AoI and Energy-Driven Dynamic Cache Updates for Wireless Edge Networks

Yuan Yuan, Bin Yang, Wei Su, Haoru Li, Chang Wang, Qi Liu, Tarik Taleb

***Abstract*—Wireless edge networks can provide edge services to support various time-critical Internet of Things (IoT) applications, like autonomous vehicles, where cache content updates are significant to maintaining information freshness quantified as information of age (AoI). However, frequent content updates result in high energy consumption at the edge nodes. This paper investigates the cache content updates in wireless edge networks, aiming to ensure information freshness and low energy consumption. To this end, we propose a rainbow deep reinforcement learning-based cache content update scheme (RB-DRN). In the RB-DRN scheme, we first establish a Markov Decision Process (MDP) to characterize the process of cache update. By fully taking advantage of R-Learning empowered Rainbow DQN, we then make optimal strategy to obtain the minimum long-term average overhead associated with energy consumption and information freshness. Extensive simulation results are presented to validate our proposed RB-DRN scheme and also to illustrate that our RB-DRN scheme outperforms the benchmark scheme in terms of information freshness and energy consumption.**

***Index Terms*—Wireless Edge Networks, Internet of Things, Age of Information, Cache Update, Rainbow DQN.**

I. INTRODUCTION

WIRELESS edge networks (WENTs) are a promising class of edge network architecture where multi-access edge computing (MEC) techniques are utilized through placing sensing nodes with the ability to compute and store on the edge side of the network close to end users, thereby reducing access to central cloud server resources [1]–[7]. Such networks have been identified as a new paradigm for supporting various compute-intensive and delay-sensitive Internet of Things (IoT) applications, such as the Internet of Vehicles (IoV), augmented reality (AR), virtual reality (VR), and extended reality (XR) [8]–[12]. In particular, the sensing nodes need to perform frequent cache updates to maintain information freshness in IoV applications [13]–[17]. However, this will incur high energy consumption for the energy limited-sensors equipped at the sensing nodes. Thus, it is critical to explore dynamic cache update scheme while ensuring information freshness and low energy consumption in WENTs [18]–[20].

Existing works on dynamic cache updates mainly considers resource scheduling [21]–[27] and status updates [28]–[34] in WENTs (See Section III for details). The above works on dynamic cache updates mainly study the issues of age of information (AoI), latency and energy via resource scheduling and status updating using mathematical optimization algorithms and learning algorithms, such as these works summarized in Table I. Note that the cache tasks are assumed to have the same priority in these works. However, the cache tasks usually employ different priorities in real scenarios [35]–[38]. For example, fire alarms have high priority because such alarm tasks require a timely response, but temperature and humidity sensors have lower priority because they are not so sensitive to delay. This means that system will assign more resources (e.g., transmit power) to the edge nodes with high priority tasks. Moreover, these works mainly adopt heuristic algorithms and traditional learning algorithms. The former cannot perceive the environmental information, while the latter suffers from either dimensional catastrophe (e.g., the Q-learning algorithms) or sample non-smoothness and hyper-parameter sensitivity (e.g., the deep Q-learning algorithms) [39]–[41]. In particular, it is of vital importance to maintain information freshness and low energy consumption for supporting various time-critical and energy-sensitive applications, like autonomous vehicles.

Motivated by the above observations, we explore the cache dynamic update problem in a WENT with one base station (BS), one edge server, multiple sensing nodes and users, which aims to provide users with cached content with high information freshness while guaranteeing low energy consumption. In our considered network, the AoI is of fundamental importance to qualify the information freshness for supporting time-critical assisted driving services (e.g., dynamic high-definition maps files). To keep information freshness, the service content needs to be frequently updated, which results in high energy consumption at sensing nodes with limited energy. To achieve our goal, traditional optimization methods need to acquire global network information (e.g., channel state information) in advance. Therefore, such methods lead to large signaling overhead, and could also not achieve an effective solution in highly dynamic environments (e.g., vehicles' mobility). Remarkably, we propose a Rainbow Deep Reinforcement

This work was supported in part by the National Key Research and Development Project of China under Grant 2022YFB2901603; in part by the Ministry of Education Innovation Group Joint Fund under Grant 8091B042222; in part by the National Natural Science Foundation of China under Grant 62372076; in part by the Natural Science Foundation of Anhui Province under Grant KJ2021ZD0128, KJ2021B01; in part by the Anhui Talent Project under Grant DTR2023051; in part by ICTFICIAL Oy, Finland; and in part by the European Union's HE Research and Innovation Program HORIZON-JUSNS-2023 through the 6G-Path Project under Grant 101139172. (Corresponding authors: Wei Su; Bin Yang.)

Y. Yuan, W. Su, H. Li, and C. Wang are with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China. E-mail: yuan.yuan@bjtu.edu.cn, wsu@bjtu.edu.cn, 22120129@bjtu.edu.cn, 21120074@bjtu.edu.cn.

B. Yang is with the School of Computer and Information Engineering, Chuzhou University, Chuzhou, Anhui, China. E-mail: yangbinchi@gmail.com.

Q. Liu is with the Smart City Research Institute of China Unicom, Beijing, China. E-mail: liuqi49@chinaunicom.cn.

Tarik Taleb is with the Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum, Bochum 44801, Germany. E-mail: tarik.taleb@rub.de.

TABLE I
COMPREHENSIVE COMPARATIVE ANALYSIS OF DYNAMIC CACHE UPDATES.

Comparison metrics	[21]	[24]	[27]	[28]	[31]	[33]	[34]	The proposed scheme
Optimization Method	(M)	(M)	(L)	(M)	(L)	(L)	(L)	(L)
Status Updating	–	–	–	✓	✓	✓	✓	✓
Resource Scheduling	✓	✓	✓	–	–	–	✓	✓
Objective of Latency	✓	–	–	–	–	–	✓	✓
Objective of AoI	✓	✓	✓	✓	✓	✓	✓	✓
Objective of Energy	–	✓	✓	✓	✓	✓	✓	✓
Objective of Priority and Antennas Power	–	–	–	–	–	–	–	✓

[1] (M): Optimization scheme based on mathematical optimization algorithm. (L): Optimization scheme based on learning algorithm.
[2] Symbol ✓ indicates a high relevance. Symbol – indicates a low relevance.

Learning-based cache content updating scheme (RB-DRN) to achieve the minimum long-term average overhead associated with energy consumption and information freshness, in which RB-DRN can utilize historical information to make predictions without instantaneous network information and efficiently reduce signaling overhead.

It is worth noting that we consider the edge networks are made of multiple BSs, multiple edge servers and multiple mobile users, where each server equipped at a BS serve a group of mobile users independently. Without loss of generality, we only focus on the scenario with one BS, one edge server and multiple users in this paper, which is widely used in previous works [21], [23], [51], [52]. This is because the high-definition maps (HDM) files requested by mobile users does not require too much storage space. The edge server, which provide powerful computation and storage capabilities, can meet mobile users' needs. Our concerned network serves as a human-vehicle collaborative assisted driving system. The sensing node can actively update the cache content once if the AoI of the cache content exceeds a given threshold or the cache content cannot meet the need of user request. When a significant change of its sensing environment (e.g., traffic accident) occurs, a human driver can also collaboratively control the vehicle. We summarize the main contribution of this paper as follows.

- We define an average system overhead function consisting of AoI demand overhead, system energy consumption overhead, antenna power allocation overhead, and task priority matching overhead. We formulate AoI and energy-driven dynamic cache update as an optimization problem aiming to minimize the average system overhead with the constraint of the maximum tolerance value of AoI.
- To solve this optimization problem, we then model the joint resource scheduling and status updating of dynamic cache update as a Markov Decision Process (MDP), which characterizes the state space including AoI value, power of antenna and task priority. The information transmission failure probability and the constraints of link rate and the number of updates are also carefully considered in this MDP.
- We further propose a rainbow deep reinforcement learning based cache content update scheme (RB-DRN) to solve this optimization problem. Notably, by fully taking advantage of R-Learning empowered Rainbow DQN, we can obtain the optimal decision of MDP to maximize the average system utility.
- Finally, we conduct extensive simulation studies to validate our proposed RB-DRN scheme and to demonstrate the impact of parameters on average system overhead. We also conduct the comparison study between our proposed scheme and some benchmark methods.

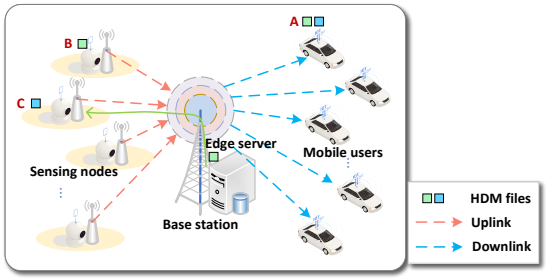


Fig. 1. Motivating scenario.

The rest of the paper is organized as follows. Section II indicates the motivating scenario. Section III describes related works. Section IV introduces the system model in the WENT scenario. Section V formulates an optimization problem with the aim to minimize the system overhead. In Section VI, we propose the dynamic cache update scheme to solve this optimization problem. Simulation results are provided in Section VII. Finally, Section VIII concludes this paper.

II. MOTIVATING SCENARIO

As shown in Fig. 1, we provide a motivation example including input data and expected outputs in which there is a WENT consisting of a base station (BS), an edge server, 10 sensing nodes, and 100 mobile users (e.g., vehicles) within the coverage area of BS. The vehicles in the network need to request the HDM files generated by the sensing nodes for assisted driving. It is noted that these HDM files need to satisfy the need for information freshness, but frequent refreshing of sensing nodes will incur a high energy overhead. For this reason, the information freshness for the vehicle's HDM file should be satisfied while keeping the sensing node refresh

frequency low. This means that the WENT needs to ensure information freshness and low energy consumption, which are measured by a utility function. A large utility function value indicates high information freshness and low energy consumption. Thus, this paper aims to maximize the utility function. We will provide the input and output data for the example.

For the input data, we give the following settings. Within the coverage area of BS, there are 10 sensing nodes for sensing road traffic information and generating dynamic HDM files. In this case, the size of each HDM file is 1 MB. All HDM files will be updated to the edge server through uplink (orange line in Fig. 1) caching to supply vehicle request services. Since there will be competition among multiple sensing nodes transmitting data at the same time, we limit the number of transmitted contents to no more than 5. The energy consumption required for each update of the sensing nodes is 10 mW. Vehicles request HDM files through the downlink (blue line in Fig. 1). The channel bandwidth is 5 MHz, and the transmit power of the BS antenna is 0.5 W. The channel capacity is 3 Mbps. There is a timeliness in HDM, and the content with fresher information will be more helpful to the vehicle. However, frequent updates will cause very high energy consumption overhead. Therefore, it is necessary to manage the updating process of HDM. At the same time, differentiated resource allocation can be made according to different priorities of vehicles. In Fig. 1, high-priority vehicle *A* requests HDM files from sensing node *B* and sensing node *C* at 3:00 a.m. If the edge server has cached the requested content originated from the sensing nodes, the content (e.g., the HDM file generated by sensing node *B*) is directly transmitted to the vehicle *A* via downlink. Otherwise, the edge server requests to update the cached content (green line in Fig. 1) and then transmit it to the vehicle after the sensing node (e.g., the sensing node *C*) has uploaded the content. As for the expected outputs, the system outputs the maximum utility function value (i.e., -25 in this example) by jointly optimizing cache updates and resource allocation.

III. RELATED WORK

In WENTs, MEC is an efficient resource management method to provide cache update services. Available dynamic cache update studies mainly consider resource scheduling [21]-[27] and status updating [28]-[34].

