI; (b) Convergence of information invalidation rate.

In addition, we quantify the prediction accuracy, and the results are summarized in Table 4. The NRMSE is obtained via min-max normalization, i.e., RMSE is divided by the value range (difference between the maximum and minimum values) of the corresponding data type in the experiment.

**Table 4:** Prediction accuracy.

|                                   | MAE    | RMSE   | NRMSE (%) |
|-----------------------------------|--------|--------|-----------|
| Offloading intervals (time slots) | 15.528 | 21.621 | 1.971     |
| Data size (Mbit)                  | 2.456  | 3.022  | 12.591    |

### 5.3 Effect of Reward Shaping

To further analyze the impact of the reward function on the performance of the proposed BP-DOD algorithm, the current reward function is compared with the pure VoI reward function. All results are statistical averages of 30 independent experiments under the same scenario.

Fig. 5a,b shows the convergence performance of average VoI and information invalidation rate under the two reward forms. It can be seen that with reward shaping, the proposed BP-DOD algorithm converges faster and more stably. Fig. 5c further presents the detailed performance after reward shaping. After reward shaping, the average VoI increases from 0.852 to 0.882, and its standard deviation decreases from 0.0111 to 0.0066; the information invalidation rate decreases from 5.2% to 2.3%, and its standard deviation decreases from 0.0114 to 0.0069; the average node residence time decreases from 261.74 to 202.54, and its standard deviation decreases from 16.6267 to 10.8105; the average number of active queues decreases from 4.464 to 3.449, and its standard deviation decreases from 0.2844 to 0.1851.

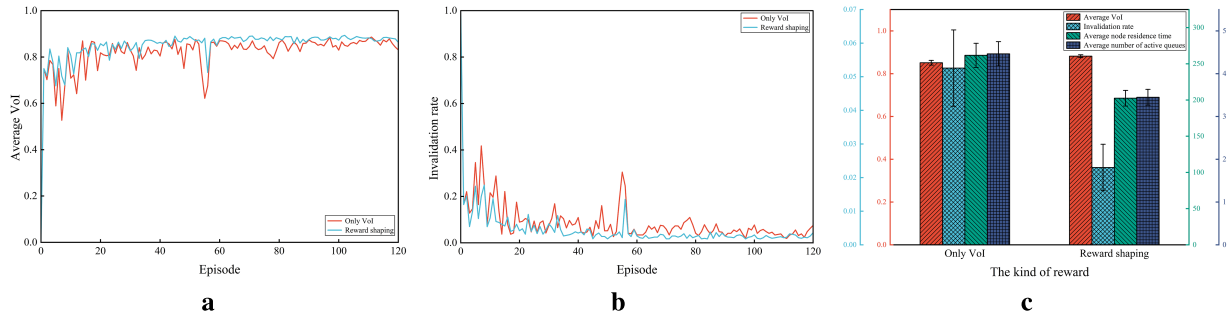


**Figure 5:** Performance comparison with and without reward shaping. (a) Convergence of average VoI; (b) Convergence of information invalidation rate; (c) Detailed performance comparison.

The above results indicate that although using VoI directly as reward is most consistent with the optimization objective in form, for information with slow or even no attenuation, although they are insensitive to delay, they still continuously occupy node queues and processing resources. If not processed in time, they will affect the VoI of other information. Theoretically, this effect can be learned through long-term rewards, but in practical scenarios, such reward signals are difficult to be effectively captured by nodes. Reward shaping enables nodes to more effectively perceive the long-term impact of resource competition by introducing auxiliary reward signals, thereby improving the performance upper bound and stability of the algorithm.

### 5.4 Parameter Sensitivity Analysis

We further analyze the impact of different scaling coefficients, learning rates and batch sizes on the convergence performance of average VoI and information invalidation rate of the proposed BP-DOD algorithm, while keeping other simulation settings unchanged.

Fig. 6a and Fig. 6b show the impact of different scaling coefficients on the convergence performance of average VoI and information invalidation rate, respectively. It can be observed that the scaling coefficient mainly affects the early training stage. When the scaling coefficient is small, the convergence performance is unstable and slow. When the scaling coefficient increases from 200 to 250, the convergence performance does not improve significantly.

