Zhang *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2020) 9:33
https://doi.org/10.1186/s13677-020-00182-x

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH **Open Access**

A joint optimization scheme of content caching and resource allocation for internet of vehicles in mobile edge computing

Mu Zhang[1], Song Wang[2] and Qing Gao[3*]

Abstract

In a high-speed free-flow scenario, a joint optimization scheme for content caching and resource allocation is proposed based on mobile edge computing in Internet of Vehicles. Vehicle trajectory prediction provides the basis for the realization of vehicle-cloud collaborative cache. By pre-caching the business data of requesting vehicles to edge cloud networks and oncoming vehicles, requesting vehicles can obtain data through V2V link and V2I link at the same time, which reduces the data acquisition delay. Therefore, this paper considers the situation where bandwidth of V2I and V2V link and the total amount of edge cloud caches are limited. Then, the bandwidth and cache joint allocation strategy to minimize the weighted average delay of data acquisition is studied. An edge cooperative cache algorithm based on deep deterministic policy gradient is further developed. Different from Q-learning and deep reinforcement learning algorithms, the proposed cache algorithm can be well applied to variable continuous bandwidth allocation action space. Besides, it effectively improves the convergence speed by using interactive iteration of value function and strategy function. Finally, the simulation results of vehicle driving path at the start and stop are obtained by analyzing real traffic data. Simulation results show that the proposed scheme can achieve better performance than several other newer cooperative cache schemes.

Keywords: Mobile edge computing, Cloud computing, Collaborative caching, Internet of vehicles, Resource allocation, Deep deterministic policy gradient, Convergence speed

Introduction

In high-speed free-flow scenarios, vehicles can travel freely at any speed. The vehicle may initiate some business requests to remote cloud networks for obtaining certain data during driving. If all such data is transmitted to the vehicle user by remote cloud networks, the vehicle user will experience a large delay. Due to vehicles switch between edge clouds frequently, vehicles need to constantly re-initiate requests to the cloud for obtaining the remaining data [1–3]. In addition, a single edge cloud-assisted caching strategy may cause vehicles to fail to obtain complete data in high-speed free-flow scenarios. Thus, vehicle cloud networks and edge clouds often need to provide caching services in a collaborative manner [4, 5].

Based on this, this paper proposes a vehicle-cloud collaborative caching strategy. That is, the edge cloud network and oncoming vehicles cache the business data for requesting vehicles and transmit it together when interacting with them. By accurately predicting the trajectories of vehicles, it is possible to determine at which moment the oncoming vehicles can meet. In this way, the collaborative edge cloud of oncoming vehicles can be utilized for collaborative caching. When a vehicle is requested to travel to a certain location, data can be obtained from the edge cloud by V2I (Vehicle to Infrastructure) link, and data can be obtained from the oncoming vehicle by V2V (vehicle to vehicle

* Correspondence: gaoqing@uestc.edu.cn
[3]School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu 611731, Sichuan, China
Full list of author information is available at the end of the article

communication) link. In the end, the vehicle can obtain required data with lower latency, which improves the QoS of cache service. However, the problems in the above scenarios are as follows: On the one hand, the bandwidth of wireless link is limited. How the limited bandwidth is allocated to each V2I link and V2V link determines the service delay of final vehicle users. On the other hand, different cache ratios of edge cloud and oncoming vehicles to data will also affect the transmission rate. For different vehicle speeds, location distributions, etc., the status of requesting vehicles to interact with edge cloud or oncoming vehicle is different. How to make effective cache adjustments affects service QoS closely. Third, the number of caches in edge cloud networks is limited. How to guarantee the QoS of services in a limited cache is also a challenge. Thus, this paper performs a modeling analysis on the above scenarios, and aims to minimize the weighted average delay of requesting vehicle users, which studies the edge cloud network bandwidth and cache allocation strategies. First, the paper establishes the mathematical model, and proposed an edge cooperative caching scheme based on deep deterministic policy gradient (DDPG). By integrating the value iteration and policy gradient methods, the convergence speed of this algorithm is accelerated. Compared with the existing two cache strategies, the proposed cache strategy can not only improve the content hit rate, but also reduce system overhead and content transmission delay. Finally, the performance of the proposed strategy is verified by simulation.