For the resource scheduling, the authors in [21] propose a freshness-aware refreshing and transmission resource allocation scheme to balance the service delay and information freshness measured by age of information. The authors in [22] prove that the AoI and delay can be minimized simultaneously in a short-packet transmission scenario, where these two performance metrics are affected by update rate and block length in a complex manner. Two schemes are proposed in [23] for cache update and content delivery at the roadside units, which can significantly reduce service latency while ensuring content freshness in heavily loaded information-centric vehicular networks. In [24], the authors propose age-minimal transmission policies for a two-hop energy harvesting communication network, and indicate that the data buffer of the relay should not store any update packets waiting for service under the optimal update policy. The work of [25] examines the age-energy tradeoff in an IoT monitoring system through adopting a practical truncated automatic repeat request scheme. Under such a scheme, when the transmission times does not reach its maximum or a new status update is not generated, the IoT device can always transmit the current status update. In [26], the authors explore the energy-AoI tradeoff for a status update system where a sensor generates and transmits status packets to a monitor over an error-prone channel. The authors in [27] propose an optimal transmission scheme to achieve the minimization of the weighted communication on AoI and total energy consumption.

As for the status updating, the authors in [28] aim to minimize the average AoI at a destination with the constraint of energy causality at an energy harvesting sensor. Here, the sensor continuously monitors a wireless communication system and transmit time-stamped status updates to the destination. In [29], the authors first model an IoT-based multi-source status update system as a multi-source M/G/1/1 bufferless preemptive queue, and then optimize the arrival rate allocation to control the maximal violation probabilities improving the overall timeliness of the multi-source system. In [30], the authors propose a context-based metric (i.e., Urgency of Information) to measure the nonlinear time-varying importance and the non-uniform context-dependence of the status information. In a real-time IoT monitoring system, the work of [31] jointly optimizes the status sampling and updating process to minimize the average AoI at the destination with the constraint of an average energy cost at each device. The work of [32] considers a scenario of cognitive radio energy harvesting communications and explores the average AoI minimization under such a scenario. In [33], the authors focus on striking a balance between the information freshness and energy consumed by sensors in a caching enabled IoT network which is formulated as a non-uniform time frame based dynamic status update optimization problem to minimize the long-term average overhead. In [34], a deep reinforcement learning-based algorithm is proposed to solve the content update and transmission resource allocation problem with the goal of minimizing the long-term average overhead.

We summarize the characteristics of some representative works mentioned above in Table I.

IV. SYSTEM MODEL

A. Network Model

As shown in Fig. 2, we consider a WENT consisting of a BS, an edge server, M sensing nodes, and multiple mobile users (e.g., vehicles) within the coverage area of BS. The edge server equipped at BS is considered as a core network entity to cache HDM files with different file sizes for mobile users, and also dynamically updates HDM files obtaining from sensing nodes to ensure information freshness. Specifically, the HDM files for assisted driving can provide road condition information, pedestrian flow information and rules for lane-changing driving. Sensing nodes embedded with roadside sensors are

responsible for sensing dynamic environmental information and generate HDM files. Due to the limited communication range of sensing nodes, we consider mobile users obtain HDM files from the edge server rather than the sensing nodes. For supporting assisted driving, the mobile users request HDM files with different priorities (i.e., task priority identifier $V_n(t)$ for user n at time frame t, which is used to quantify the priority of user n in obtaining transmit power resources.). Here, the mobile users with higher priority can obtain more transmit power resources in $Z_n(t)$, which is the number of base station's transmit power resource blocks assigned to user n at time frame t. It is used to quantify the transmit power allocated to user n by the BS in the downlink. In the WENT, there exist two types of communication links, i.e., downlink from BS to each mobile user, and uplink from each sensing node to BS. We consider that the BS is equipped with multiple orthogonal antennas adopting multiple-input multiple-output (MIMO) technique, where each antenna communicates with one mobile user using equal-sized spectrum resource with the constraint of maximum transmit power p_{max}.

It is notable that our WENT is critical to support assisting driving for connected vehicles. mobile users, i.e., vehicles, input requests for assisted driving (i.e., input data including file sizes, transmission capacity, energy consumption, etc), and then the edge server in the WENT outputs HDM files to the mobile users for helping assisted driving system in making decisions (i.e., expected outputs for assisted driving decisions). The execution process is as follows.

Mobile user n first requests an HDM file cached at the edge server with request identifier $D_n^m(t)$ (i.e., User n request identifier $D_n^m(t)$ for sensing node m at time frame t, which is used to quantify the state of user n when generating HDM file requests).

The edge server then judges whether the freshness of the HDM file (i.e., the AoI value of the HDM file from sensing node m by user n at time frame t) meets the mobile user's needs with update identifier $I_m(t)$ (i.e., update identifier $I_m(t)$ for sensing node m at time frame t, which is used to quantify the actions of the sensing node m when updating the HDM file). If yes, the edge server sends the HDM file (with different file sizes F_m for mobile users) directly to the mobile user via the downlink transmission with the downlink transmission rate $R_n(t)$ between the edge server and user n. Otherwise, the edge server requires a sensing node to update the content of the HDM file. Note that the downlink transmission latency $\tau_n^{\mathrm{tra}}(t)$ represents the time consumption for the edge server to deliver the cached HDM file with user n at time frame t, and the energy consumption E_{upd} represents the energy consumption for each sensing node m update operation. The sensing node updates the content and then sends the HDM file to the edge server via uplink transmission.

Finally, the edge server sends it to the mobile user. This execution process can maintain high information freshness of HDM files and low system energy consumption by a joint optimization of the HDM file update scheme of sensing nodes and power resource allocation with overhead function. Note that we construct an overhead function consisting of AoI demand overhead C_A, system energy consumption overhead C_E, antenna power allocation overhead C_P, and task priority matching overhead C_V.

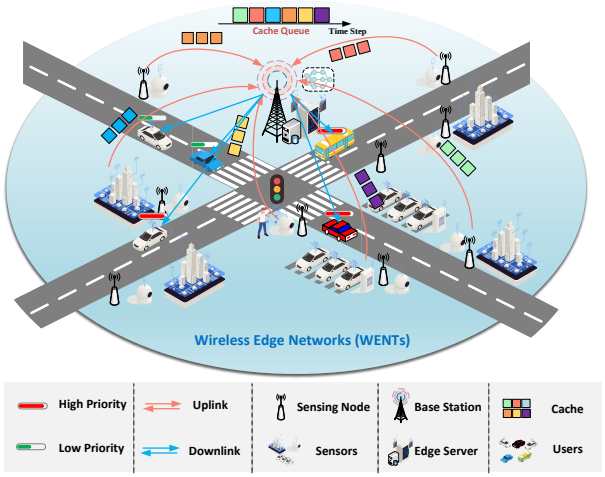


Fig. 2. Edge service network scenario.

B. Cache-Updated Model

We use $\mathcal{N}$=$\{1, 2, ..., N\}$ and $\mathcal{M}$=$\{1, 2, ..., M\}$ to represent the sets of the mobile users and sensor-equipped sensing nodes, respectively. The set of task requests from the user n within the time frame t can be represented as $D_n(t) = \{D_n^1(t), D_n^2(t), ..., D_n^m(t), ..., D_n^M(t)\}$, where $m \in \mathcal{M}$. Here, $D_n^m(t) = 1$ indicates that the user n sends a content caching request to the sensing node m in the time frame t, and $D_n^m(t) = 0$ indicates that there is no content caching request. The edge server needs to conduct an appropriate scheme for real-time AoI of the content caching after receiving the request, which can be represented by the set $I(t) = \{I_1(t), I_2(t), ..., I_m(t)..., I_M(t)\}$. Here, $I_m(t) = 0$ means that the edge server delivers the cached content directly to user n, and $I_m(t) = 1$ means that the edge server will update the cached content immediately.

To avoid uplink channel congestion, we use Y to denote the upper bound of the number of updated content, and then

$$\sum_{m=1}^{M} I_m(t) \leq Y. \tag{1}$$

C. Performance Model

1) AoI Model: The real-time information freshness of cached content depends on the decision of cache updating. We will construct AoI model to characterize the information freshness of different users with different sensing nodes in the same time frame. The information freshness of cached content is affected by the cache updating time consumption and the transmission latency (i.e., uplink transmission latency and downlink transmission latency). Meanwhile, we define the maximum allowable value of AoI as A_{max} for the cached content. When the value of AoI exceeds A_{max}, the cached content will be updated

TABLE II
DEFINITION OF MATHEMATICAL SYMBOLS

Symbol	Definition
$D_n^m(t)$	User n request identifier for sensing node m at time frame t. It is used to quantify the state of user n when generating HDM file requests.
$I_m(t)$	Update identifier for sensing node m at time frame t. It is used to quantify the actions of the sensing node m when updating the HDM file.
$V_n(t)$	Task priority identifier for user n at time frame t. It is used to quantify the priority of user n in obtaining transmit power resources.
$A_0^m(t)$	The AoI value of the HDM file from sensing node m in the edge server at time frame t.
$A_n^m(t)$	The AoI value of the HDM file from sensing node m by user n at time frame t.
F_m	The size of the HDM file from the sensing node m.
h_n	The channel gain between user n and edge server.
$R_n(t)$	The downlink transmission rate between the edge server and user n at time frame t.
$\tau_m^{\text{upd}}(t)$	The uplink transmission latency from the sensing node m to the edge server at time frame t.
$\tau_n^{\text{tra}}(t)$	The downlink transmission latency for the edge server to deliver the cached HDM file with user n at time frame t.
A_{max}	The maximum tolerance AoI value for all HDM files.
α_n	The intelligent level of users.
$\Delta_n^m(t)$	The AoI demand value of the HDM file from the sensing node m by user n at time frame t.
$\overline{\Delta}_n(t)$	The average AoI demand overhead of user n for all HDM files at time frame t.
E_{upd}	The energy consumption for each sensing node m update operation.
$Z_n(t)$	The number of base station's transmit power resource blocks assigned to user n at time frame t. It is used to quantify the transmit power allocated to user n by the base station in the downlink.
p_0	The transmit power resources contained in a unit transmit power resource block. In the case of multiple resource blocks, the base station will provide transmit power in multiples of the number of resource blocks.
p_{max}	The maximum power of an antenna.
$p_n^{\text{sup}}(t)$	The power supplied to the user n by the antenna at time frame t.
B	The bandwidth size of each antenna assigned to a single user.
C_A	The AoI demand overhead.
C_E	The energy consumption overhead.
C_P	The antenna power allocation overhead.
C_V	The task priority matching overhead.
η	The unit price of C_A. It is used to quantify the impact of HDM file's information freshness on system overhead.
μ	The unit price of C_E. It is used to quantify the impact of energy consumption of sensing nodes on the system overhead.
κ	The unit price of C_P. It is used to quantify the impact of the power resources provided by the base station on the system overhead.
ζ	The unit price of C_V. It is used to quantify the impact of the additional overhead of task priority matching on the system overhead.