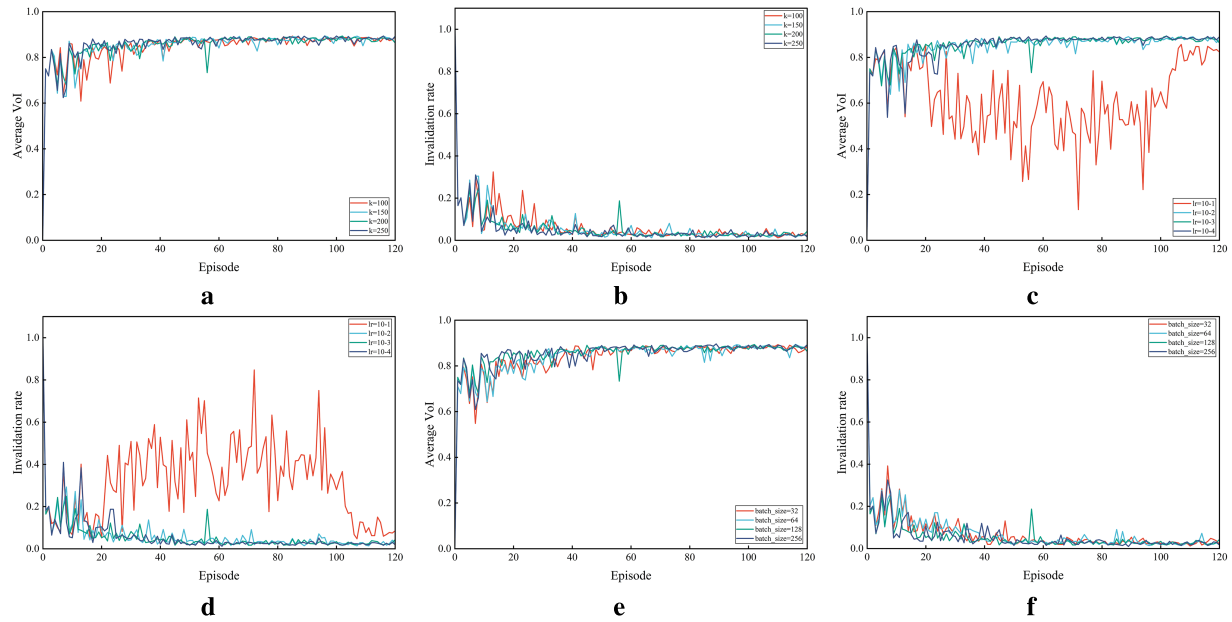


**Figure 6:** Parameter sensitivity analysis. (a) Scaling coefficient vs. average VoI; (b) Scaling coefficient vs. information invalidation rate; (c) Learning rate vs. average VoI; (d) Learning rate vs. information invalidation rate; (e) Batch size vs. average VoI; (f) Batch size vs. information invalidation rate.

Fig. 6c,d shows the impact of different learning rates. When the learning rate increases from 0.0001 to 0.001, the convergence speed in the early training stage is improved, while there is little difference in the later stage. When the learning rate continues to increase, the early convergence speed decreases. Especially when the learning rate equals 0.1, obvious oscillation occurs during training.

Fig. 6e,f shows the impact of different batch sizes. The batch size also mainly affects the early training stage. In the early stage, as the batch size increases from 32 to 128, the convergence speed is improved. When the batch size further increases to 256, the convergence performance does not improve significantly.

### 5.5 Information Exchange Frequency Experiment

Fig. 7a,b shows the impact of different state information exchange frequencies between nodes on the convergence performance of average VoI and information invalidation rate. In the default scenario, the original BP-DOD algorithm assumes that nodes can exchange state information every time slot, and the minimum bits required for sharing the three types of state information are 5, 27 and 32 bits, respectively. To evaluate the sensitivity of the algorithm to information exchange frequency, we compare the training performance when nodes exchange state information every 100, 500, 800 and 1000 time slots.

The results show that the impact of information exchange frequency is mainly concentrated in the early training stage. The scheme with information exchange every 100 time slots achieves almost the same performance as the per-slot exchange scheme. As the information exchange frequency further decreases, the convergence performance of the algorithm degrades gradually.

Frequent state information exchange helps the algorithm converge quickly. However, the algorithm training is based on random sampling from the experience replay buffer rather than relying solely on the latest experience, so it has a certain tolerance to the reduction of information exchange frequency. In addition, with the accumulation of interactions between nodes, nodes can fully understand the offloading behaviors of UAVs, which compensates for the adverse effect of reduced exchange frequency. Therefore, the

communication overhead can be reduced by appropriately lowering the information exchange frequency without significantly deteriorating the algorithm performance.

