Computer Systems Science & Engineering
DOI:10.32604/csse.2022.022103
Article

Tech Science Press

Deep Reinforcement Learning Empowered Edge Collaborative Caching Scheme for Internet of Vehicles

Xin Liu[1], Siya Xu[1], Chao Yang[2], Zhili Wang[1,*], Hao Zhang[3], Jingye Chi[1] and Qinghan Li[4]

[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China
[2]Information and Communication Branch, State Grid Liaoning Electric Power Co., Ltd., Shenyang, 110000, China
[3]China Global Energy Interconnection Research Institute, Nanjing, 210000, China
[4]Syracuse University, New York, 13244, USA
*Corresponding Author: Zhili Wang. Email: zlwang@bupt.edu.cn
Received: 27 July 2021; Accepted: 30 August 2021

Abstract: With the development of internet of vehicles, the traditional centralized content caching mode transmits content through the core network, which causes a large delay and cannot meet the demands for delay-sensitive services. To solve these problems, on basis of vehicle caching network, we propose an edge collaborative caching scheme. Road side unit (RSU) and mobile edge computing (MEC) are used to collect vehicle information, predict and cache popular content, thereby provide low-latency content delivery services. However, the storage capacity of a single RSU severely limits the edge caching performance and cannot handle intensive content requests at the same time. Through content sharing, collaborative caching can relieve the storage burden on caching servers. Therefore, we integrate RSU and collaborative caching to build a MEC-assisted vehicle edge collaborative caching (MVECC) scheme, so as to realize the collaborative caching among cloud, edge and vehicle. MVECC uses deep reinforcement learning to predict what needs to be cached on RSU, which enables RSUs to cache more popular content. In addition, MVECC also introduces a mobility-aware caching replacement scheme at the edge network to reduce redundant cache and improving cache efficiency, which allows RSU to dynamically replace the cached content in response to the mobility of vehicles. The simulation results show that the proposed MVECC scheme can improve cache performance in terms of energy cost and content hit rate.

Keywords: Internet of vehicles; vehicle caching network; collaborative caching; caching replacement; deep reinforcement learning

1 Introduction

With the development of internet of vehicles, the content demand of mobile vehicles has increased rapidly. Artificial intelligence-driven vehicles need to constantly learn their surrounding environment and make instant decisions. Vehicles can be regarded as mobile devices to collect and process environmental data and support various information services. The development of Internet of Vehicles (IoV) and wireless technology provides a way to improve the driving experience [1]. Due to high latency and network energy consumption [2], traditional mobile cloud computing is not suitable for latency-sensitive applications [3,4], which affects the experience of vehicle users.

To solve the problems of high latency and energy consumption in IoV, road side units (RSU) are used to help collect vehicle data and provide vehicle-based information services, mobile edge computing (MEC) is used to calculate content popularity [5–7]. Caching popular content in RSU can bring valuable data, thereby reducing access latency and network energy consumption. However, the caching performance is limited by the RSU storage capacity. The storage capacity of single RSU is not enough to support the intensive content requests in IoV, which leads to an unbalanced and redundant cache in adjacent RSUs [8,9]. In addition, due to the high-speed mobility, the vehicle will quickly pass through its coverage region when sending a request to RSU, which means that the RSU cached content is easily out of date [10,11]. In order to improve the caching efficiency, the caching scheme should respond to the vehicle mobility and enable the RSU to selectively replace part of the cached content.

Therefore, an intelligent collaborative caching mechanism is needed to improve caching efficiency. Considering that collaborative caching can enable the cached content to be shared between RSUs, the implementation of collaborative caching at the edge of IoV can alleviate caching load of a single RSU. In addition, vehicle-to-vehicle (V2V) communication can further shorten the transmission distance and reduce the transmission delay. Therefore, we integrate MEC, collaborative caching and V2V caching to realize collaborative caching among cloud, RSU and vehicle.

However, the complex and dynamically changing IoV network environment makes it difficult to determine what content should be cached on RSU. Moreover, the caching optimization problem is a long-term mixed integer linear programming (LT-MILP), which has been proven to be NP-hard [12]. Through neural networks to continuously interact with the environment, deep reinforcement learning (DRL) has excellent performance in dealing with complex dynamic environments.

For this reason, we propose a MEC-assisted vehicle edge collaborative caching (MVECC) scheme based on the vehicle caching network. MVECC uses DRL to predict content popularity and cache popular content in RSUs. Each RSU shares the cached content to achieve collaborative caching. In MVECC, RSUs collect and upload the learning data to macro base station (MBS). Then MBS acts as a DRL agent to determine content caching and delivery schemes. In addition, we have also designed a mobility-aware caching replacement algorithm, RSU can selectively replace and update the cached content according to vehicle mobility. Our contributions can be summarized as follows:

- We build a MVECC scheme. In this scheme, we make full use of the caching capability of RSU and smart vehicles and design two content caching modes, including RSU caching and assisted caching vehicle (ACV) caching. In addition, MVECC includes four content delivery modes, namely content center server (CCS) delivery, RSU direct delivery, RSU collaborative indirect delivery and ACV assisted delivery. MVECC can maximize performance through the coordination of multiple caching and delivery modes.
- We propose a content caching algorithm based on deep deterministic policy gradient (DDPG) and a mobility-aware caching replacement algorithm. Through the learning process of DRL, MVECC continuously adapt to the dynamic changes of IoV network and determine caching scheme based on content popularity. We use Markov chains to simulate state changes. The content transmission energy consumption is regarded as a criterion for supervising the learning process. Agent decide the caching and delivery scheme based on historical experience. At the same time, caching replacement enables RSUs to replace the cached content in response to vehicle mobility, which ensures that RSUs do not store outdated and unpopular content.

2 Related Works

In order to solve the caching optimization problem, many excellent works have studied mobile edge computing and caching methods in wireless networks. In terms of caching allocation, Huang et al. [13,14] considered the best combination of multiple transmissions. Campolo et al. [15] evaluated a solution that can reduce V2R link gap and improve the utilization of wireless channel. Kwon et al. [16] encoded the network and realizes data transmission between multiple RSUs. Mei et al. [17] optimized the radio resources and coding scheme of V2V communication to reduce transmission delay.

In addition, based on the similarity and population of the user community in the V2V communication scenario, Zhao et al. [18] proposed an effective caching scheme to improve the content hit rate. Ding et al. [19] proposed a cache-assisted data transmission scheme, which uses a large number of cache servers deployed on the roadside to support vehicle applications. To solve the data loss caused by vehicle mobility, Abdelhamid et al. [20,21] proposed a vehicle-assisted content caching scheme to distribute content at each exit and entrance of road segment.

Above works solves the cache optimization problem in IoV with the help of edge caching, the cache efficiency of these schemes in processing intensive service requests is still limited by the storage capacity of RSU. Specifically, adjacent RSUs tend to cache the same content, resulting in redundant caching. Through the collaboration between RSUs, MBS can make cache decisions for each RSU from a global perspective and effectively avoid redundant cache.