We further need to distinguish between AoI values for edge server or mobile users. Fig. 3 shows how the AoI of an HDM file at the edge server varies with the time t. In Fig. 3, the vertical axis represents the AoI value of an HDM file stored at the edge server, and the horizontal axis represents the time lapse t. A_{max} represents the maximum allowable value of AoI. $\tau_m^{\text{upd}}(t)$ represents the uplink transmission latency of an HDM file from a sensing node to the edge server. The initial

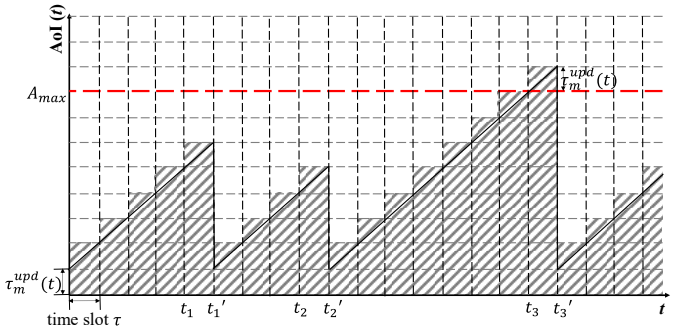


Fig. 3. The variation of AoI at the edge server.

value of AoI is the time duration from time when the sensing node generates the HDM file to the time when the file is received by the edge server, which is equal to the transmission latency $\tau_m^{\text{upd}}(t)$. We can observe from Fig. 3 that in the initial phase, the value of AoI increases linearly as t increases. At the moments t_1', t_2' and t_3', the AoI immediately returns to the initial value and then increases again. In particular, the value of AoI at moment t_3' is significantly different from other two moments. The reasons behind these phenomena can be explained as follows. Starting from an initial value, AoI is actually a function of time t with a slope of 1, and thus it increases linearly with t. When t increases up to t_1' and t_2' moments, the user requests to update the HDM file, and then the edge server experiencing $\tau_m^{\text{upd}}(t)$ transmission latency receives the updated HDM file from the sensing node. On the other hand, when the AoI achieves its maximum allowable value at t_3 moment, and then the edge server needs to request an update of the HDM file from the sensing node. Thus, the AoI continues to increase $\tau_m^{\text{upd}}(t)$ with t, and then returns to the initial value $\tau_m^{\text{upd}}(t)$ when t reaches t_1', t_2' and t_3' moments.

To better design the cached content update scheme, we formulate the rule that the AoI of all cached content cannot exceed the maximum value A_{max}. Otherwise, an update will be forced as bandwidth allows. Regarding the content from the sensing node m, there exist two cases for the AoI value $A_0^m(t)$ of the content stored in the edge server. One case is determined by the duration of the previous time frame. In this case, the cache update is performed at the previous time frame. Another case is determined by the sum of the previous information freshness and the duration for the last time frame. However, the value of AoI cannot exceed A_{max} in the second case or it will be replaced. Among them, $A_0^m(t)$ can be expressed as

$$A_0^m(t) = \begin{cases} T_{\text{span}}(t-1), & \text{if } I_m(t-1) = 1 \\ \min\left\{A_0^m(t-1) + T_{\text{span}}(t-1), A_{max}\right\}, & \text{otherwise} \end{cases} \tag{2}$$

where $T_{\text{span}}(t)$ represents the duration of the time frame t.

For mobile users, they need to obtain the cached information from the edge server. Therefore, the AoI value at this time depends on the freshness of the content and the transmission latency between devices. For a user n requesting the content cached by the sensing node m, the corresponding AoI value $A_n^m(t)$ contains two cases. One case is determined by the information freshness of the content on the server and the downlink transmission latency. In this case, no cache update

is performed at the beginning. Another case is determined by the time consumption consisting of the downlink transmitting and content updating. In this case, cache update is performed at the beginning. Among them, $A_n^m(t)$ can be expressed as

$$A_n^m(t) = \begin{cases} A_0^m(t) + \tau_n^{\text{tra}}(t), & \text{if } I_m(t) = 0 \\ \tau_n^{\text{tra}}(t) + \tau_m^{\text{upd}}(t), & \text{otherwise} \end{cases} \tag{3}$$

where $\tau_m^{\text{upd}}(t)$ represents the uplink transmission latency for sensing node m to update a single task with the edge server, and $\tau_n^{\text{tra}}(t)$ represents the downlink transmission latency for the edge server to deliver cached content with user n. In the following, we describe the calculation procedure of these two transmission latencies.

In our network model, the transmit power of any antenna can be flexibly controlled through adjusting task priority and the number of resource blocks. The number of power resource blocks $Z_n(t)$ allocated to each user is not fixed. Specifically, we model continuous power resources as discrete power resource blocks, each of which is p_0. The allocation of power resource blocks is based on the optimal decision introduced in Section V. The transmit power $p_n^{\text{sup}}(t)$ of the antenna is given by

$$p_n^{\text{sup}}(t) = V_n(t) Z_n(t) p_0 \tag{4}$$

where $V_n(t)$ is the task priority of user n, $Z_n(t)$ is the number of resource blocks assigned to user n, and p_0 is the power resources contained in a unit resource block.

The number of resource blocks allocated to all N users does not exceed G, and then

$$\sum_{n=1}^{N} Z_n(t) \leq G. \tag{5}$$

Therefore, the downlink transmission rate $R_n(t)$ is determined by the transmit power, channel gain, channel bandwidth, and noise power, which can be expressed as

$$R_n(t) = B \log_2 \left(1 + \frac{p_n^{\text{sup}}(t)|h_n|^2}{\sigma^2}\right) \tag{6}$$

where B is the channel bandwidth of the base station, $p_n^{\text{sup}}(t)$ is the power provided by the antenna to the user n, h_n is the channel gain between user n and base station, and σ^2 is noise power. Since we consider an orthogonal frequency-division multiple access (OFDMA) system, there is no channel interference. A signal received at each user can be successfully decoded if and only if the transmission rate from BS to the user is greater than some threshold value R_T, i.e.,

$$R_n(t) \geq R_T. \tag{7}$$

Then, the downlink transmission latency can be expressed as

$$\tau_n^{\text{tra}}(t) = \frac{F_m}{R_n(t)} \tag{8}$$

where F_m is the file size of the cached content, and $R_n(t)$ is the downlink transmission rate.

For the uplink, we use τ_{upd}^m to denote the update latency of any task at sensing node m. Then, the update latencies of all tasks are denoted as a set $T_{\text{upd}} = \{\tau_{\text{upd}}^1, \tau_{\text{upd}}^2, ..., \tau_{\text{upd}}^M\}$. We use τ_{max} to denote the maximum tolerable latency of the task. This means that when the update latency of the task is greater than τ_{max}, the task update is failed.

2) Energy Consumption Model: In the WENT, the energy is mainly consumed at the edge server and sensing nodes. Since edge server usually has a stable power supply, we only consider the energy consumption at sensing nodes. In particular, the energy consumption of sensing nodes consists of the energy of state sensing E_{sen} and the energy of content uploading E_{up}. We use E_{upd} to denote the energy consumption required for each content update, i.e., $E_{\text{upd}} = E_{\text{sen}} + E_{\text{up}}$. Then, the total energy consumption $E_{\text{tot}}(t)$ of the WENT can be expressed as

$$E_{\text{tot}}(t) = \sum_{m=1}^{M} I_m(t) E_{\text{upd}}. \tag{9}$$

V. PROBLEM FORMULATION

A. Overhead Function

We construct an overhead function consisting of AoI demand overhead C_A, system energy consumption overhead C_E, antenna power allocation overhead C_P, and task priority matching overhead C_V. The overhead function $K_{\text{tot}}(t)$ of the system in time frame t can be expressed as

$$K_{\text{tot}}(t) \quad = \eta C_A + \mu C_E + \kappa C_P + \zeta C_V \tag{10}$$

where η is the unit price of the AoI demand overhead, μ is the unit price of the energy consumption overhead, κ is the unit price of the antenna power overhead, and ζ is the unit price of the priority matching overhead. Note that the prices here are all dimensionless units, and different prices imply tradeoffs between different overheads. Unit price plays the role of weight in the overhead function, which affects the importance metrics of different components and thus the performance of the average system utility. By adjusting the prices, a balance can be found between different components to achieve a comprehensive optimization of the system. For example, choosing the option with a higher price for energy consumption will extend the battery life of the device and reduce the energy consumption, but may lead to higher AoI. Choosing the option with higher AoI price will improve response time and quality of service, but may lead to higher energy consumption. In this scheme, we pay more attention to the impact of AoI and system energy consumption, so we give a larger weight. The antenna power allocation overhead and priority matching overhead are considered as additional overheads, so they are given less weight.

Now we need to determine C_A, C_E, C_P, and C_V, respectively. The AoI demand overhead C_A is determined by the intelligence level of network-connected user n and the average demand value of AoI. First, we use $\Delta_n^m(t)$ to denote the AoI demand value, which can be expressed as

$$\Delta_n^m(t) = |A_n^m(t) - A_{sta}| \tag{11}$$

where $A_n^m(t)$ is the AoI in real time, and A_{sta} is the optimal AoI value for each user. We consider the long-term average benefit of the system. Thus, the average AoI demand value

of user n for all caching content in the time frame t can be expressed as

$$\overline{\Delta}_n(t) = \frac{1}{\sum_{m=1}^{M} D_n^m(t)} \sum_{m=1}^{M} \Delta_n^m(t) D_n^m(t). \tag{12}$$

We further give the AoI demand overhead C_A of all users as

$$C_A = \sum_{n=1}^{N} \alpha_n \overline{\Delta}_n(t) \tag{13}$$

where $\sum_{n=1}^{N} \alpha_n = 1$, $\alpha_n \in [0, 1]$, and the value of α_n depends on the intelligence level of the network-connected user.

The system energy consumption overhead C_E represents the total energy consumption that equal to the sum of the energy consumption required for each task given in (9).

The antenna power allocation overhead C_P is defined as the sum of the power allocated to each antenna. There are N antennas equipped at the edge server. Each user can get several power resource blocks for data transmission. The transmit power of each antenna belongs to $[0, p_{max}]$, and can be adjusted via changing the number of resource blocks based on task priority. High priority tasks are allocated more resource blocks. Then, we can determine C_P as

$$C_P = \sum_{n=1}^{N} \sum_{m=1}^{M} D_n^m(t) p_n^{\text{sup}}(t) \tag{14}$$

where p_n^{sup} is the power of each antenna.

The task priority matching overhead C_V represents the additional overhead introduced by considering the priority, which is determined as

$$C_V = \sum_{n=1}^{N} \rho V_n(t) \tag{15}$$

where ρ is the costing factor for priority. In the overhead function, priority matching overhead is an additional overhead incurred by users' priority allocation. Specifically, the high priority users can be allocated more transmit power resources. To obtain priority, these users need to pay fee to the resources' provider, resulting in the priority matching overhead.

B. Optimization Problem

We focus on the long-term average benefit of the system, and thus the average system overhead is given by

$$K_{\text{ave}} = \lim_{T \to \infty} \frac{1}{T} \mathbb{E}\left(\sum_{t=0}^{T} K_{\text{tot}}(t)\right). \tag{16}$$

Our objective is to minimize the average system overhead, which can be formulated as the following constrained optimization problem,

$$\mathbf{P1}: \quad \min \ K_{\text{ave}} \tag{17}$$

$$\text{s.t.} \quad A_0^m(t) \le A_{max}, \ \ m \in \mathcal{M} \tag{17a}$$

$$\sum_{m=1}^{M} I_m(t) \le Y \tag{17b}$$

$$\sum_{n=1}^{N} Z_n(t) \le G \tag{17c}$$

where constraint (17a) represents that the AoI value $A_0^m(t)$ of all cached content cannot exceed the maximum value A_{max} or an update will be forced, constraint (17b) represents the upper bound of the number of updated content, and constraint (17c) represents that the number of resource blocks allocated to all N users does not exceed G.

It is usually challenging to solve the nonlinear and noncovex optimization problem. In the next sections, we propose an RB-DRN scheme to solve this problem.

VI. DYNAMIC CACHE UPDATE SCHEME

This section first establishes an MDP to characterize the cache update process, including joint resource scheduling and status updating. Then we propose an RB-DRN algorithm to solve the optimization problem and analyze its time complexity.