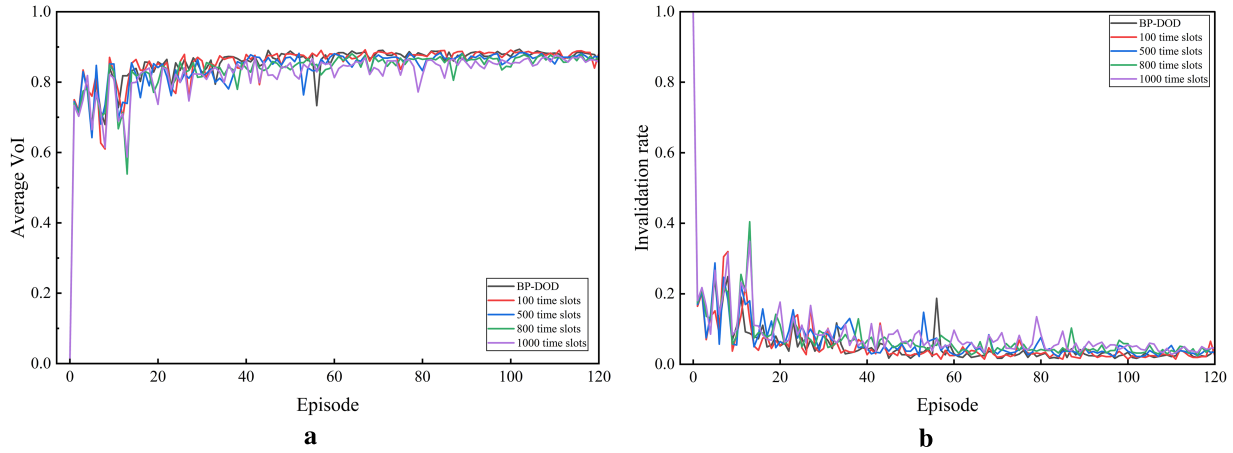


**Figure 7:** Performance under different information exchange frequencies. (a) Average VoI; (b) Information invalidation rate.

### 5.6 Impact of UAV and Node Scales

Fig. 8 shows the impact of different UAV and node scales on algorithm performance, where solid lines represent average values of 30 independent experiments and shaded areas represent standard deviations.

In Fig. 8a,b, the number of edge nodes is fixed at 4 and the number of UAVs is adjusted to investigate the impact of UAV scale. As the number of UAVs increases, the average VoI of the random selection algorithm drops from 0.807 to 0.576, and its information invalidation rate rises from 11.367% to 36.038%, with consistently large standard deviation. The nearest node first algorithm always performs the worst: its average VoI increases from 0.119 to 0.319, and its information invalidation rate decreases from 87.367% to 65.43%. The DRL-based algorithm achieves an average VoI of 0.769 with 40 UAVs, but drops sharply to 0.437 at 45 UAVs and then stabilizes; its information invalidation rate surges from 14.897% to 52.564% and then levels off. The MAPPO algorithm sees its average VoI decrease from 0.85 to 0.47 and its information invalidation rate increase from 6.985% to 46.872%. In contrast, the proposed BP-DOD algorithm always maintains the highest average VoI: it reaches 0.87 with 40 UAVs and remains basically stable, staying at 0.838 with 55 UAVs. Its information invalidation rate is always the lowest, 5.128% at 40 UAVs and basically unchanged afterwards, remaining at 6.962% with 55 UAVs. The results verify that the adaptability of the proposed BP-DOD algorithm under different UAV scales is significantly better than other algorithms.

In Fig. 8c,d, the number of UAVs is fixed at 40 and the number of edge nodes is adjusted to investigate the impact of node scale. The nearest node first algorithm still achieves the worst overall performance. The average VoI of the random selection algorithm rises from 0.018 to 0.837, and its information invalidation rate drops from 97.981% to 8.214%. The DRL-based algorithm sees its average VoI increase from 0.07 to 0.868 and its information invalidation rate decrease from 92.304% to 5.323%. The MAPPO algorithm achieves an average VoI from 0.016 to 0.873 and an information invalidation rate from 98.238% to 5.063%. The proposed BP-DOD algorithm yields an average VoI from 0.075 to 0.908 and an information invalidation rate from 92.256% to 0.686%, consistently outperforming other algorithms. The above results indicate that when the number of nodes is small and computing resources are tight, the performance of all algorithms is at a low level with little difference. As the number of nodes increases, the performance gap between algorithms widens rapidly. When the number of nodes further increases, sufficient computing resources compensate for the

performance limitations of each algorithm, and the performance of all algorithms reaches a satisfactory level with gradually narrowing gaps.