Considering the randomness of content popularity and IoV network state, it is very complicated to make effective caching decisions in IoV network. As an intelligent learning solution, DRL can effectively overcome these problems and find an optimal caching scheme. There have been some works focusing on DRL-based caching solutions. To solve the problem of continuously acting variables in Markov decision process (MDP) model, Li et al. [22] used a deterministic strategy gradient learning algorithm to provide the optimal resource pricing strategy. Sadeghi et al. [23] introduced the concept of global content popularity and proposed a novel RL-based content caching scheme to implement the best caching strategy.

For reader convenience, the above references are presented in Tab. 1. Although several excellent caching schemes have been proposed to improve resource utilization and vehicle user experience. But most of these studies ignore the high-speed mobility of vehicle. After the vehicle moves, the RSU passing by the vehicle still caches the requested content, which may cause a waste of storage resources. To minimize the energy cost of vehicle caching network, this article introduce an artificial intelligence-assisted caching scheme, which allows multiple caching and delivery modes to coexist, and obtains the best caching and delivery by applying DRL. The simulation results show that we have proposed an effective and feasible reference scheme for vehicle caching network.

3 System Model

3.1 Network Model

In this section, we propose a DRL-based MVECC scheme. As shown in Fig. 1, the MVECC scheme includes a content center server (CCS), a macro base station (MBS), multiple RSUs and assisted caching vehicles (ACVs). We define RSU and ACV as caching nodes, suppose set $\mathcal{K}=\{0, 1, 2, \ldots, K\}$, $\mathcal{E}=\{1,2, \ldots, E\}$ denote RSU and ACV respectively (where k = 0 denotes CCS), $\mathcal{M}=\{1,2, \ldots ,M\}$ represents mobile request vehicle (MRV), which sends out a content request. Let $F=\{1, \ldots, F\}$. represent all content set, CCS stores all content files and can always satisfy any content request from MRV. RSUs are connected by optical fiber to realize the collaborative caching between cloud, edge and vehicles.

Table 1: Comparison of related works

Simple cache optimization with edge cache [13–17]	Cache optimization scheme considering popularity and V2V [18–21]	Cache optimization scheme under DRL [22–23]	Our paper
Cache content at edge nodes	Cache content at edge nodes	Cache content at edge nodes	Cache content at edge nodes
No collaboration between edge nodes	Considering popularity and V2V	Solve optimization problem by DRL	Considering popularity and V2V
No consideration about redundant cache	No collaboration between edge nodes	No collaboration between edge nodes	Collaborative caching between RSUs
	No consideration about redundant cache	No consideration about redundant cache	Considering redundant cache
			Solve optimization problem by DRL

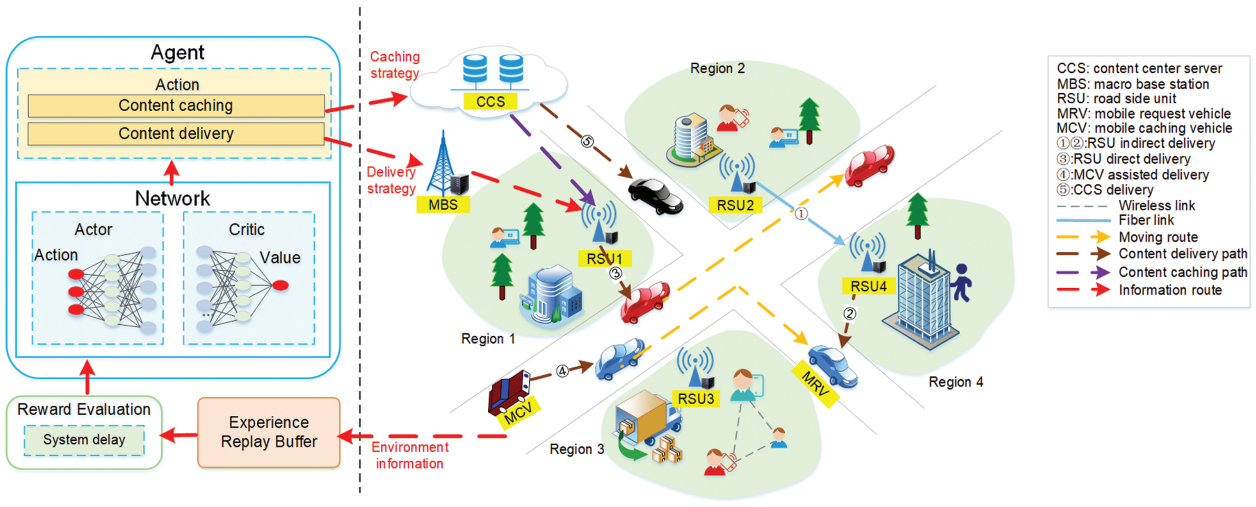

Figure 1: System model

The proposed MVECC scheme uses DRL agents to monitor environment and extract vehicle and content features to estimate vehicle mobility and content popularity. By using Deep Deterministic Policy Gradient (DDPG) algorithm and caching replacement scheme, content can be cached and delivered more efficiently at edge network. DDPG algorithm are calculated and executed on MBS, the content is cached on RSU and ACV. Fig. 1 shows the proposed scheme, where on the left is a DRL module, responsible for determining the best caching and delivery decision. On the right is a MVECC network, where the content is cached from CCS to RSU and ACV, and finally delivered to MRV.

In MVECC, MBS is acted as a DRL agent to monitor the network environment, DDPG are calculated and executed on RSU. In content caching process, caching nodes caches content from CCS according to content popularity; in content delivery process, MRV requests content, MBS selects the optimal delivery path to complete content delivery with lower system energy consumption. Specifically, if the requested content is cached in MRV itself, the content can be obtained without any system energy consumption. Otherwise, the content will be obtained from the communication range, such as obtaining content from ACV or RSU in communication region of MRV. When neither RSU nor ACV within MRV communication range has cached the required content, it will consider forwarding the cached content from adjacent RSU to implement the delivery process. When the cache nodes cannot deliver the cached content, the content will be obtained from CCS.

3.2 Content Caching and Delivery Model

Considering that the content popularity and location of MRV/ACV are time-varying and uncertain, the time-varying scale of content popularity update is much larger than MRV/ACV location changes. Therefore, the content caching process and delivery process should be constructed based on different time scales. As shown in Fig. 2, the large time scale is content caching process. Let $\mathbf{X}=\{1,\ldots,X\}$ be the set of content caching time slot, t_x represents the x_{th} caching time slot, the content caching decision is made at the beginning of caching time slot. The small time scale is content delivery process, divide each t_x time slot into y small time slots. Let $\mathbf{Y}=\{1,\ldots,Y\}$ be the set of content delivery time slot, t_x^y represents the y_{th} delivery time slot in x_{th} caching time slot, content delivery decision is decided at delivery time slot.