A. MDP Model

We construct the cache updating and resource scheduling process as an MDP process defined by the quaternion tuple $\langle S, A, P, U \rangle$, where S, A, P and U denotes the state space, action space, state transition probability and utility function, respectively. The specific process is described as follows.

- ***State Space S:*** $s(t) = (A_0(t), A_1(t), ..., A_N(t), V_1(t), ..., V_N(t))$ is defined as the system state at time frame t, which is composed of the real-time AoI value on the edge server $A_0(t) = (A_0^1(t), A_0^2(t), ..., A_0^M(t))$, the real-time AoI value of the user $A_n(t) = (A_n^1(t), A_n^2(t), ..., A_n^M(t)), n \in \mathcal{N}$ and the task priority $V_n(t)$ for each user. The state space S is finite due to the following two constraints: the maximum tolerance AoI value and the task priority-based resource allocation.
- ***Action space A:*** $a(t) = (I_1(t), I_2(t), ..., I_M(t), Z_1(t), Z_2(t), ..., Z_N(t))$ is defined as the system action set A in time frame t, which represents the content update decision of the sensing node.
- ***State Transition Probability P:*** $P = S \times A \times S \to [0, 1]$ represents the distribution of the transition probability $P(s' \mid s, a)$ from the system state s to a new system state s' $(s, s' \in S)$ when an action $a \in A$ is chosen, which is mainly affected by environmental changes, such as the user's request arrival rate, the priority of cache items, the threshold for antenna power, and the transmission failure probability, etc. It is worth noting that the probability of exploring the unknown environment can be modeled through a noisy network layer.
- ***Utility Function U:*** $S \times A \to U$ represents a mapping relationship between an input state-action parameter and an output utility value $U(s(t), a(t))$. We aim at minimizing the long-term average overhead K_{ave} given in Eq. (17). Then, the value of the utility function can be defined as $U(s(t), a(t)) = -K_{\text{ave}}$.

Here, we describe the strategy π by the agent (i.e., edge server) action $a(t) \in A$ and the system state $s(t) \in S$. Then, the optimal strategy π^* with state $s(t')$ is given by

$$\pi^* = \arg\max_{\pi} \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{t=0}^{T} U(s(t),a(t)) \mid s(t')\right]. \quad (18)$$

B. RB-DRN Algorithm

The constructed MDP model can characterize the impact of the content update decisions on the system utility function. Thus, we need to design an efficient cache update scheme for achieving the highest utility.

Since partial reinforcement learning approaches explore the long-term average return of the system, we propose a dynamic cache content update scheme (RB-DRN) by combining R-learning [42] with Rainbow DQN method. We improve on the state-value function $V_\pi(s)$ of the DQN-based algorithm as

$$V_\pi(s) = \mathbb{E}\left[\sum_{T=0}^{\infty}(U(s(t+T),\pi(t+T)) - \overline{U}) \mid s(t)\right] \quad (19)$$

and the action-value function $Q_\pi(s,a)$ can be expressed as

$$Q_\pi(s,a) = \mathbb{E}\left[\sum_{T=0}^{\infty}(U(s(t+T),a(t+T)) - \overline{U}) \mid (s,a)\right] \quad (20)$$

where $\overline{U}$ represents the long-term average utility of taking strategy π, which can be expressed as

$$\overline{U} = \lim_{T\to\infty}\frac{1}{T}\sum_{T=1}^{\infty} U(s(t+T),\pi(t+T)). \quad (21)$$

The advantage function $G_\pi(s,a)$ under each action a satisfies the following equation

$$Q_\pi(s,a) = V_\pi(s) + G_\pi(s,a) - \frac{1}{|A|}\sum_{a'\in A}(s,a'). \quad (22)$$

The optimal strategy π^* of MDP can be acquired by using the Bellman function as follows

$$V_{\pi^*}(s) = \max_{a\in A} Q_{\pi^*}(s,a). \quad (23)$$

The framework of our dynamic cache update scheme is depicted in Fig. 4.

In our proposed algorithm, we update the state-action value function at each time frame as

$$Q_\pi(s,a;\theta_1,\theta_2) = V_\pi(s;\theta_1)+G_\pi(s,a;\theta_2)-\frac{1}{|A|}\sum_{a'\in A}(s,a';\theta_2). \quad (24)$$

Initially, we employ a prioritized experience replay scheme [43] to collect a batch of cached experience tuples $\mathbb{R} = \{\xi_1,\xi_2,...,\xi_j,...\xi_J\}$, where $\xi_j = (s_j,a_j,U(s_j,a_j),s'_j)$, $j \in \{1,2,...,J\}$. Here, the size of each batch is W_R. This method of experience replay preferentially sample experiences with high priority. Note that the priority for sampling each experience tuple ξ_j is calculated as $p_j = |\delta_j| + e, j \in \{1,2,...,J\}$. Here, δ_j represents the temporal-difference (TD) error, and the value of e is a very small positive number such as 1×10^{-5} or 1×10^{-6}. This operation ensures that a sample will be selected with some probability even if its TD error is small, thus increasing the diversity and robustness of the training. The TD error δ_j can be expressed as

$$\delta_j = U(s_j,a_j) - \overline{U} + \max_{a'_j} Q(s'_j,a'_j;\theta_1^*,\theta_2^*) - Q(s_j,a_j;\theta_1,\theta_2) \quad (25)$$

where $\overline{U}$ represents the average utility of the cached experiences in the cached experience replay buffer. Among them, sampling can be executed by using the *SumTree* method, which allows experience tuples with higher sampling priority to be selected with greater likelihood. Based on this method, the probability that each experience is sampled in prioritized experience replay can be expressed as

$$Pr_j = \frac{p_j^\phi}{\sum_{j=1}^{J} p_j^\phi} \quad (26)$$

where ϕ represents the parameter for adjusting the influence degree of experience priority. In our proposed algorithm, sampling is based entirely on priority, i.e. $\phi = 1$. In order to correct the bias introduced in the prioritized experience replay, importance sampling weight (ISW) is used to modify the learning update and thus reduce the effect of non-uniform sampling. The ISW is given by

$$W_j = (\frac{1}{JPr_j})^\psi \quad (27)$$

where Pr_j is the probability of sampled experience, and ψ represents the parameter for controlling the influence degree of weights.

The average utility $\overline{U}$ combines the batch size of cached experience tuples and the TD errors, and then it can be updated as

$$\overline{U} = \overline{U} + \lambda\sum_{j=1}^{W_R}\delta_j. \quad (28)$$

According to the update of average utility, the estimated value of the target network $\hat{Q}_t$ is given by

$$\hat{Q}_t(s_j,a_j) = U(s_j,a_j) - \overline{U} + \max_{a'_j\in A} Q(s'_j,a'_j;\theta_1^*,\theta_2^*). \quad (29)$$

Then, the evaluation network and the target network can be updated by minimizing the loss function $L(\theta)$. Here, the loss function is given by

$$L(\theta) = \frac{1}{W_R}\sum_{j=1}^{J} W_j\left((\hat{Q}_t(s_j,a_j) - Q(s_j,a_j;\theta_1,\theta_2)\right)^2. \quad (30)$$

In our proposed algorithm, we conduct training of t_{pre} loops to update the parameters of the network via mini-batch stochastic gradient descent with Adam [44] [45]. The update parameter is given by

$$\theta'_f = \theta_f - \zeta\nabla L(\theta_f), \quad f\in\{1,2\} \quad (31)$$

where ζ is the learning rate. The parameter θ_f for the main network is updated at each time frame, whereas the parameter θ_f^* for the target network is updated every C time frames, such that $\theta_f^*(t) = \theta_f(t-C)$. The proposed RB-DRN algorithm is presented in **Algorithm 1**, which can be summarized as the following steps.

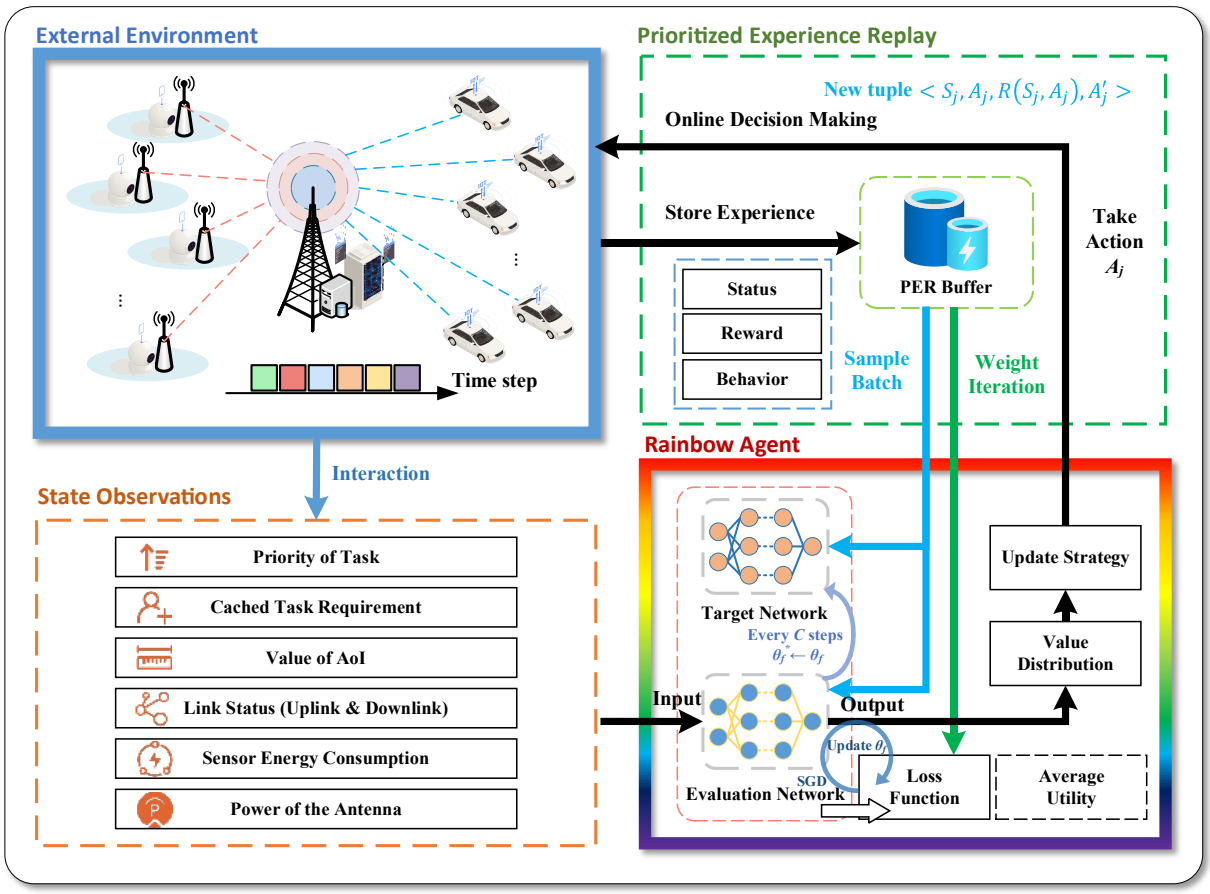


Fig. 4. The framework of RB-DRN.

Step 1: Initialization. At the beginning of the RB-DRN algorithm, the system initializes the parameters of the main and target networks, the average utility and the time parameter.

Step 2: Observing and Acting. The edge server acting as an agent observes its surrounding environment, and then obtains the current system state $s(t)$ by observation. For a given state $s(t)$, the agent chooses action $a(t)$ using the epsilon-greedy method. Since the agent does not know the environment well at the beginning of the iteration under the epsilon-greedy method, it tends to randomly select an action from the action set. After performing several iterations, as the decay and the agent's knowledge of the environment increases, it will select the action with the largest Q value with higher probability. Based on the chosen action $a(t)$, the immediate system utility $U(s(t), a(t))$ is obtained, as well as the state $s(t+1)$ for the next time period.