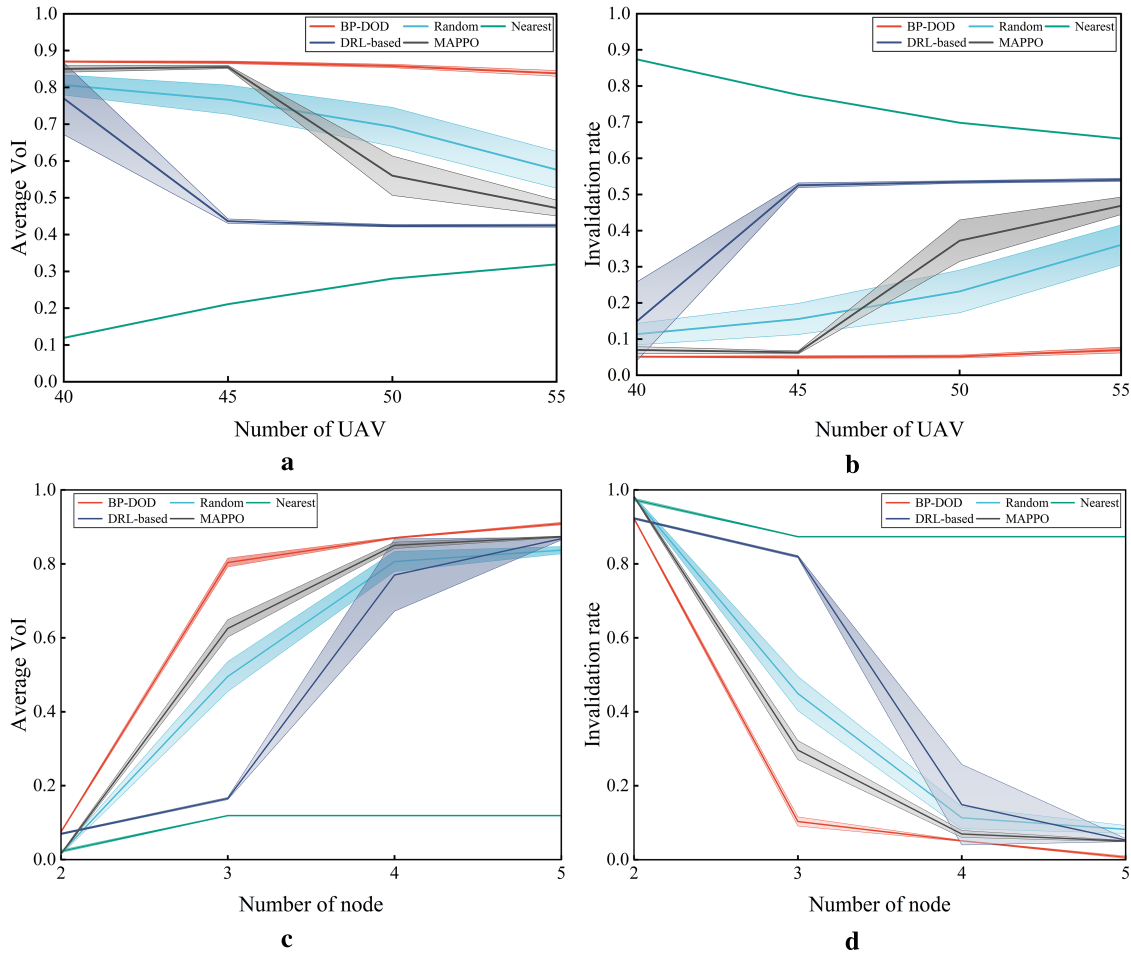


**Figure 8:** Performance under different UAV and node scales. (a) Average VoI vs. number of UAVs; (b) Information invalidation rate vs. number of UAVs; (c) Average VoI vs. number of nodes; (d) Information invalidation rate vs. number of nodes.

### 5.7 Computational and Storage Overhead

We evaluate the practical computational and storage overhead of the BP-DOD algorithm under the default scenario. In each episode, the total training time includes both algorithm computation and discrete-time environment simulation, while the decision latency only covers the computation from processing the observed state to generating the offloading action, excluding environment advancement and task scheduling.