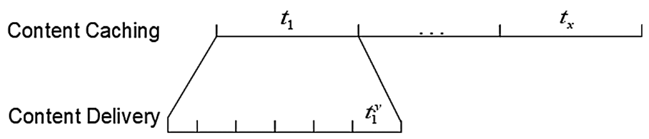

Figure 2: Content caching and delivery time slot

We define caching matrix $\alpha(t_x) \in \{0,1\}$ to represent the content cache state on caching nodes at t_x time slot. $\alpha(t_x)[e][f]=1$ denotes content f is cached on ACV_e, $\alpha(t_x)[k][f]=1$ denotes content f is cached on RSU_k. Define the delivery matrix $\beta\left(t_x^y\right) \in \{0,1\}$, which represents the state of the caching nodes processing MRV content requests at t_x^y time slot. $\beta\left(t_x^y\right)[m] = e$ and $\beta\left(t_x^y\right)[m] = k$ denotes at t_x^y time slot, the content requested by MRV_m is delivered by ACV_e and RSU_k, respectively. Therefore, at t_x and t_x^y time slot, making content caching and delivery decisions is equivalent to update $\alpha(t_x)$ and $\beta\left(t_x^y\right)$, respectively.

To improve content delivery efficiency, we also propose a RSU collaborative content delivery model. Specifically, RSUs are linked by optical fiber links, they share the cached content with each other and perform content delivery process in a collaborative manner. For example, as shown in Fig. 1, RSU_4 cannot perform the content delivery process because the content f requested by MRV is not cached. Based on the proposed collaborative model, when RSU_2 has cached content f, then f can be forwarded from RSU_2 to RSU_4. At last, f is delivered to MRV by RSU_4. By using the collaborative caching model, although additional content forwarding costs will be incurred, the number of times that MRV obtains content through backhaul link will be greatly reduced, which can effectively reduce content delivery energy consumption.

In summary, based on our proposed RSU collaborative content delivery model, there are four content delivery models for MRV content requests:

1. RSU direct delivery: as shown in step 1 of Fig. 1, the directly connected RSU_1 has cached the requested content f, RSU_1 directly delivers content f to MRV;
2. RSU indirect delivery: as shown in steps 2–3 of Fig. 1, the directly connected RSU_2 has not cached content f, while RSU_3 caches content f. At this time slot, RSU_2 can get content by forwarding content f from RSU_3, and then, deliver the content f to MRV;

3. ACV assisted delivery: as shown in steps 4 of Fig. 1, in the communication range of MRV, ACV has cached content f, ACV assisted delivers content f to MRV;
4. CCS delivery: all caching nodes (RSU and ACV) cannot delivery content f to MRV, the content f is delivered by CCV to MRV.

3.3 Content Popularity Model

Most of the current work assumes that the popularity of content follows the Zipf distribution in mobile social networks. Since the MRV may pass through multiple RSU regions during the content delivery process, this assumption may not always be applicable in IoV network, and distribution information of content popularity cannot correctly reflect the real-time content requirements of vehicles.

To provide a reasonable content popularity model, we define a global content popularity based on content request probability. RSU collects the content request information of all MRV and ACV in the region, calculates and updates the content popularity at the beginning of t_x time slot. Let p_f denote the popularity of content f, s_f denote the size of content f, d_f denote the maximum acceptable delay to obtain content f. The popularity of content f at t_x time slot is:

$$p_f(t_x) = \frac{\sum_{m=1}^{M} q_{m,f}^{x}}{M}, \quad f \in \mathcal{F} \tag{1}$$

where $q_{m,f}^{x} \in \{0,1\}$ denotes MRV_m requested state for content f at t_x time slot. $q_{m,f}^{x} = 1$ denotes MRV_m request content f at t_x time slot. According to content popularity p_f, MBS updates $\alpha(t_x)$ and makes content caching decisions. And then, a caching request is sent to CCS, the requested content will be download and cached on RSU or ACV.

3.4 Communication Model

In this section, we use the communication connection state between caching nodes (RSU, ACV) and MRV, as well as the communication connection state between cache nodes RSU_k and $RSU_{k'}$ to express the path that cache nodes obtain content.

Let $l_{m,i,f}\left(t_x^y\right), \quad i \in \mathcal{E} \cup \mathcal{K}$ represent connection state between MRV_m and caching node i at t_x^y time slot, $l_{m,i,f}\left(t_x^y\right) = 1$ represent MRV_m get content f from caching node i. Let $h_{m,k,k',f}\left(t_x^y\right)$ represent the connection state between caching node RSU_k and $RSU_{k'}$ at t_x^y time slot. $h_{m,k,k',f}\left(t_x^y\right) = 1$ and $k,k' \in \mathcal{K}$ represent connected, that is a collaborative delivery content process. $h_{m,k,k',f}\left(t_x^y\right) = 0$ and $k,k' \in \mathcal{K}$ represent unconnected, that is delivery content without collaborative process.

3.5 Channel Transmission Model

In this section, based on the real mobile network scenario, we analyze the link state between MRV and caching nodes and build an energy consumption model.

The signal-to-noise ratio (SNR) between MRV_m and caching node i at t_x^y time slot can be formulated as:

$$SNR_{m,i}\left(t_x^y\right) = \frac{p_{m,i}\left(t_x^y\right) g_{m,i}\left(t_x^y\right)}{\xi_{m,i}\left(t_x^y\right) d_{m,i}\left(t_x^y\right)^k \sigma_{m,i}\left(t_x^y\right)^2}, \quad m \in M \cup \mathcal{E}, \quad i \in \mathcal{E} \cup \mathcal{K} \tag{2}$$

where $p_{m,i}\left(t_x^y\right)$ and $d_{m,i}\left(t_x^y\right)$ represent transmission power and distance between MRV_m and caching node i, respectively. $g_{m,i}\left(t_x^y\right)$ is the antenna gain, $\xi_{m,i}\left(t_x^y\right)$ and k are path loss at and path loss exponent, respectively. $\sigma_{m,i}\left(t_x^y\right)$ is the additive white Gaussian noise.

Let W_0^{mbs}, W_k^{mbs} and W_e^{mbs} represent the available bandwidth resources between CCS and MRV, RSU and ACV respectively. $w_{m,i}\left(t_x^y\right)\ m \in M, \quad i \in \mathcal{E} \cup \mathcal{K}$ represents the bandwidth resource allocated by caching node i to MRV_m at t_x^y time slot. $w_{k,k'}\left(t_x^y\right)k, \quad k' \in \mathcal{K}$ represents the bandwidth resources between RSU_k and $RSU_{k'}$ in collaborative delivery process.

According to Shannon's formula, the data transmission rate when MRV_m get content from caching node i can be formulated as:

$$r_{m,i,f}\left(t_x^y\right)=l_{m,i,f}\left(t_x^y\right)w_{m,i}\left(t_x^y\right)log_2\left(1+SNR_{m,i}\left(t_x^y\right)\right), \quad m \in \mathcal{M}, \quad i \in \mathcal{E} \cup \mathcal{K}, \quad f \in \mathcal{F} \tag{3}$$

4 Problem Formulation

In this section, we transform the joint optimization problem of content caching and delivery into MDP. The behavior of MRV request content is modeled as Markov chain, in which each vehicle changes state with probability. The basic elements of characterizing MDP are: state set S, action set A, and reward R. The basic elements are defined below.