Step 3: Replaying. The system refreshes the experience buffer. The agent can obtain the corresponding experience tuple $\langle s(t), a(t), U(s(t), a(t)), s(t+1)\rangle$ and cache the tuple in the experience replay buffer for subsequent learning. Subsequently, a small batch is taken from the experience replay buffer and trained using the prioritized experience replay method.

Step 4: Updating. The algorithm updates the average utility function value via Eq. (28). At the same time, the loss function is minimized by the gradient descent method and the network parameters are updated until the utility function converges. The exploration rate is initially set to 1 and is decayed during each training session by multiplying it by an exploration decay factor.

C. Algorithm Complexity Analysis

The time complexity for an Artificial Neural Network (ANN)-based algorithm is influenced by the number of neurons within the network [46]. Consider a fully-connected network with x_I input neurons, x_O output neurons, and H hidden layers, each containing x_h neurons ($h \in \{1, 2, ..., H\}$). The time complexity can be expressed as $\mathcal{O}(x_I x_1 + x_O x_H + \sum_{h=1}^{H-1} x_h x_{h+1})$ [47]. Additionally, the time complexity for the *SumTree* method is $\mathcal{O}(\log_2|\mathbb{R}|)$ [43]. Since this value is small, it can be ignored. Therefore, the overall time complexity of the proposed algorithm is given by

$$\mathcal{O}(|S|(x_{1,S} + x_{1,A}) + \sum_{h=1}^{H-1}(x_{h,S}x_{h+1,S} + x_{h,A}x_{h+1,A}) + x_{H,S} + |A|x_{H,A}) \tag{32}$$

where $x_{h,S}$ and $x_{h,A}$ represent two ANNs of state-value function and advantage function, $|S|$ and $|A|$ represent the

Algorithm 1: RB-DRN: A rainbow deep reinforcement learning-based cache content update scheme.

Input: environmental exploration rate ϵ, exploration decay factor ϵ', cached experience replay buffer size $|\mathbb{R}|$, system state S, learning rate λ and ξ.
Output: value of Q and $\hat{Q}_t$.

▷ **Initialization**
Initialize model parameters θ_f and θ_f^*, and let $\theta_f^* \leftarrow \theta_f$, $f \in \{1, 2\}$;
Initialize average utility $\overline{U} = 0$ and $t = 0$;
▷ **Implementation**
for $t \leq T_{max}$ **do**
  ▷ **Observing & Acting**
  Observe current state $s(t)$ of system;
  Obtain the value of $Q(s(t), a; \theta_1, \theta_2)$ for state $s(t)$;
  Select $a(t)$ from A with probability ϵ or select $a(t) = \arg\max_A Q(s(t), a; \theta_1, \theta_2)$ otherwise;
  Execute $a(t)$;
  Observe and record the system state under the action $a(t)$;
  Calculate the system utility value $U(s(t), a(t))$ based on the current state and obtain the state $s(t+1)$ for next time frame;
  ▷ **Replaying**
  Refreshing the cached experience replay buffer;
  Add the new tuple $\langle S, A, U, S(t+1)\rangle$ to the cached experience replay buffer $\mathbb{R}$;
  Sample the cached experience tuples with batch size W_R from the cached experience replay buffer by prioritized experience replay method;
  ▷ **Updating**
  Update the average utility $\overline{U}$ with Eq.(28);
  Obtaining TD errors and calculating the loss function $L(\theta_f), f \in \{1, 2\}$;
  Update parameters (θ_1,θ_2) with Eq.(31);
  if $mod(t, C) = 0$ **then**
    Set the target network parameters $\theta_1^* \leftarrow \theta_1$ and $\theta_2^* \leftarrow \theta_2$;
  end
  $t \leftarrow t + 1$;
  $\epsilon \leftarrow \epsilon\epsilon'$;
end

sizes of the state space and action space, respectively. According to the MDP and the proposed algorithm, we can express the size of $|S|$ and $|A|$ as

$$\begin{cases} |S| = (1 + M)N + M, \\ |A| = M + N. \end{cases} \tag{33}$$

Assuming that the neurons in the two ANNs have identical parameters, this expression can be simplified as

$$\mathcal{O}(MNx + x^2) \tag{34}$$

where M represents the number of sensing nodes, N represents the number of users, and x represents the number of NNs.

TABLE III
SIMULATION PARAMETERS

Parameters	Values
Mobile users	15, 20, 25, 30
Sensing nodes	10
Maximum power of an antenna	0.5 W
Channel bandwidth of the base station	5 MHz
Time slot	1 s
Uplink transmission latency for sensing nodes	$\{0.8, 0.9, 1.0, 1.1, 1.2\}$ s
Energy consumption for updating	[5,10] mW
Noise power	-114 dBm
Maximum allowable value of AoI	20 s
Size of the cached content	[1,2] MB
Maximum number of updated content	5
Total number of resource blocks	200
Optimizer	Adam [45]
Activation function	ReLU/GELU [49], [50]
Learning rate of utility	0.005, 0.0005, 0.00005
Learning rate of neural network	0.00005
Batch size	32, 64, 128, 256
Cached experience replay buffer size	1×10^5
Iteration step	2000
Training epoch	150

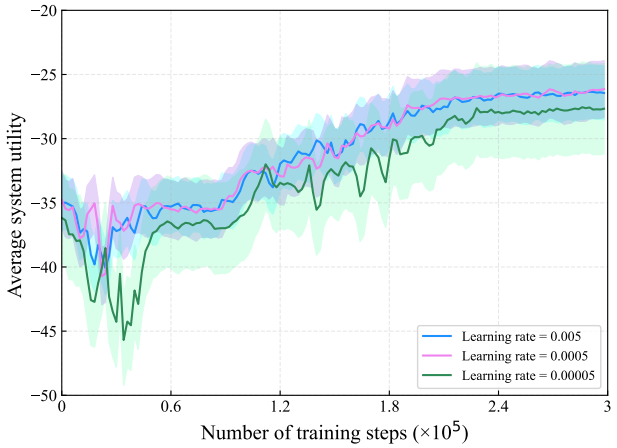


Fig. 5. The average system utility of the RB-DRN algorithm for different learning rates.

VII. SIMULATION AND ANALYSIS

We conduct simulation study to analyze convergence performance of RB-DRN algorithm, and also to illustrate the impact of critical parameters on system performance. We further provide efficiency analysis compared with other methods. The whole simulation is implemented via the TensorFlow frame and runs on a PC with Intel Core i9-10980XE CPU @3.00GHz, Memory 16G, and GPU for NVIDIA GeForce RTX 3090.

A. Parameter Settings

We construct a WENT simulation scenario in which there is a base station equipped with an edge server, N ($N \in \{15, 20, 25, 30\}$) connected vehicles, and 10 sensing nodes. At each time slot, the arriving task requests for each cached HDM file follow a *Zipf* distribution, where the value of the distribution parameter is set to 1.0 [47]. This distribution is representative of real vehicular networks and is widely

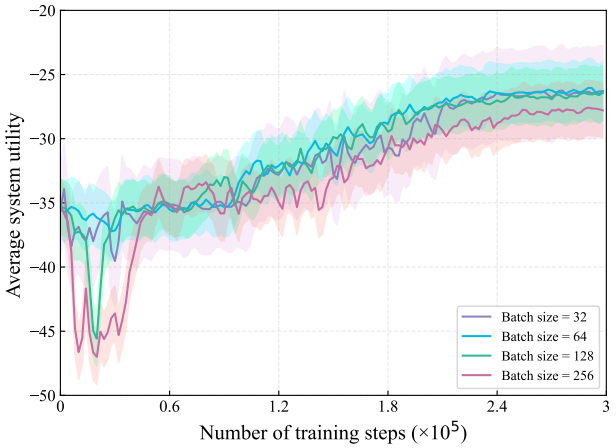


Fig. 6. The average system utility of the RB-DRN algorithm for different batch sizes.

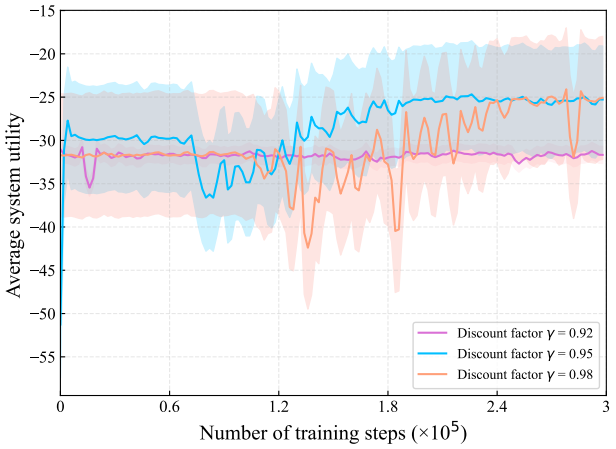


Fig. 7. The average system utility of the PDSU algorithm for different discount factors.

used in many related references [53]-[55]. The BS uses the MIMO model. At the same time, the allocated power of each resource block is 2.5 mW, and the number of resource blocks is set to 200. For each sensing node, the HDM file update latency is randomly selected from the set of values of $\{0.8\tau, 0.9\tau, 1.0\tau, 1.1\tau, 1.2\tau\}$, The value of time slot τ is set to 1 s. Once the file update latencies for each roadside sensor are determined, their values will remain constant throughout the simulation. The AoI maximum tolerance is set to 20 time slots. Meanwhile, the power of each sensor is set to 10 mW. For sensing nodes, each update of the HDM file consumes the same amount of energy. Note that all parameter settings are widely used and verified to be effective in simulating real vehicular networks in previous studies [47],[53]-[57].

Small scale data is not suitable for training deep reinforcement learning and it may face the problem of insufficient exploration. Deep reinforcement learning schemes (e.g., DQN approach) relies on exploration and exploitation mechanisms to learn the dynamics of the environment. With small scale data, exploration may be insufficient, resulting in a model that does not fully understand the environment. The detailed parameter setting is listed in Table III.

TABLE IV
THE TIME CONSUMPTION OF TRAINING FOR DIFFERENT ALGORITHMS (UNIT IN HOUR)

Parameter	RB-DRN	PDSU	Greedy	Random
$N = 15$	4.97	4.61	3.60	3.58
$N = 20$	5.37	5.13	4.16	4.01
$N = 25$	5.65	5.52	4.79	4.78
$N = 30$	5.91	5.85	5.31	5.10

B. Performance Analysis

*1) **Convergence Performance**:* We validate the convergence performance of the algorithm to guarantee its reliability.

Fig. 5 shows the average system utility under the RB-DRN algorithm with different learning rates. We can see that different learning rate settings have a significant impact on the reward profile of the RB-DRN algorithm, and the appropriate learning rate helps the RB-DRN algorithm to converges more quickly to a certain constant. This is due to the following reason. First, a larger learning rate allows the agent to explore the unknown environment quickly and improve the efficiency of convergence. However, a larger learning rate may make the update step of the agent larger, which may cause the model to oscillate around the optimal solution and fail to converge. In the subsequent experiments, we set the learning rate to 0.0005.

Fig. 6 shows the average system utility under the RB-DRN algorithm for different batch sizes. It is obvious that varying batch sizes substantially influence the utilities obtained by the RB-DRN algorithm, and the appropriate batch size helps the model converge quickly. The reasons for this are described below. First, larger batch sizes make the empirical samples required for model training richer, which speeds up the convergence of the model. However, too large batch size makes the model fall into local optimal solutions, which reduces the generalization ability of the model. In the subsequent experiments, we set the batch size to 64.