As shown in Table 5, completing 120 training episodes takes approximately 62.32 min, i.e., an average of 31.16 s per episode. Each training episode contains an average of 406.28 offloading decisions and 122.13 network-update rounds. The mean decision latency and the 95th-percentile decision latency are 15.30 and 18.68 ms, respectively, both significantly shorter than the duration of a single time slot. This indicates that the BP-DOD algorithm can generate an offloading decision within one time slot on the simulation platform.

In terms of storage, each node contains 120,201 trainable parameters plus target network parameters, which occupy only about 0.46 MiB when stored in FP32 format. Under the configured capacity, the replay buffer of each node occupies approximately 21.90 MiB. During training, the full simulation process reaches

peak CPU and GPU memory usage of about 1193.80 and 17.89 MiB, respectively. Note that the CPU memory here includes the simulation environment, replay buffer, Python runtime and TensorFlow runtime, so this value is a conservative upper bound rather than the storage requirement of the BP-DOD algorithm itself.

**Table 5:** Computational and storage overhead of BP-DOD.

| Metric                                | Value             |
|---------------------------------------|-------------------|
| Total training time, 120 episodes     | 62.32 min         |
| Average training time per episode     | 31.16 s           |
| Mean/95th percentile decision latency | 15.30/18.68 ms    |
| Model storage per node                | 0.46 MiB          |
| Replay-buffer storage per node        | 21.90 MiB         |
| Peak CPU/GPU memory during training   | 1193.80/17.89 MiB |

## 6 Conclusion

This paper investigates the offloading node selection problem after multi-UAV information collection, considering the impact of users' personalized demands on information timeliness. In this problem, UAVs can neither know each other's offloading behaviors nor perceive the real-time status of nodes. On this basis, the concept of Value of Information is introduced, the offloading node selection problem is modeled as a semi-Markov decision process, and a Behavior Prediction-based Distributed Offloading Decision (BP-DOD) algorithm is proposed. The algorithm is designed on the DQN architecture, with LSTM introduced to learn and predict UAV offloading behaviors, and conflict cost adopted to estimate future resource competition of nodes. Dueling DQN, Double DQN and reward shaping techniques are further integrated to improve algorithm performance. Simulation results show that the proposed BP-DOD algorithm can reasonably allocate offloading nodes for UAVs according to their behaviors, effectively improve the average VoI of information and reduce the information invalidation rate, and exhibits favorable adaptability across different scales of UAVs and edge nodes. Future work will consider more sophisticated UAV data collection behaviors, and conduct further research on flight obstacle avoidance, complex electromagnetic environments, heterogeneous demands of multiple users, and node energy consumption, so as to enhance the adaptability and scalability of the proposed algorithm in more complex scenarios.

**Acknowledgement:** The authors would like to express sincere gratitude to the supervisor Wendong Zhao for valuable guidance and suggestions throughout the whole research and manuscript writing process.

**Funding Statement:** This work was supported by the National Natural Science Foundation of China under Grant No. 62171465.

**Author Contributions:** Zengqi Ma: Conceptualization, manuscript writing, algorithm design and simulation debugging, figure drawing, data analysis; Wendong Zhao: Supervision, methodology guidance, revision of the manuscript. All authors reviewed and approved the final version of the paper.

**Availability of Data and Materials:** The simulation code and raw experimental data supporting the findings of this study are available from the corresponding author upon reasonable request.