4.1 System State

RSU gets all MRVs content request information at t_x time slot, including MRV location $MRV_m(X_m,Y_m)$, content popularity p_f, content size s_f and maximum access delay d_f. Therefore, the system state includes the following parts:

$f(t_x^y)$: content requested by MRV at t_x^y time slot, $f(t_x^y) \in \mathcal{F}$.

$z(t_x^y)$: location of MRV and ACV at t_x^y time slot.

$p(t_x)$: content popularity at t_x time slot.

$o(t_x)$: content delivery deadline, $o(t_x) \leq d_f$

$\alpha(t_x)$: content cache state at t_x time slot.

The system state set $s_{t_x^y}$ can be formulated as:

$$s_{t_x^y}=\left\{\left[f\left(t_x^y\right)\right],\left[z\left(t_x^y\right)\right],[p(t_x)],[o(t_x)],[\alpha(t_x)]\right\} \tag{4}$$

4.2 System Action

At content caching time slot, after receiving the content request from MRV, MBS calculates content popularity, and then decides which content to cache in caching node. At content delivery time slot, MBS selects the best content delivery path to reduce the content delivery cost. Therefore, the action space contains the following parts:

$\alpha(t_x)$: content cache action at t_x time slot

$l(t_x^y)$: MRV get content f from caching node

$h(t_x^y)$: RSU collaborative action with RSU' when delivering content

The system action space $a_{t_x^y}$ can be formulated as:

$$a_{t_x^y}=\left\{[\alpha(t_x)],\left[l\left(t_x^y\right)\right],\left[h\left(t_x^y\right)\right],\right\} \tag{5}$$

4.3 Content Caching and Delivery Energy Consumption

4.3.1 Content Caching Energy Consumption

On the basis of the previous section, we can calculate that the delay of RSU_k get content f from CCS at t_x time slot is:

$$t_{k,0,f}\left(t_x^y\right)=\frac{s_f}{r_k},\quad f\in\mathcal{F},\quad k\in\mathcal{K} \tag{6}$$

the delay of MCE_e get content f from CCS is:

$$t_{e,0,f}\left(t_x^y\right)=\frac{S_f}{r_e},\quad f\in\mathcal{F},\quad e\in\mathcal{E} \tag{7}$$

The updated content size on caching node i at t_x time slot can be expressed as:

$$\Delta S_i(t_x)=\sum_{f=1}^{F}\left(s_f-\xi\{\alpha_i(t_x)[i][f]\;\&\;\alpha_i(t_{x-1})[i][f]\}\right),\quad i\in\mathcal{E}\cup\mathcal{K} \tag{8}$$

where ξ is the same content size on the caching node at t_x and t_{x-1} time slot, s_f is the content size, r is the data transmission rate.

The content caching cost is the energy consumption cost when transmitting updated content from backhaul link. It can be formulated as the product of transmission time and transmission power.

Therefore, according to Eqs. (6)–(8), the energy consumption of caching node RSU_k and ACV_e are as follows:

$$c_1(t_x)=\lambda_1\frac{\sum_{f=1}^{F}\left(s_f-\xi\{\alpha_k(t_x)[k][f]\;\&\;\alpha_k(t_{x-1})[k][f]\}\right)}{r_i}p_k \tag{9}$$

$$c_2(t_x)=\lambda_2\frac{\sum_{f=1}^{F}\left(s_f-\xi\{\alpha_e(t_x)[e][f]\;\&\;\alpha_e(t_{x-1})[e][f]\}\right)}{r_{m,i,f}\left(t_x^y\right)}p_e \tag{10}$$

where p_k is the transmission power between RSU_k and CCS, p_e is the transmission power between ACV_e and CCS, r is the data transmission rate, λ_1,λ_2 represent the weight factors on different cost.

4.3.2 Content Delivery Energy Consumption

On the basis of the previous section, we can calculate that the delay of MRV_m get content f from caching node i at t_x time slot is:

$$t_{m,i,f}\left(t_x^y\right)=\frac{s_f}{r_{m,i,f}\left(t_x^y\right)} \tag{11}$$

If the content is delivered by collaborative forwarding, the delay of forwarding content f is:

$$t_{k,k',m,f}\left(t_x^y\right)=\frac{s_f}{r_{k,k'}\left(t_x^y\right)} \tag{12}$$

where s_f is the content size, r is the data transmission rate.

Therefore, according to Eqs. (11)–(12), the total energy consumption of content delivery process at t_x^y time slot is:

$$c_3\left(t_x^y\right)=\lambda_3\sum_{i=1}^{E+K}\frac{s_f}{r_{m,i,f}\left(t_x^y\right)}p_{m,i}+\lambda_4\sum_{k=1}^{K}\frac{s_f}{r_{k,k'}\left(t_xy\right)}p_{k,k'} \tag{13}$$

where $p_{m,i}$ is the transmission power between MRV_m and caching node i, $p_{k,k'}$ is the transmission power between RSU_k and $RSU_{k'}$, λ_3,λ_4 represent the weight factors on different cost functions.

4.3.3 Penalty Cost

If the whole content is not obtained before the delivery deadline, the penalty cost will be:

$$\begin{aligned} & c_4\left(t_x^y\right)=\lambda_5 p_f\left(t_x^y\right)d_f \\ & s.t.\ t_{m,i,f}\left(t_x^y\right)+t_{k,k',m,f}\left(t_x^y\right)>d_f \end{aligned} \tag{14}$$

where p_f is content popularity, d_f is the maximum access delay, λ_5 represent the weight factors.

4.3.4 Cost Function

According to Eqs. (6)–(14), the total system cost function at t_x^y time slot can be formulated as:

$$\begin{aligned} c(t_x^y) &= \sum_{k\in K}c_1(t_x)+\sum_{e\in E}c_2(t_x^y)+\sum_{m\in M}c_3(tx^y)+c_4(tx^y) \\ &= \lambda_1\sum_{k=1}^{K}\sum_{f=1}^{F}(s_f-\xi\{\alpha_k(t_x)[k][f]\ \&\ \alpha_k(t_{x-1})[k][f]\})\frac{p_k}{r_i} \\ &+ \lambda_2\sum_{e=1}^{E}\sum_{f=1}^{F}(s_f-\xi\{\alpha_e(t_x)[\mathrm{e}][f]\ \&\ \alpha_e(t_{x-1})[\mathrm{e}][f]\})\frac{p_e}{r_{m,i,f}\left(t_x^y\right)}+ \\ & \lambda_3\sum_{m=1}^{M}\sum_{i=1}^{E+K}\frac{s_f}{r_{m,i,f}\left(t_x^y\right)}p_{m,i}+\lambda_4\sum_{m=1}^{M}\sum_{k=1}^{K}\frac{s_f}{r_{k,k'}\left(t_x^y\right)}p_{k,k'}+\lambda_5 p_f\left(t_x^y\right)d_f \end{aligned} \tag{15}$$

where $\lambda_1-\lambda_5$ represent the weight factors on different cost functions.