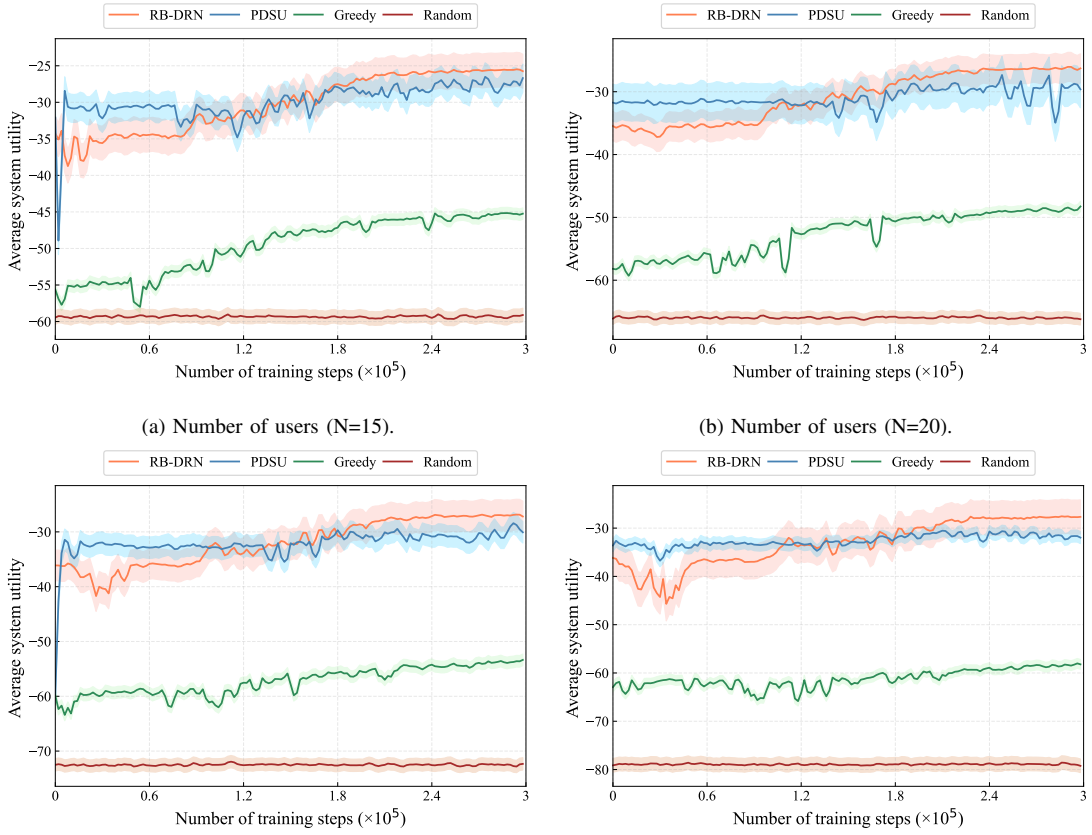
*2) **Efficiency Analysis**:* We conduct a comparison study between the RB-DRN algorithm and the following baseline methods.

Random algorithm (Random): For the current state, the edge server randomly chooses an update action at each time frame. This algorithm does not consider antenna power control, meaning that each user is allocated an equal number of resource blocks.

Greedy algorithm (Greedy): The edge server aims to maximize the immediate utility by implementing the update action at each time frame, i.e., minimum sum of AoI for cached content. This algorithm doesn't take the antenna power control into consideration, which is similar to the Random algorithm.

Prioritized Double DQN-based status update algorithm (PDSU): The network architecture of PDSU algorithm is based on Prioritized Double DQN (DDQN). During each time frame, the Optimization Objective of network is to maximize the cumulative discount return by executing the update action.

Fig. 7 shows the average system utility under the PDSU algorithm for different discount factors, i.e. $\gamma \in$



(a) Number of users (N=15).

(b) Number of users (N=20).

(c) Number of users (N=25).

(d) Number of users (N=30).

Fig. 8. The average system utility of the RB-DRN algorithm for different users.

$\{0.92, 0.95, 0.98\}$. We can see that the discount factor could affect the performance of utility and convergence. The smaller $\gamma = 0.92$ acquires higher convergence speed and lower utility. The larger $\gamma = 0.98$ acquires higher utility, but its convergence performance is unstable. These can be explained as follows. When γ is smaller, the agent is more concerned with immediate benefits and is less difficult to train. When γ is larger, the agent is more concerned with long-term benefits, which may make it difficult for the algorithm to converge. In the subsequent experiments, we set the discount factor $\gamma = 0.95$.

Fig. 8 shows the convergence of the RB-DRN algorithm compared to the other three algorithms. We set the number of users $N \in \{15, 20, 25, 30\}$. we can observe that the greedy algorithm and the Random algorithm obtain lower utility after the model converges. Our proposed RB-DRN algorithm ensures faster convergence with higher utility values. The reasons for this are described below. The RB-DRN algorithm combines the advantages of Rainbow DQN, i.e., it allows the network to learn higher-valued state-action pairs and mitigates the problem of Q-value overestimation. Meanwhile, a prioritized experience replay mechanism is introduced to learn important experiences more frequently. By giving higher learning priority to important samples, the efficiency of estimating the value of key state-action pairs can be improved. Second, we add a Gaussian noise component to the fully connected layer of the network, which improves the generalization ability of the model and makes the algorithm better adapted to unknown environments. In addition, the RB-DRN algorithm incorporates R-Learning, i.e., obtaining higher long-term average utility values without adjusting the discount factor, which also saves computational resources to some extent. The training time consumption of each algorithm is listed in Table IV. It can be seen that our proposed algorithm can incur a large training time consumption. However, the training of our proposed algorithm can be completed offline using historical data. Thus, the offline training time consumption of our proposed algorithm does not degrade system performance for online prediction. We can train offline when computational resources are idle (e.g., at night) to better cope with the need for online prediction in IoV scenarios.

Fig. 9 shows the impact of different number of users on the average system utility. We can see that the average system utility values under different algorithms have a decreasing trend with the increase of the number of users. Particularly, the RB-DRN algorithm has a smaller decreasing trend. This can be explained as follows. As the number of users increases, the number of states observed by the system increases exponentially. As a result, the system needs more frequent cache updates to cope with the demand of the varied tasks. Compared to greedy and random algorithms, learning-based algorithms save unnecessary cache status updates. This is because learning-

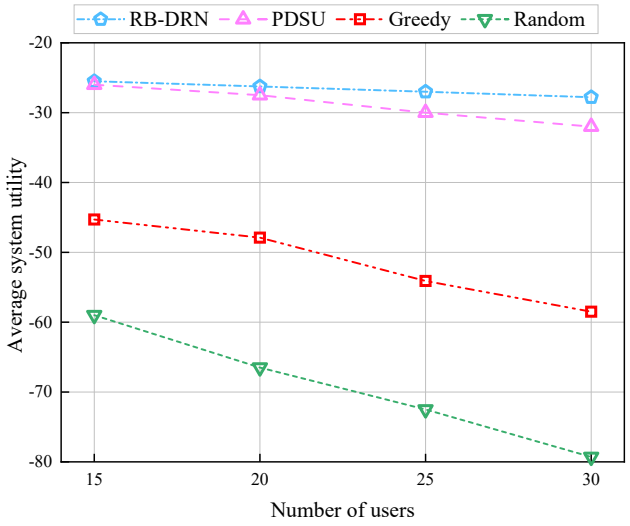


Fig. 9. The impact of different number of users on the average system utility.

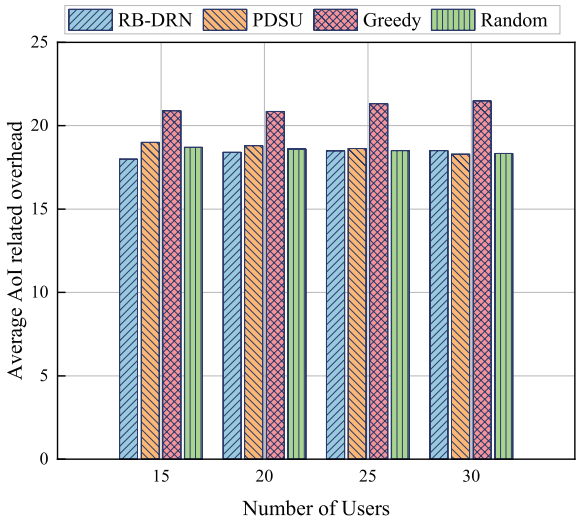


Fig. 10. The impact of different number of users on the average AoI related overhead of the system.

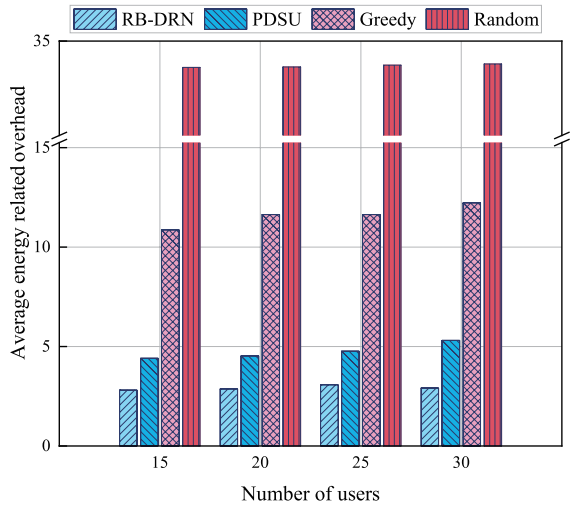


Fig. 11. The impact of different number of users on the average energy related overhead of the system.

TABLE V
THE PERFORMANCE BETWEEN RB-DRN, GUROBI AND THE EXHAUSTIVE METHOD FOR SMALL INSTANCE SCENARIOS

Mobile users	RB-DRN	Gurobi	Optimal solution
N=3	-25.31	-25.06	-25.03
N=4	-25.60	-25.19	-25.10
N=5	-25.95	-25.32	-25.29

based algorithms pay more attention to the long-term utility of the system, while the RB-DRN algorithm also combines the power control and priority matching strategies.

We solve the optimization problem on small instances with the setting of different mobile users using the Gurobi solver, exhaustive method, and our proposed RB-DRN method, respectively. As illustrated in Table V, the optimal solutions are provided to check the optimality gap between our proposed method and traditional mathematical methods (i.e., Gurobi solver, exhaustive method). We can see from Table V that the optimal solutions under our proposed method are almost the same as those under the other two methods. However, the traditional mathematical methods are difficult to solve the optimization problem on large-scale instances due to high time complexity. It is notable that our proposed method is more suitable to solve the large-scale optimization problem.

Fig. 10 shows the impact of different number of users on the average AoI related overhead. We can see that the RB-DRN algorithm maintains a lower average AoI related overhead compared to the other three algorithms. Meanwhile, the effect of different number of users on the average AoI related overhead under the RB-DRN algorithm is small. The reasons are explained as follows. On one hand, the RB-DRN algorithm pays more attention to maximizing the long-term utility of the system, so it can execute the optimal cache update strategy based on the real-time status information and historical status information of the environment. On the other hand, the RB-DRN algorithm combines power control to constrain the variation range of transmission latency, which makes the average AoI related overhead more stable.

Fig. 11 shows the impact of different number of users on the average energy consumption of the system. We can see that the average system energy consumption under the RB-DRN algorithm is lowest compared to all the other three algorithms. The reasons are explained as follows. First, the RB-DRN algorithm accomplishes the tradeoff between AoI related overhead and energy related overhead by maximizing the long-term average utility, i.e., the RB-DRN algorithm does not greatly increase the number of cache updates by reducing the short-term AoI related overhead. Second, the RB-DRN algorithm also combines the power control and priority matching strategies to optimize the content update process, thus avoiding unnecessary updates.