**Ethics Approval:** Not applicable. This paper only involves simulation experiments without human subjects or animal experiments.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Feng X, Wang D, Wang T, Liu X, Cheng H, Li J. Optimal topology control and energy-efficient scheduling strategy for UAV-assisted data collection in wireless sensor networks. *IEEE Internet Things J.* 2026;13(5):8955–67. doi:10.1109/JIOT.2025.3642310.
2. Zhang Z, Jiang J, Ling KV, Wang X, Zhang W-A. Cooperative task allocation and path planning for multi-UAVs in low-altitude urban intelligent transportation systems. *IEEE Trans Intell Transp Syst.* 2026;27(4):4112–24. doi:10.1109/TITS.2026.3667967.
3. Sangaiah AK, Anandakrishnan J, Meenakshisundaram V, Rahman MAA, Arumugam P, Das M. Edge-IoT-UAV adaptation toward precision agriculture using 3D-LiDAR point clouds. *IEEE Internet Things Magazine.* 2025;8(1):19–25. doi:10.1109/IOTM.001.2400027.
4. Mondal MS, Ramasamy S, Humann JD, Dotterweich JM, Bhounsule PA. Deep reinforcement learning enabled persistent surveillance with energy-aware UAV-UGV systems for disaster management applications. *IEEE Tran Intell Veh.* 2026;11(8):914–29. doi:10.1109/TIV.2026.3696943.
5. Porambage P, Okwuibe J, Liyanage M, Ylianttila M, Taleb T. Survey on multi-access edge computing for internet of things realization. *IEEE Commun Surv Tutor.* 2018;20(4):2961–91. doi:10.1109/COMST.2018.2849509.
6. Goudarzi S, Soleymani SA, Anisi MH, Jindal A, Xiao P. Optimizing UAV-assisted vehicular edge computing with age of information: an SAC-based solution. *IEEE Inter Things J.* 2025;12(5):4555–69. doi:10.1109/JIOT.2025.3529836.
7. Sun Y, Wu Z, Meng K, Zheng Y. Vehicular task offloading and job scheduling method based on cloud-edge computing. *IEEE Trans Intell Transp Syst.* 2023;24(12):14651–62. doi:10.1109/TITS.2023.3300437.
8. Ma X, Lin C, Xiang X, Chen C. Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing. In: *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'15)*; 2015 Nov 2–6; Cancun, Mexico. p. 271–8. doi:10.1145/2811587.2811598.
9. Cao X, Xu J, Zhang R. Mobile edge computing for cellular-connected UAV: computation offloading and trajectory optimization. In: *Proceedings of the 2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*; 2018 Jun 25–28; Kalamata, Greece. p. 1–5. doi:10.1109/SPAWC.2018.8445936.
10. Dai M, Su Z, Li J, Zhou J. An energy-efficient edge offloading scheme for UAV-assisted Internet of Things. In: *Proceedings of the 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*; 2020 Nov 29–Dec 1; Singapore. p. 1293–7. doi:10.1109/ICDCS47774.2020.00167.
11. Zhang B, Xiao F, Wu L. Offline reinforcement learning for asynchronous task offloading in mobile edge computing. *IEEE Trans Netw Serv Manag.* 2024;21(1):939–52. doi:10.1109/TNSM.2023.3316626.
12. Sun C, Li X, Wang C, He Q, Wang X, Leung VCM. Hierarchical deep reinforcement learning for joint service caching and computation offloading in mobile edge-cloud computing. *IEEE Trans Serv Comput.* 2024;17(4):1548–64. doi:10.1109/TSC.2024.3355937.
13. Vidya S, Gopalakrishnan R. Dynamic task offloading in edge computing for computer access point selection based on adaptive deep reinforcement learning with meta-heuristic optimization. *Appl Soft Comput.* 2025;176(1):113105. doi:10.1016/j.asoc.2025.113105.
14. Tang M, Wong VWS. Deep reinforcement learning for task offloading in mobile edge computing systems. *IEEE Trans Mob Comput.* 2022;21(6):1985–97. doi:10.1109/TMC.2020.3036871.
15. Rahmaty I, Shah-Mansouri H, Movaghar A. QECO: a QoE-oriented computation offloading algorithm based on deep reinforcement learning for mobile edge computing. *IEEE Trans Netw Sci Eng.* 2025;12(4):3118–30. doi:10.1109/TNSE.2025.3556809.
16. Wang F, Ma X, Meng P, Zhao W, Zhai W. Task offloading strategy of UAV edge computing based on deep reinforcement learning. *J Zhengzhou Univ Eng Sci.* 2025;46(4):16–23. doi:10.13705/j.issn.1671-6833.2025.01.018.
17. Deng F, Shan Y, Xie Z, Zhang P, He Y. Power inspection and unloading strategy of UAV based on game theory and reinforcement learning. *Power Syst Technol.* 2021;45(9):3649–57. doi:10.13335/j.1000-3673.pst.2020.1753.
18. Sacco A, Esposito F, Marchetto G, Montuschi P. Sustainable task offloading in UAV networks via multi-agent reinforcement learning. *IEEE Trans Veh Technol.* 2021;70(5):5003–15. doi:10.1109/TVT.2021.3074304.