4.3.5 Reward

After taking action $a_{t_x^y}$, the system will get rewarded $r_{t_x^y}$. In the DRL method, actions affect not only the immediate reward, but also the next situation and all subsequent rewards. In order to get more rewards, DRL agent must like the actions that have been tried in the past and are considered to produce rewards effectively. This feature needs to be reflected in the total system energy consumption cost. The purpose of our problem is to minimize the energy consumption cost, the reward can be formulated as

$$r_{t_x^y}=-c(t_x^y) \tag{16}$$

5 MVECC Caching Scheme

5.1 Principle of DDPG Algorithm

For some actions, it may be a continuous value, or a very high-dimensional discrete value, so that the spatial dimension of the action is great. If the random policy is used, it is possible to study the probability of all possible actions like DQN, and calculate the value of each possible action, the sample amount of that requires is very large. Although at the same state, the transition probability is different, but the maximum probability is unique. Determinative strategies take only the maximum probability, the strategy is: $\pi_\theta(s)=a$.

Therefore, based on the deterministic Actor-Critic model, DDPG provide an accurate estimate decisive policy function and value function. Similar to the Actor-Critic model, the actor network uses policy functions, responsible for generating action and interacts with the environment. The critic network uses value function, responsible for assessing the actor's performance and guides the actor's next action. This combination can be used to implement joint optimization of the proposed content caching and delivery issues. The schematic diagram of DDPG is shown in Fig. 3. There are three modules: 1) actor network; 2) critic network; 3) experience replay buffer.

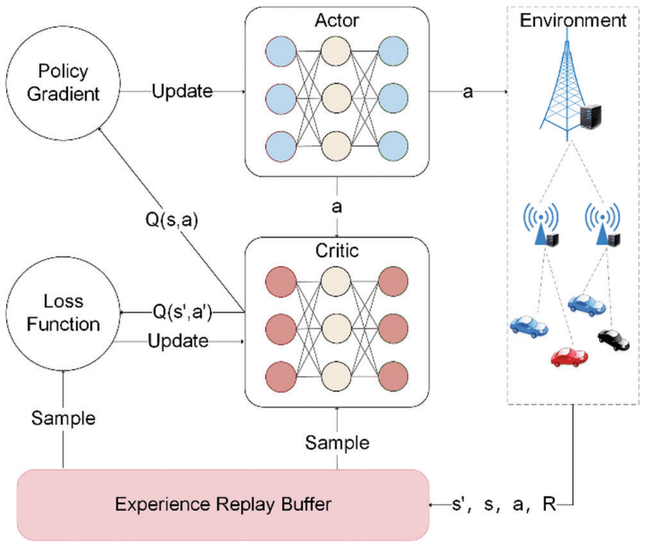

Figure 3: Algorithm flow chart

5.2 DDPG Based Mobile Edge Collaborative Caching Algorithm

We develop DDPG based solutions to jointly optimize caching and delivery strategies and minimize energy consumption costs, the algorithm is shown in Tab. 2. Network parameters are initialized at the beginning of the training process. The training dataset is collected by MBS. Then, vehicle information is calculated according to different models, and the parameters are input into the training network in the form of state and action matrix. Finally, after continuous training, the model will converge, and the output data is the optimal cache and delivery decision.

Table 2: DDPG based mobile edge collaborative caching algorithm

```
Algorithm 1: DDPG Based Mobile Edge Collaborative Caching Algorithm
1: Input: Actor-current network, Actor-target network, Critic-current network, Critic-target network
2: Parameters: θ, θ′, w, w′, attenuation factor γ, soft renewal coefficient τ, X, Y, minimum batch size D
3: Output: Actor-current network parameters θ, Critical-current network parameter w, action $a$
4: Initialize θ, w, w′ = w, θ′ = θ. Clear experience replay buffer
5: For episode = 1, 2, …, X do
6:   Initializethe first state of the current state sequence, $s(t_x^y)$ is the eigenvector.
7:   For episodes = 1, 2,…,Y do
8:     Gets action $a(t_x^y)=\pi_\theta(s(t_x^y))+N$ based on state $s(t_x^y)$ in Actor-current network
```

Table 2 (continued).

```
Algorithm 1: DDPG Based Mobile Edge Collaborative Caching Algorithm
9:    Execute action $a(t_x^y)$, get new state $s(t_x^{y+1})$, reward R
10:   Put $\{s(t_x^y), a(t_x^y), s(t_x^{y+1}), c(t_x^y)\}$ into experience replay buffer
11:   Sampling D samples from the experience replay buffer and calculate the current target Q-value $y_i = R + \gamma Q'(S', A', w')$
12:   Update all parameters of Critic-current network w thr-ough gradient reverse propagation of neural network
13:   Update all parameters of Actor-current network θ thr-ough gradient reverse propagation of neural network
14:   Update Critic-target network and Actor-target network parameters
15:  end for
16: end for
```

5.3 Mobile-Aware Caching Replacement Scheme

RSU should selectively replace outdated content based on the mobility of MRV. For example, when MRV moves from RSU1 to RSU2, RSU1 still stores the content requested by MRV, which causes a waste of storage resources. Fig. 4 shows the detailed process of three caching replacement rounds. MRV1, MRV2, and MRV3 request content *a*, *b*, and *c* respectively. The table shows the caching status of RSU in each round.

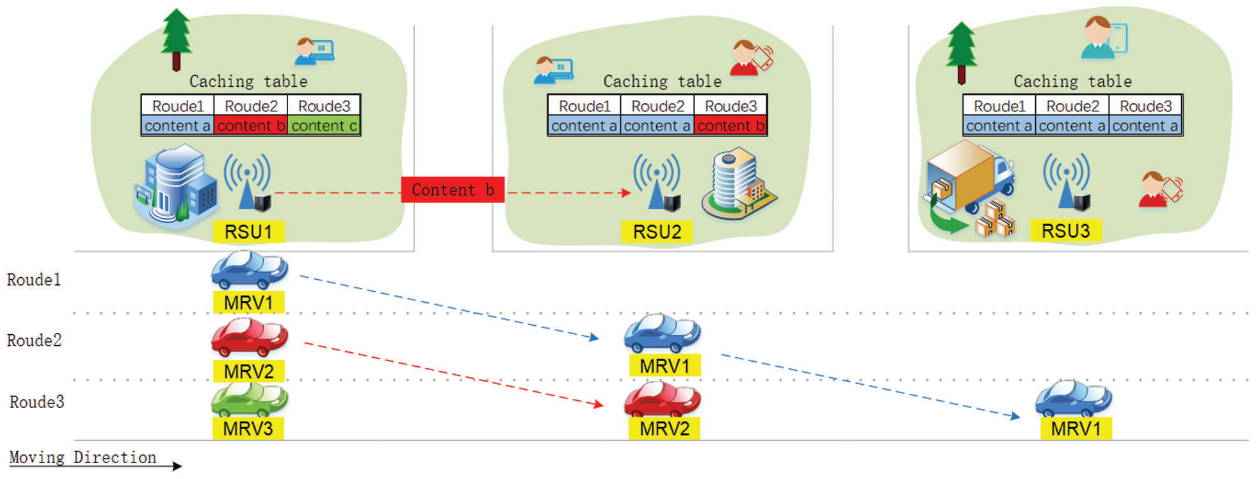

Figure 4: Caching replacement process

In the first caching round, MRV1 passes through the coverage area of RSU1 and requests content *a*. Assuming that RSU1 has cached content *a*, RSU1 directly delivers content *a* to MRV1.