Fig. 12 shows the impact of different number of users on the average number of cached content updates. We can see that the average number of updates of the RB-DRN algorithm stays low compared to all the other three algorithms. The reason is explained as follows. Reinforcement learning-based algorithms can obtain the global optimal solution through the

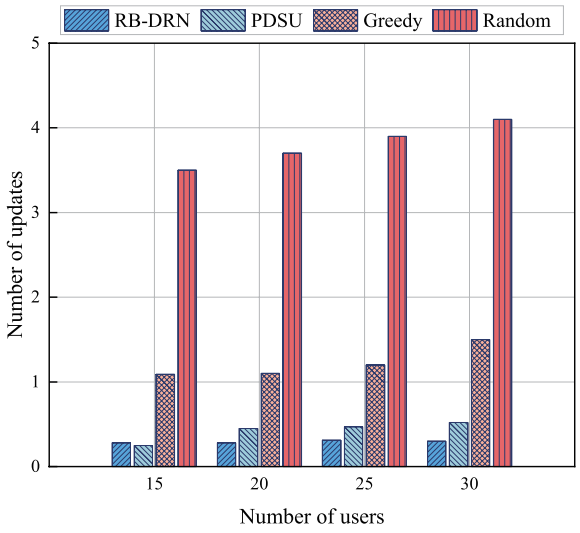


Fig. 12. The average number of cached content updates.

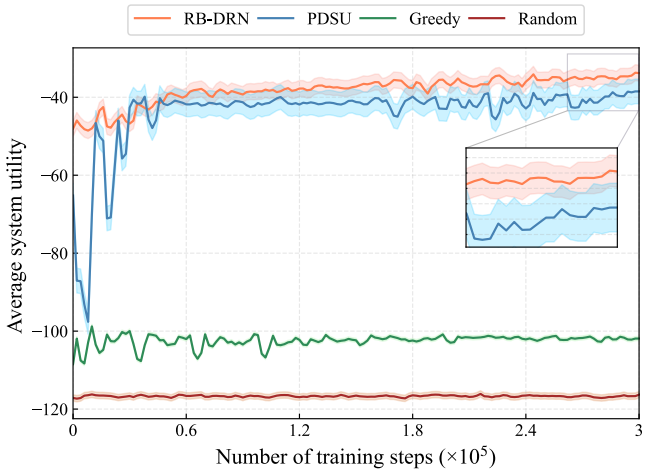


Fig. 13. The average system utility of the RB-DRN algorithm for 100 users.

interaction between the agent and the environment, so the average number of updates of the algorithms is smaller. The greedy algorithm pays more attention to the current system utility and easily obtains the local optimal solution, so the average number of updates of the algorithm is larger. At the same time, the RB-DRN algorithm also combines the power control and priority matching strategies to complete the allocation of power resources to different users, thus optimizing the performance of the system.

We scaled up the experimental size of the edge network, where one edge server equipped at one BS can serve 100 mobile users. Fig. 13 shows the convergence of the RB-DRN algorithm compared to the other three algorithms. We then scaled up the experimental size of the edge network with 10 BSs, 10 servers and 1000 mobile users, where one edge server equipped at one BS can serve 100 mobile users. Fig. 14 shows the convergence of the RB-DRN algorithm compared to the other three algorithms. Similar to the limited input size, our proposed RB-DRN algorithm ensures more stable convergence

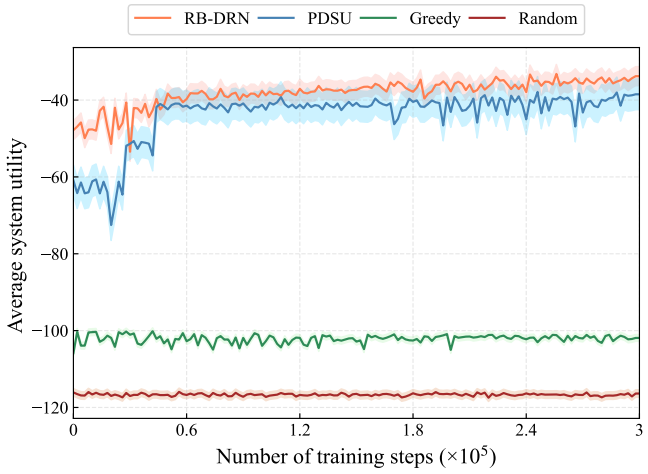


Fig. 14. The average system utility of the RB-DRN algorithm for 1000 users.

and higher utility values.

VIII. CONCLUSION

This paper explored the AoI and energy-driven dynamic cache update scheme in wireless edge networks. Specifically, we formulated it as a nonlinear and nonconvex optimization problem, which aims at minimizing long-term average overhead while ensuring information freshness and low energy consumption. For this purpose, we proposed an RB-DRN algorithm fully leveraging the advantages of both Rainbow DQN and R-learning. Simulation results show that our proposed RB-DRN algorithm can achieve higher system utility, higher information freshness, and lower system energy consumption in comparison with the benchmark algorithms. An interesting study is to explore the cache update schemes in the case of high-level autonomous driving in our future work, where the sensing node can actively update the cache context, when it finds a significant change of its sensing environment.

REFERENCES

[1] Y. Chen, Y. Sun, B. Yang and T. Taleb, "Joint Caching and Computing Service Placement for Edge-Enabled IoT Based on Deep Reinforcement Learning," in IEEE Internet of Things Journal, vol. 9, no. 19, pp. 19501-19514, 1 Oct.1, 2022.

[2] X. Wang, R. Li, C. Wang, X. Li, T. Taleb and V. C. M. Leung, "Attention-Weighted Federated Deep Reinforcement Learning for Device-to-Device Assisted Heterogeneous Collaborative Edge Caching," in IEEE Journal on Selected Areas in Communications, vol. 39, no. 1, pp. 154-169, Jan. 2021.

[3] Y. Yuan, W. Su, G. Hong, H. Li and C. Wang, "A Joint Caching and Offloading Strategy Using Reinforcement Learning for Multi-access Edge Computing Users," Mobile Networks and Applications, 1-14, 2024.

[4] H. Djigal, J. Xu, L. Liu and Y. Zhang, "Machine and Deep Learning for Resource Allocation in Multi-Access Edge Computing: A Survey," in IEEE Communications Surveys & Tutorials, vol. 24, no. 4, pp. 2449-2494, Fourthquarter 2022.

[5] Z. Ding, J. Xu, O. A. Dobre and H. V. Poor, "Joint Power and Time Allocation for NOMA–MEC Offloading," in IEEE Transactions on Vehicular Technology, vol. 68, no. 6, pp. 6207-6211, June 2019.

[6] T. K. Rodrigues, J. Liu and N. Kato, "Application of Cybertwin for Offloading in Mobile Multiaccess Edge Computing for 6G Networks," in IEEE Internet of Things Journal, vol. 8, no. 22, pp. 16231-16242, 15 Nov.15, 2021.

[7] Y. Dang, C. Benzaïd, B. Yang, T. Taleb and Y. Shen, "Deep-Ensemble-Learning-Based GPS Spoofing Detection for Cellular-Connected UAVs," in IEEE Internet of Things Journal, vol. 9, no. 24, pp. 25068-25085, 15 Dec.15, 2022.

[8] O. El Marai, T. Taleb and J. Song, "AR-Based Remote Command and Control Service: Self-Driving Vehicles Use Case," in IEEE Network, vol. 37, no. 3, pp. 170-177, May/June 2023.

[9] B. Jedari, G. Premsankar, G. Illahi, M. D. Francesco, A. Mehrabi and A. Ylä-Jääski, "Video Caching, Analytics, and Delivery at the Wireless Edge: A Survey and Future Directions," in IEEE Communications Surveys & Tutorials, vol. 23, no. 1, pp. 431-471, Firstquarter 2021.

[10] M. Chen, U. Challita, W. Saad, C. Yin and M. Debbah, "Artificial Neural Networks-Based Machine Learning for Wireless Networks: A Tutorial," in IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3039-3071, Fourthquarter 2019.

[11] S. Jošilo and G. Dán, "Joint Wireless and Edge Computing Resource Management With Dynamic Network Slice Selection," in IEEE/ACM Transactions on Networking, vol. 30, no. 4, pp. 1865-1878, Aug. 2022.

[12] D. Ren, X. Gui and K. Zhang, "Adaptive Request Scheduling and Service Caching for MEC-Assisted IoT Networks: An Online Learning Approach," in IEEE Internet of Things Journal, vol. 9, no. 18, pp. 17372-17386, 15 Sept.15, 2022.

[13] C. Marina Martinez, M. Heucke, F. -Y. Wang, B. Gao and D. Cao, "Driving Style Recognition for Intelligent Vehicle Control and Advanced Driver Assistance: A Survey," in IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 3, pp. 666-676, March 2018.

[14] J. Zhang and K. B. Letaief, "Mobile Edge Intelligence and Computing for the Internet of Vehicles," in Proceedings of the IEEE, vol. 108, no. 2, pp. 246-261, Feb. 2020.

[15] W. Duan, J. Gu, M. Wen, G. Zhang, Y. Ji and S. Mumtaz, "Emerging Technologies for 5G-IoV Networks: Applications, Trends and Opportunities," in IEEE Network, vol. 34, no. 5, pp. 283-289, September/October 2020.

[16] A. M. Vegni and V. Loscrí, "A Survey on Vehicular Social Networks," in IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2397-2419, Fourthquarter 2015.

[17] F. Tang, B. Mao, N. Kato and G. Gui, "Comprehensive Survey on Machine Learning in Vehicular Network: Technology, Applications and Challenges," in IEEE Communications Surveys & Tutorials, vol. 23, no. 3, pp. 2027-2057, thirdquarter 2021.

[18] M. Bastopcu and S. Ulukus, "Information Freshness in Cache Updating Systems," in IEEE Transactions on Wireless Communications, vol. 20, no. 3, pp. 1861-1874, March 2021.

[19] Z. Zhang, C. -H. Lung, X. Wei, M. Chen, S. Chatterjee and Z. Zhang, "In-Network Caching for ICN-Based IoT (ICN-IoT): A Comprehensive Survey," in IEEE Internet of Things Journal, vol. 10, no. 16, pp. 14595-14620, 15 Aug.15, 2023.

[20] T. Taleb, Z. Nadir, H. Flinck and J. Song, "Extremely Interactive and Low-Latency Services in 5G and Beyond Mobile Systems," in IEEE Communications Standards Magazine, vol. 5, no. 2, pp. 114-119, June 2021.

[21] S. Zhang, L. Wang, H. Luo, X. Ma and S. Zhou, "AoI-Delay Tradeoff in Mobile Edge Caching With Freshness-Aware Content Refreshing," in IEEE Transactions on Wireless Communications, vol. 20, no. 8, pp. 5329-5342, Aug. 2021.

[22] J. Cao, X. Zhu, Y. Jiang and Z. Wei, "Can AoI and Delay be Minimized Simultaneously with Short-Packet Transmission?," IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Vancouver, BC, Canada, 2021, pp. 1-6.

[23] S. Zhang, J. Li, H. Luo, J. Gao, L. Zhao and X. Sherman Shen, "Low-Latency and Fresh Content Provision in Information-Centric Vehicular Networks," in IEEE Transactions on Mobile Computing, vol. 21, no. 5, pp. 1723-1738, 1 May 2022.