19. Li M, He S, Li H. Minimizing mission completion time of UAVs by jointly optimizing the flight and data collection trajectory in UAV-enabled WSNs. *IEEE Internet Things J.* 2022;9(15):13498–510. doi:10.1109/JIOT.2022.3142764.
20. Fu X, Song X, Pace P, Aloï G, Fortino G. Reinforcement learning-based low-delay data collection in UAV-assisted IoT for secondary geological hazard monitoring. *IEEE Trans Veh Technol.* 2025;74(8):12935–50. doi:10.1109/TVT.2025.3554815.
21. Yuan X, Hu Y, Zhang J, Schmeink A. Joint user scheduling and UAV trajectory design on completion time minimization for UAV-aided data collection. *IEEE Trans Wirel Commun.* 2023;22(6):3884–98. doi:10.1109/TWC.2022.3222067.
22. Li M, Liu X, Wang H. Completion time minimization considering GNs' energy for UAV-assisted data collection. *IEEE Wirel Commun Lett.* 2023;12(12):2128–32. doi:10.1109/LWC.2023.3309625.
23. Liu W, Li D, Liang T, Zhang T, Lin Z, Al-Dhahir N. Joint trajectory and scheduling optimization for age of synchronization minimization in UAV-assisted networks with random updates. *IEEE Trans Commun.* 2023;71(11):6633–46. doi:10.1109/TCOMM.2023.3297198.
24. Liu J, Tong P, Wang X, Bai B, Dai H. UAV-aided data collection for information freshness in wireless sensor networks. *IEEE Trans Wirel Commun.* 2021;20(4):2368–82. doi:10.1109/TWC.2020.3041750.
25. Kaul S, Yates R, Gruteser M. Real-time status: how often should one update? In: *Proceedings of the 2012 IEEE INFOCOM*; 2012 Mar 25–30; Orlando, FL, USA. p. 2731–5. doi:10.1109/INFCOM.2012.6195689.
26. Liu Z, Meng X, Liu Y, Yang Y, Wang Y. AUV-aided hybrid data collection scheme based on value of information for internet of underwater things. *IEEE Internet Things J.* 2022;9(9):6944–55. doi:10.1109/JIOT.2021.3115800.
27. Alawad F, Kraemer FA. Value of information in wireless sensor network applications and the IoT: a review. *IEEE Sens J.* 2022;22(10):9228–45. doi:10.1109/JSEN.2022.3165946.
28. Abbas N, Eid H, Farhat MJ. VoI-aware energy-efficient UAV-assisted data collection using genetic algorithms. In: *Proceedings of the 2025 IEEE 22nd Consumer Communications & Networking Conference (CCNC)*; 2025 Jan 10–13; Las Vegas, NV, USA. p. 1–6. doi:10.1109/CCNC54725.2025.10976207.
29. Jing Z, Dang W, Li J, Ma C, Yang Q, Wu Y. VoI-driven multi-UAV collaboration for data collection with partially outdated observation. *IEEE Wirel Commun Lett.* 2026;15:1717–21. doi:10.1109/LWC.2026.3660405.
30. Sutton RS, Precup D, Singh S. Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artif Intell.* 1999;112(1):181–211. doi:10.1016/S0004-3702(99)00052-1.
31. Minoshima K, Arai S. Autonomous reward shaping via self-generated trajectories for sparse-reward reinforcement learning. *IEEE Access.* 2026;14:33890–907. doi:10.1109/ACCESS.2026.3662779.
32. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. *Nature.* 2015;518(7540):529–33. doi:10.1038/nature14236.
33. Wang Z, Schaul T, Hessel M, Hasselt HV, Lanctot M, de Freitas N. Dueling network architectures for deep reinforcement learning. In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)*; 2016 Jun 19–24; New York, NY, USA. p. 1995–2003.
34. Hasselt HV, Guez A, Silver D. Deep reinforcement learning with double Q-learning. In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*; 2016 Feb 12–17; Phoenix, AZ, USA. p. 2094–100.
35. Zhao M, Xiao Y, Yao J, Wang T, Lee J, Quek TQS. Up-downlink AoI-driven multi-source data collection in UAV-assisted wireless sensor networks. *IEEE Trans Wirel Commun.* 2025;24(2):1178–92. doi:10.1109/TWC.2024.3506121.
36. Nugroho AK, Kim T. Traffic-aware intelligent association and task offloading for multi-access edge computing. *Electronics.* 2024;13(16):3130. doi:10.3390/electronics13163130.