In the second caching round, MRV1 leaves the coverage area of RSU1. At the same time, MRV2 will pass the coverage area of RSU1 and request content *b*. Content *a* that has been cached on RSU1 will not be delivered again. Therefore, in the second caching round, RSU1 replaces content *a* with content *b*, and RSU2 still caches content *a*.

At the beginning of the third caching round, the cache status of each RSU is: RSU1 cache content *b*, RSU1 cache content *a*, and RSU3 cache content *a*. MRV1 leaves the coverage area of RSU2, MRV2 will pass the coverage area of RSU2, MRV3 will pass the coverage area of RSU1, and request content *c*. At this time, the cached content *b* on RSU1 and the cached content *a* on RSU2 will no longer be delivered. Therefore, in the third caching round, RSU1 replaces content *c* with content *b*, RSU2 replaces content *b* with content *a*, RSU3 still caches content *a*. We found that during the first caching round, RSU1 has cached content *b*. Therefore, before RSU1 replaces content *c* with content *b*, RSU2 can first obtain content *b* from RSU1.

The flow of caching algorithm is shown in Fig. 5. In DDPG based collaborative caching algorithm, the input dataset is the state information of IoV environment, and the output dataset is content caching and delivery decisions. According to the decisions and node cache state, the cache replacement scheme determines the contents that need to be replaced.

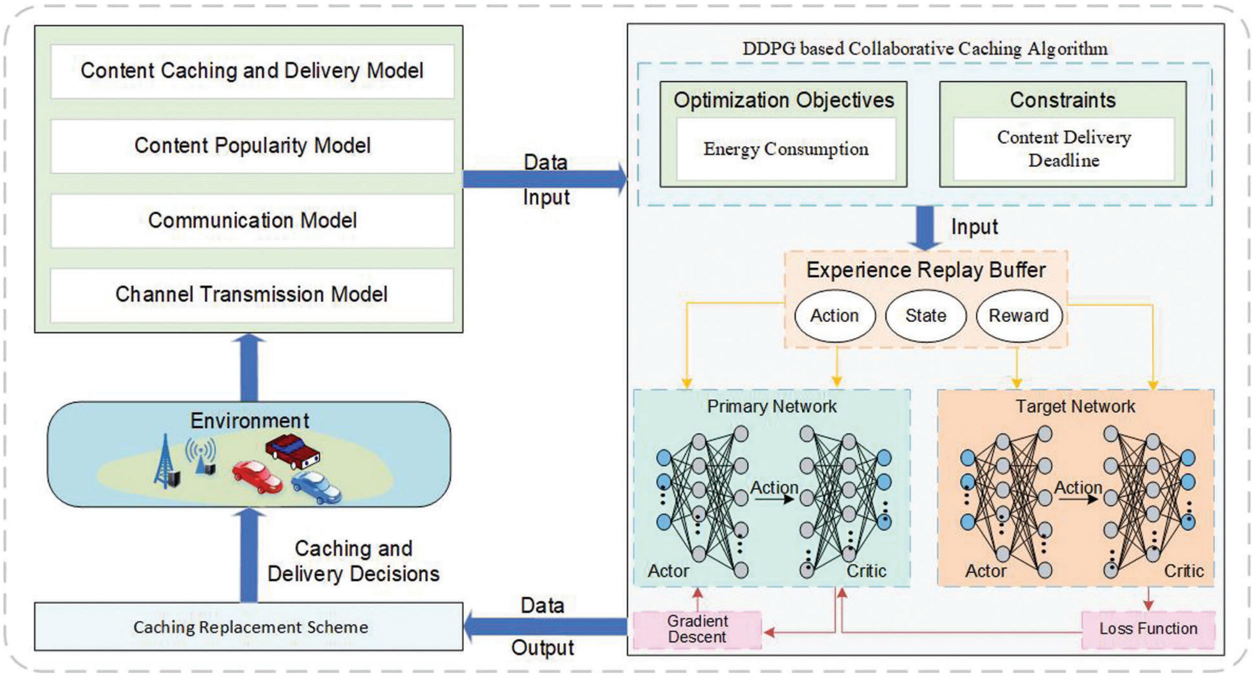

Figure 5: Flow chart of collaborative caching scheme

6 Simulation Results and Discussions

6.1 Parameter Settings

In this section, we use Python to build a simulation environment for the MVECC scheme. Using the TensorFlow platform to implement a DDPG-based collaborative caching solution (DDPG-Caching). Tab. 3 summarizes the main parameters used in the simulation.

As a reference, we implemented a random caching scheme and a deep Q-network (DQN)-based caching scheme (DQN-Caching) [24] as benchmark caching scheme. In the random caching scheme, the system randomly makes content caching and delivery decisions. In the DQN-caching scheme, there is no RSU collaborative caching, and only the RSU or ACV is responsible for delivery. We take energy consumption and content hit rate as performance measures. The energy consumption is generated by the content transmission, it can be calculated by Eqs. (9)–(13). We take the ratio of content hit number to the total requests number as content hit rate, which measures the caching efficiency of RSUs. The content hit rate can be formulated as:

$$ht_f = \frac{\sum_{m=1}^{M} a_{m,f}^{x}}{M}, \quad f \in \mathcal{F} \tag{17}$$

where $a_{m,f}^{x} \in \{0, 1\}$ denotes content hit state, M is the total requests number.

Table 3: Simulation parameters

System parameters	Value
Learning rate of actor-network	0.001
Learning rate of critic-network	0.002
Number of RSU	10
Number of ACV	[50, 200]
Number of MRV	[150, 350]
CCS transmit power	75 dBm
RSU transmit power	60 dBm
RSU cache capacity	[5–10 GBytes]
ACV transmit power	50 dBm
Bandwidth range	[10–30 MHz]
Content size	[300–500 MBytes]

6.2 System Performance Analysis

Fig. 6 shows the comparison of system energy consumption costs when MRV = 200 and ACV = 50. As the number of training increases, we can draw the following observations from Fig. 6. First of all, the random cache scheme has the highest average system energy cost, the energy cost does not decrease with the increase of the number of trainings. This is because the random content caching and delivery decisions are not optimal system decisions. Therefore, random cache has a higher energy cost and does not have convergence. Secondly, because the collaboration of RSU and the assisted caching of ACV provide a more effective caching and delivery mode, the MVECC scheme can effectively reduce system costs. Third, compared with other benchmark schemes, through the DDPG reinforcement learning algorithm, our proposed MVECC scheme successfully solves the complex problem of high-dimensional action space in the joint optimization. So that MVECC can quickly find the best caching and delivery solution.