[24] A. Arafa and S. Ulukus, "Timely Updates in Energy Harvesting Two-Hop Networks: Offline and Online Policies," in IEEE Transactions on Wireless Communications, vol. 18, no. 8, pp. 4017-4030, Aug. 2019.

[25] Y. Gu, H. Chen, Y. Zhou, Y. Li and B. Vucetic, "Timely Status Update in Internet of Things Monitoring Systems: An Age-Energy Tradeoff," in IEEE Internet of Things Journal, vol. 6, no. 3, pp. 5324-5335, June 2019.

[26] J. Gong, J. Zhu, X. Chen and X. Ma, "Sleep, Sense or Transmit: Energy-Age Tradeoff for Status Update With Two-Threshold Optimal Policy," in IEEE Transactions on Wireless Communications, vol. 21, no. 3, pp. 1751-1765, March 2022.

[27] H. Huang, D. Qiao and M. C. Gursoy, "Age-Energy Tradeoff in Fading Channels with Packet-Based Transmissions," IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 2020, pp. 323-328.

[28] X. Wu, J. Yang and J. Wu, "Optimal Status Update for Age of Information Minimization With an Energy Harvesting Source," in IEEE Transactions on Green Communications and Networking, vol. 2, no. 1, pp. 193-204, March 2018,.

[29] T. Zhang et al., "AoI and PAoI in the IoT-Based Multisource Status Update System: Violation Probabilities and Optimal Arrival Rate Allocation," in IEEE Internet of Things Journal, vol. 10, no. 23, pp. 20617-20632, 1 Dec.1, 2023.

[30] X. Zheng, S. Zhou and Z. Niu, "Urgency of Information for Context-Aware Timely Status Updates in Remote Control Systems," in IEEE Transactions on Wireless Communications, vol. 19, no. 11, pp. 7237-7250, Nov. 2020.

[31] B. Zhou and W. Saad, "Joint Status Sampling and Updating for Minimizing Age of Information in the Internet of Things," in IEEE Transactions on Communications, vol. 67, no. 11, pp. 7468-7482, Nov. 2019.

[32] S. Leng and A. Yener, "Age of Information Minimization for an Energy Harvesting Cognitive Radio," in IEEE Transactions on Cognitive Communications and Networking, vol. 5, no. 2, pp. 427-439, June 2019.

[33] C. Xu, Y. Xie, X. Wang, H. H. Yang, D. Niyato and T. Q. S. Quek, "Optimal Status Update for Caching Enabled IoT Networks: A Dueling Deep R-Network Approach," in IEEE Transactions on Wireless Communications, vol. 20, no. 12, pp. 8438-8454, Dec. 2021.

[34] G. Hong, B. Yang, W. Su, H. Li, Z. Huang and T. Taleb, "Joint Content Update and Transmission Resource Allocation for Energy-Efficient Edge Caching of High Definition Map," in IEEE Transactions on Vehicular Technology.

[35] J. Shi, J. Du, J. Wang, J. Wang and J. Yuan, "Priority-Aware Task Offloading in Vehicular Fog Computing Based on Deep Reinforcement Learning," in IEEE Transactions on Vehicular Technology, vol. 69, no. 12, pp. 16067-16081, Dec. 2020.

[36] R. Chai, M. Li, T. Yang and Q. Chen, "Dynamic Priority-Based Computation Scheduling and Offloading for Interdependent Tasks: Leveraging Parallel Transmission and Execution," in IEEE Transactions on Vehicular Technology, vol. 70, no. 10, pp. 10970-10985, Oct. 2021.

[37] Q. He, x. jiang, N. Guan and Z. Guo, "Intra-Task Priority Assignment in Real-Time Scheduling of DAG Tasks on Multi-Cores," in IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 10, pp. 2283-2295, 1 Oct. 2019.

[38] Y. Yuan, C. Yi, B. Chen, Y. Shi and J. Cai, "A Computation Offloading Game for Jointly Managing Local Pre-Processing Time-Length and Priority Selection in Edge Computing," in IEEE Transactions on Vehicular Technology, vol. 71, no. 9, pp. 9868-9883, Sept. 2022.

[39] N. C. Luong et al., "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," in IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3133-3174, Fourthquarter 2019.

[40] M. -A. Lahmeri, M. A. Kishk and M. -S. Alouini, "Artificial Intelligence for UAV-Enabled Wireless Networks: A Survey," in IEEE Open Journal of the Communications Society, vol. 2, pp. 1015-1040, 2021.

[41] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," AAAI, vol. 30, no. 1, Mar. 2016.

[42] A. Schwartz, "A reinforcement learning method for maximizing undiscounted rewards," in Proc. ICML, 1993, pp. 298-305.

[43] Schaul, Tom and Quan, John and Antonoglou, Ioannis and Silver, David, "prioritized experience replay," 2015, arXiv:1511.05952. [Online]. Available: https://arxiv.org/abs/1511.05952.

[44] S. Amari, "Backpropagation and stochastic gradient descent method," in Neurocomputing, pp. 185-196, 1993.

[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization,"in Computer Science, 2014.

[46] J. Wu, C. Leng, Y. Wang, Q. Hu and J. Cheng, "Quantized Convolutional Neural Networks for Mobile Devices," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4820-4828.

[47] F. Wu, H. Zhang, J. Wu, Z. Han, H. V. Poor and L. Song, "UAV-to-Device Underlay Communications: Age of Information Minimization by Multi-Agent Deep Reinforcement Learning," in IEEE Transactions on Communications, vol. 69, no. 7, pp. 4461-4475, July 2021.

[48] A. Sadeghi, F. Sheikholeslami and G. B. Giannakis, "Optimal and Scalable Caching for 5G Using Reinforcement Learning of Space-Time Popularities," in IEEE Journal of Selected Topics in Signal Processing, vol. 12, no. 1, pp. 180-190, Feb. 2018.

[49] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[50] D. Hendrycks and K. Gimpel, "Gaussian Error Linear Units (GELUs)," 2016, arXiv e-prints, arXiv-1606.

[51] X. He, C. You and T. Q. S. Quek, "Age-Based Scheduling for Mobile Edge Computing: A Deep Reinforcement Learning Approach," in IEEE Transactions on Mobile Computing, Early Access.
[52] X. Xu, S. Gao and M. Tao, "Distributed Online Caching for High-Definition Maps in Autonomous Driving Systems," in IEEE Wireless Communications Letters, vol. 10, no. 7, pp. 1390-1394, July 2021.
[53] S. Krishnan, M. Afshang and H. S. Dhillon, "Effect of Retransmissions on Optimal Caching in Cache-Enabled Small Cell Networks," in IEEE Transactions on Vehicular Technology, vol. 66, no. 12, pp. 11383-11387, Dec. 2017.
[54] T. Liu, S. Zhou and Z. Niu, "Joint Optimization of Cache Allocation and Content Placement in Urban Vehicular Networks," 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 2018, pp. 1-6.
[55] C. Hou, C. Zhou, Q. Huang and C. -B. Yan, "Cache Control of Edge Computing System for Tradeoff Between Delays and Cache Storage Costs," in IEEE Transactions on Automation Science and Engineering, Early Access.
[56] B. Abolhassani, J. Tadrous and A. Eryilmaz, "Optimal Load-Splitting and Distributed-Caching for Dynamic Content Over the Wireless Edge," in IEEE/ACM Transactions on Networking, Early Access.
[57] M. -C. Lee and A. F. Molisch, "Optimal Delay-Outage Analysis for Noise-Limited Wireless Networks With Caching, Computing, and Communications," in IEEE Transactions on Wireless Communications, vol. 22, no. 2, pp. 1417-1431, Feb. 2023.
[58] The code and data for RB-DRN model. [Online]. Available: https://github.com/ACCIO-Y97/CACHE-UPDATE.

Yuan Yuan was born in 1997, and received his bachelor's degree from Beijing Jiaotong University in 2018. He is now a doctoral student at the National Engineering Research Center of Advanced Network Technologies, Beijing Jiaotong university. He is mainly engaged in the research of the vehicular network and multi-access edge computing.

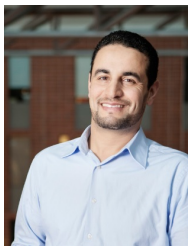
Bin Yang received his Ph.D. degree in systems information science from Future University Hakodate, Japan in 2015. He was a research fellow with the School of Electrical Engineering, Aalto University, Finland, from Nov. 2019 to Nov. 2021. He is currently a professor with the School of Computer and Information Engineering, Chuzhou University, China. His research interests include unmanned aerial vehicle networks, cyber security, edge computing, and Internet of Things.

Wei Su was born in October 1978. He got the Ph.D. degrees in Communication and Information Systems from Beijing Jiaotong University in January 2008. Now he is a professor in the School of Electronics and Information Engineering, Beijing Jiaotong University. Dr. Su Wei is mainly engaged in researching key theories and technologies for the next generation Internet and has taken part in many national projects such as National Basic Research Program(also called 973 Program), the Projects of Development Plan of the State High Technology Research, the National Natural Science Foundation of China. He currently presides over the research project Fundamental Research on Cognitive Services and Routing of Future Internet, a project funded by the National Natural Science Foundation of China.

Haoru Li was born in Tianjin in 1999. He is currently pursuing his M.S. degree at the National Engineering Research Center of Advanced Network Technologies, Beijing Jiaotong University. He is mainly engaged in the research of vehicular service pre-caching and migration in the edge cloud network.

Wang Chang was born in 2000, and is currently a master's student at the National Engineering Research Center of Advanced Network Technologies, Beijing Jiaotong university. He is mainly engaged in the research of vehicle network and edge caching.

Qi Liu received the B.S. degree in information and communication engineering and Ph.D. degree in communication and information system from Beijing Jiaotong University, Beijing, China, in 2003 and 2009, respectively. Then she works as a post-doctor in electronic engineering department in Tsinghua University from 2009 to 2011. She is currently a professorate senior engineer in Smart City Research Institute of China Unicom. Her research interests focus on 5G, cooperation of heterogeneous networks, Internet of Vehicles and High-Precision Positioning.



Tarik Taleb received the B.E. degree Information Engineering with distinction and the M.Sc. and Ph.D. degrees in Information Sciences from Tohoku University, Sendai, Japan, in 2001, 2003, and 2005, respectively. He is currently a Full Professor with the Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum. He is the founder and director of the MOSA!C Lab (www.mosaic-lab.org). Between Oct. 2014 and Dec. 2021, he was a Professor at the School of Electrical Engineering, Aalto University, Finland. Prior to that, he was working as Senior Researcher and 3GPP Standards Expert at NEC Europe Ltd, Heidelberg, Germany. Before joining NEC and till Mar. 2009, he worked as assistant professor at the Graduate School of Information Sciences, Tohoku University, Japan, in a lab fully funded by KDDI, the second largest mobile operator in Japan. From Oct. 2005 till Mar. 2006, he worked as research fellow at the Intelligent Cosmos Research Institute, Sendai, Japan. His research interests lie in the field of telco cloud, network softwarization & network slicing, AI-based software defined security, immersive communications, mobile multimedia streaming, and next generation mobile networking. He has been also directly engaged in the development and standardization of the Evolved Packet System as a member of 3GPP's System Architecture working group. He served as the general chair of the 2019 edition of the IEEE Wireless Communications and Networking Conference (WCNC'19) held in Marrakech, Morocco. He was the guest editor in chief of the IEEE JSAC Series on Network Softwarization & Enablers. He was on the editorial board of the IEEE Transactions on Wireless Communications, IEEE Wireless Communications Magazine, IEEE Journal on Internet of Things, IEEE Transactions on Vehicular Technology, IEEE Communications Surveys & Tutorials, and a number of Wiley journals. Till Dec. 2016, he served as chair of the Wireless Communications Technical Committee.