Next, in order to verify the performance of our proposed collaborative caching scheme, we compared the DQN-Caching algorithm with the proposed DDPG-Caching algorithm in terms of content hit rate and energy consumption.

As shown in Fig. 7, compared with the non-collaborative algorithm, the proposed MVECC scheme increases the content hit rate by about 15%. As the number of ACVs increases, MVECC can provide MRV with more opportunities for V2V content delivery. In addition, through the caching and delivery decision made by DRL agent, RSU can cache more valuable content and deliver the content with a higher hit rate. Since DQN does not have strong learning ability when dealing with high-latitude spatial problems, the more ACVs there are, the higher the content hit rate of DRL based-MVECC will be.

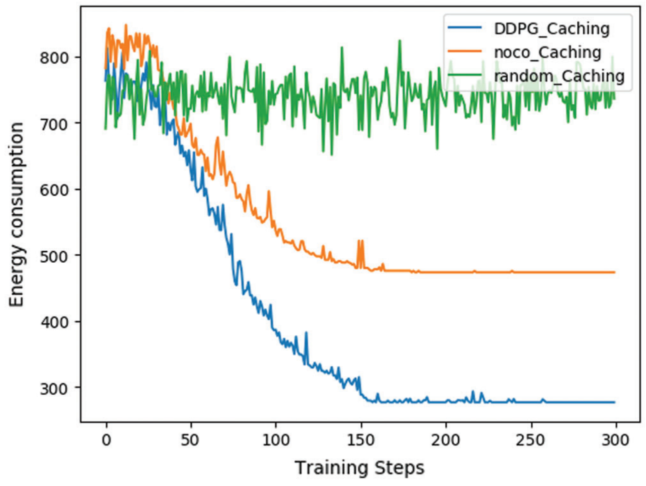

Figure 6: Comparison of energy consumption

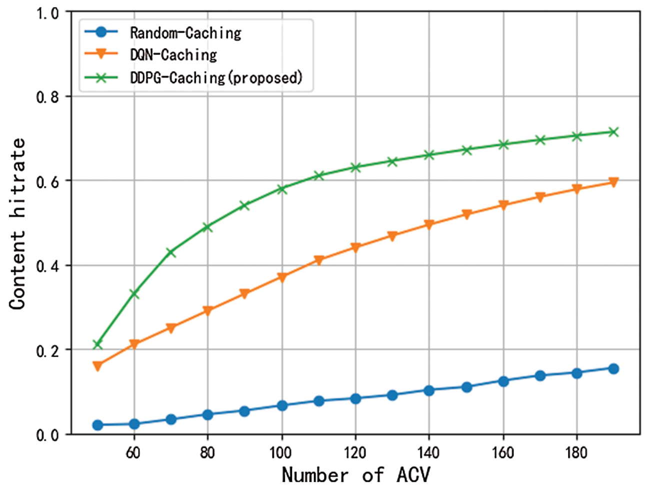

Figure 7: Content hit rate under different ACV numbers

Then, we use the overall system energy consumption as a performance indicator. As shown in Fig. 8, our proposed MVECC scheme has the lowest system energy consumption. The reason is that in the MVECC scheme, RSUs can respond to MRV content requests on the edge side, which can reduce system latency by reducing transmission distance and sharing cached content. When there are more MRV numbers, the number of requests will increase accordingly, so the system energy consumption will also increase. The increase in ACV will provide more V2V opportunities, so the system energy consumption will continue to decrease. Since there is no learning process for random caching, the content caching and delivery decisions executed each time are random rather than optimal, which results in a large system energy consumption. However, for DRL, when the number of MRV and ACV increases, there will be a disaster of dimensionality. Therefore, compared with the DQN caching algorithm, the DRL agent can avoid the dimensionality disaster through the DDPG algorithm and determine the best caching and delivery strategy for MVECC. Therefore, the optimal caching and delivery decisions made by the DRL agent in the MVECC scheme can still minimize the energy consumption.

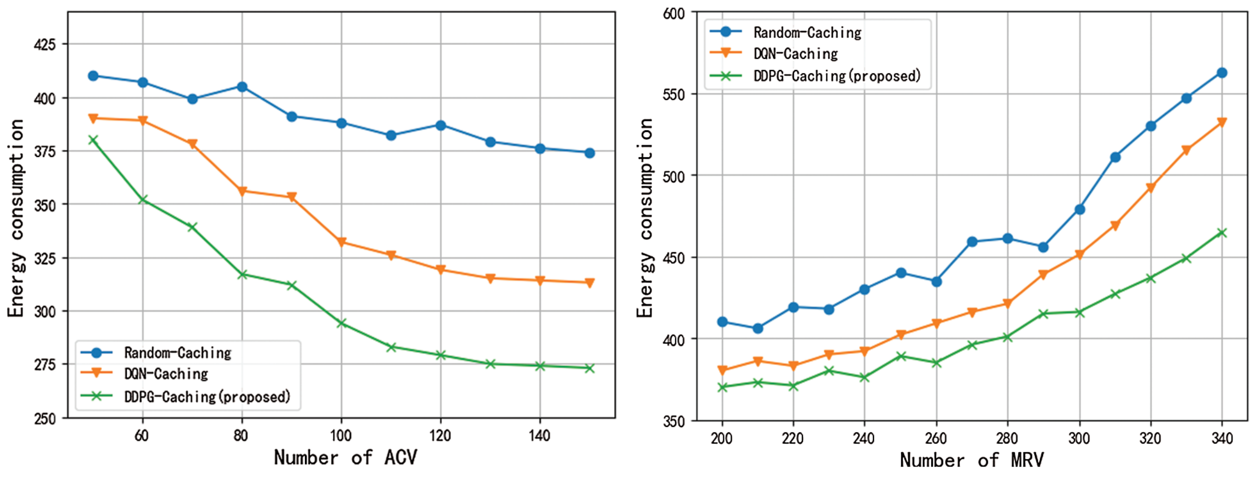

Figure 8: System energy consumption under different ACV and MRV numbers

Finally, we compared the system energy consumption under different cache capacities. As shown in Fig. 9, as the cache capacity increases, the system energy consumption continues to decrease. This is because more content can be cached on the edge side, MRV content requests will be responded to more on the edge side. The number of content requests forwarded by the link to the CCS will be greatly reduced, thereby effectively reducing the high energy consumption costs on the backhaul link. In addition, the DRL agent selects the optimal content caching and delivery decision for the MVECC scheme, so that the caching node can cache the content that is more likely to be delivered under the limited cache capacity. RSU and ACV have more opportunities for content delivery, so that the system has the lowest energy consumption.

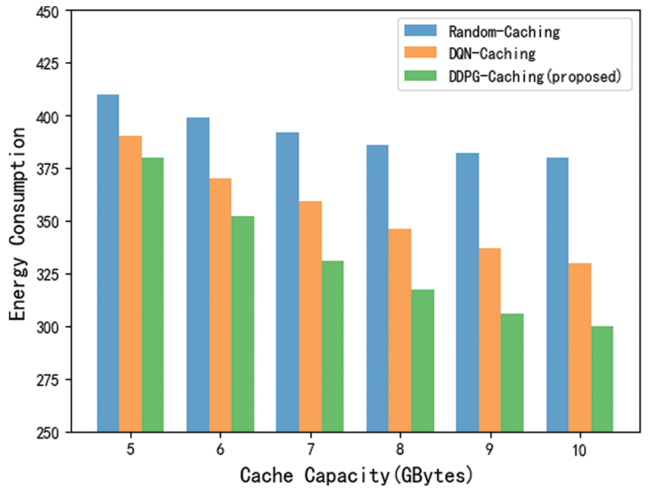

Figure 9: System energy consumption under different cache capacity

7 Conclusion

In this article, we take the vehicle caching network as an example to study the collaborative caching problem in the internet of vehicles. We propose a novel MEC-assisted vehicle edge collaborative caching (MVECC) scheme to improve cache efficiency. With the help of the roadside unit (RSU) and assisted caching vehicle, MVECC has two content caching modes and four content delivery modes coexist. MVECC also uses DRL to actively cache popular content on RSU, so as to reduce cache energy consumption. In addition, based on the vehicle mobility, we designed a mobility-aware cache replacement strategy to dynamically update the cached content on RSU. The simulation results show that the method is better than the random caching scheme and the DQN caching scheme in terms of content hit rate and system energy consumption. We do not consider the impact of link state changes on the transmission process, in our future work, we intend to study this problem and design a caching scheme based on federated learning to better adapt to user behavior.

Acknowledgement: Thanks for the technical support provided by the State Key Laboratory of Networking and Switching Technology.

Funding Statement: This work is supported by the Science and Technology Project of State Grid Corporation of China: Research and Application of Key Technologies in Virtual Operation of Information and Communication Resources.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Y. Kim, N. An, J. Park and H. Lim, “Mobility support for vehicular cloud radio-access-networks with edge computing,” in *2018 IEEE 7th Int. Conf. on Cloud Networking (CloudNet)* , Tokyo, pp. 1–4, 2018.

[2] Y. Yan, Y. Dai, Z. Zhou, W. Jiang and S. Guo, “Edge computing-based tasks offloading and block caching for mobile blockchain,” *Computers Materials & Continua*, vol. 62, no. 2, pp. 905–915, 2020.

[3] D. Zhu, Y. Sun, X. Li and R. Qu, “Massive files prefetching model based on LSTM neural network with cache transaction strategy,” *Computers Materials & Continua*, vol. 63, no. 2, pp. 979–993, 2020.

[4] M. Yu, R. Li and Y. Chen, “A cache replacement policy based on multi-factors for named data networking,” *Computers Materials & Continua*, vol. 65, no. 1, pp. 321–336, 2020.

[5] G. Qiao, S. Leng, S. Maharjan, Y. Zhang and N. Ansari, “Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks,” *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 247–257, 2020.

[6] Z. Li, W. Wei, T. Zhang, M. Wang, S. Hou *et al.,* “Online multi-expert learning for visual tracking,” *IEEE Transactions on Image Processing*, vol. 29, pp. 934–946, 2020.

[7] Q. Deng, Z. Li, J. Chen, F. Zeng, H. Wang *et al.,* “Dynamic spectrum sharing for hybrid access in OFDMA-based cognitive femtocell networks,” *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10830–10840, 2018.

[8] M. A. Maddah-Ali and U. Niesen, “Fundamental limits of caching,” *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, 2014.

[9] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang *et al.,* “A survey on mobile edge networks: Convergence of computing, caching and communications,” *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[10] X. Wang, S. Leng and K. Yang, “Social-aware edge caching in fog radio access networks,” *IEEE Access*, vol. 5, pp. 8492–8501, 2017.

[11] L. Yao, A. Chen, J. Deng, J. Wang and G. Wu, “A cooperative caching scheme based on mobility prediction in vehicular content centric networks,” *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 5435–5444, 2018.

[12] H. Wang, Y. Li, X. Zhao and F. Yang, “An algorithm based on markov chain to improve edge cache hit ratio for blockchain-enabled IoT,” *China Communications*, vol. 17, no. 9, pp. 66–76, 2020.

[13] J. Huang, C. Zhang and J. Zhang, “A multi-queue approach of energy efficient task scheduling for sensor hubs,” *Chinese Journal of Electronics*, vol. 29, no. 2, pp. 242–247, 2020.

[14] J. Huang, J. Liang and S. Ali, “A simulation-based optimization approach for reliability-aware service composition in edge computing,” *IEEE Access*, vol. 8, pp. 50355–50366, 2020.

[15] C. Campolo and A. Molinaro, “Improving V2R connectivity to provide ITS applications in IEEE 802.11p/WAVE VANETs,” in *2011 18th Int. Conf. on Telecommunications*, Ayia Napa, pp. 476–481, 2011.

[16] J. Kwon and H. Park, “Reliable data dissemination strategy based on systematic network coding in V2I networks,” in *2019 Int. Conf. on Information and Communication Technology Convergence (ICTC)*, Jeju Island, Korea (South), pp. 744–746, 2019.

[17] J. Mei, K. Zheng, L. Zhao, Y. Teng and X. Wang, “A latency and reliability guaranteed resource allocation scheme for LTE V2V communication systems,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 3850–3860, 2018.

[18] W. Zhao, Y. Qin, D. Gao, C. Foh and H. Chao, “An efficient cache strategy in information centric networking vehicle-to-vehicle scenario,” *IEEE Access*, vol. 5, pp. 12657–12667, 2017.

[19] R. Ding, T. Wang, L. Song, Z. Han and J. Wu, “Roadside-unit caching in vehicular ad hoc networks for efficient popular content delivery,” in *2015 IEEE Wireless Communications and Networking Conf. (WCNC)*, New Orleans, LA, USA, pp. 1207–1212, 2015.

[20] S. Abdelhamid, H. S. Hassanein and G. Takahara, “On-road caching assistance for ubiquitous vehicle-based information service,” *IEEE Transactions on Vehicular Technology*, vol. 64, no. 12, pp. 5477–5492, 2015.

[21] B. Hu, L. Fang, X. Cheng and L. Yang, “Vehicle-to-vehicle distributed storage in vehicular networks,” in *2018 IEEE Int. Conf. on Communications (ICC)*, Kansas City, MO, USA, pp. 1–6, 2018.

[22] Z. Li, T. Chu, I. Kolmanovsky, Y. Xiang and X. Yin, “Cloud resource allocation for cloud-based automotive applications,” *Mechatronics*, vol. 50, no. 4, pp. 356–365, 2018.

[23] A. Sadeghi, F. Sheikholeslam and G. B. Giannakis, “Optimal and scalable caching for 5G using reinforcement learning of space-time popularities,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 180–190, 2018.

[24] J. Tang, H. Tang, X. Zhang, K. Cumanan, G. Chen *et al.,* “Energy minimization in D2D-assisted cache-enabled internet of things: A deep reinforcement learning approach,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5412–5423, 2020.