Related works

Vehicle cloud networks based on mobile edge computing can effectively use vehicle resources to provide low-latency cloud services and improve the dynamic accessibility of resources. However, due to the limited scope and scale of vehicle cloud network, how to efficiently manage and allocate resources has become a major difficulty for the vehicle cloud network. There are three types of resources in the Internet of Vehicles: communication resources, computing resources and storage resources. Communication resources are frequency resources, time domain resources and airspace resources required for V2V and V2I communication. Computational resources are the processor resources required to perform complex operations and storage resources are the storage space required for data storage. Many studies have provided their own solutions to the problem of vehicle cloud network resource allocation [6–15]. Reference [6] defined vehicle cloud network as a paradigm shift of traditional vehicle self-organizing network. Since vehicles not only use their on-board resources in the vehicle cloud network, they also use cloud resources. Reference [7] first proposed a layered vehicle cloud network and analyzed feasible application scenarios. Then, for the above network architecture they studied how computing and storage resources are shared in different applications. And they proposed a virtual machine migration strategy based on game theory, which can achieve higher resource utilization by virtual resource migration. Reference [8] proposed a distributed and adaptive resource management method, which achieved the optimal utilization of cognitive radio and soft input / soft output data fusion in vehicle access networks. In such a way, smartphones in cars with limited energy and computing power can improve performance by uploading their computing tasks to the local or remote cloud. Reference [9] proposed a framework based on cloud computing, which allows vehicles to switch between a Central Processing Unit (CPU) and a Graphics Processing Unit (GPU) in real-time when performing video processing. Reference [10] studied the joint allocation strategy of computing resources and communication resources (slots) in edge cloud computing task offloading based on non-orthogonal multiple access (NOMA). Reference [11] studied the problem of computing task offloading for cognitive vehicle cloud networks. The object considered in reference [12] was an unpredictable available computing resource, and a linear programming-based scheduling model was proposed. Besides, the model's input was a task with processing time. However, it assumed that tasks cannot be processed in parallel. That is, a task is randomly assigned to only one vehicle at a time. Reference [13] considered the dynamic nature of vehicles and the change in the amount of resources will result when vehicles enters or leaves vehicle cloud networks. In this case, they presented a computing resource offloading strategy based on the Semi-Markov decision process (SMDP). Reference [14] considered a cross-domain resource allocation strategy. That is, calculation requests that cannot be completed in the current cloud domain will be offloaded to other cloud domains for completion, and an optimal cross-domain resource allocation algorithm was given. Reference [15] proposed an adaptive computing resource allocation algorithm for real-time vehicle cloud network services with the goal of minimizing energy consumption.

For the collaborative cache service in Internet of Vehicles, some research work has proposed their own solutions [16–19]. Reference [16] investigated the cooperative caching strategies based on MANETs, and pointed out which ones can be applied to Internet of Vehicles. Reference [17] gave a cluster-based Internet of Vehicles content distribution strategy. They used the cache resources of multiple vehicles in the vehicle cloud network for collaborative caching, thereby reducing the request loss and cache loss of backhaul link. This work only considered cache requests in the vehicle cloud network, whose request content is greatly restricted by the surrounding vehicles. Reference [18] proposed a peer-to-peer (P2P) direct content distribution strategy between vehicles. Vehicles can share the resources required by each other when they meet each other. When there are no required resources (that is, cache misses), the vehicle can also obtain data through the edge cloud. They proposed cache placement and content selection algorithms to reduce the loss caused by access to cache. However, the work did not take full advantage of edge cloud's advantages in caching services. Reference [19] considered the mobility of vehicles and proposed a cluster-based collaborative caching strategy. Based on the vehicle mobility, they clustered the vehicles. The node selection and content placement strategies of the intra-cluster and inter-cluster cooperative caches are given to minimize the transmission delay. Similarly, this work did not consider the role of edge clouds. From the above research status, it can be found that very few studies have been conducted on collaborative caching strategies for vehicle cloud networks and edge cloud networks.

System model

Scenario description

Assume that in a high-speed free-flow scenario, N vehicles pass through a two-way two-car road segment, as shown in Fig. 1. The vehicle edge cache scenario studied in this paper includes a Macro-cell Base Station (MBS), several Roadside Units (RSUs) and Content Caching Vehicles (CCVs). These vehicles initiate a request for K, $K \leq N$ type service data, and mark V as a collection of requested vehicles. Since the edge cloud can obtain the trajectories of vehicles in advance, it can cache the business data in advance. When a vehicle passes its coverage, data is obtained through V2I. At the same time, K vehicles in oncoming vehicles on the road will also cache some business data, which can improve the efficiency of data acquisition. The set of oncoming vehicles is represented by the symbol K. In this case, when vehicles in the two directions meet, the opposite vehicle can transmit buffered data to the requesting vehicle through V2V communication method, thereby realizing the collaborative cache between vehicles and edge cloud. Assume that each vehicle runs at a constant speed on the road, and let v_n be the speed of the n th vehicle. The maximum coverage of the base station in edge cloud is R_I, the length of road in coverage is L, and the longest communicable distance of V2V link is R_V.

Vehicle position model

The relative position models of vehicle and base station, vehicle and opposite vehicle are shown in Figs. 2 and 3 respectively, where Fig. 2 shows the relative positions of vehicles and base stations. h represents the vertical distance from the base station to the road, h_t and h_t represent the antenna heights of base stations and vehicles respectively. Then there are:

$$R_I^2 = h^2 + \frac{L^2}{4} + (h_t - h_v)^2 \quad (1)$$

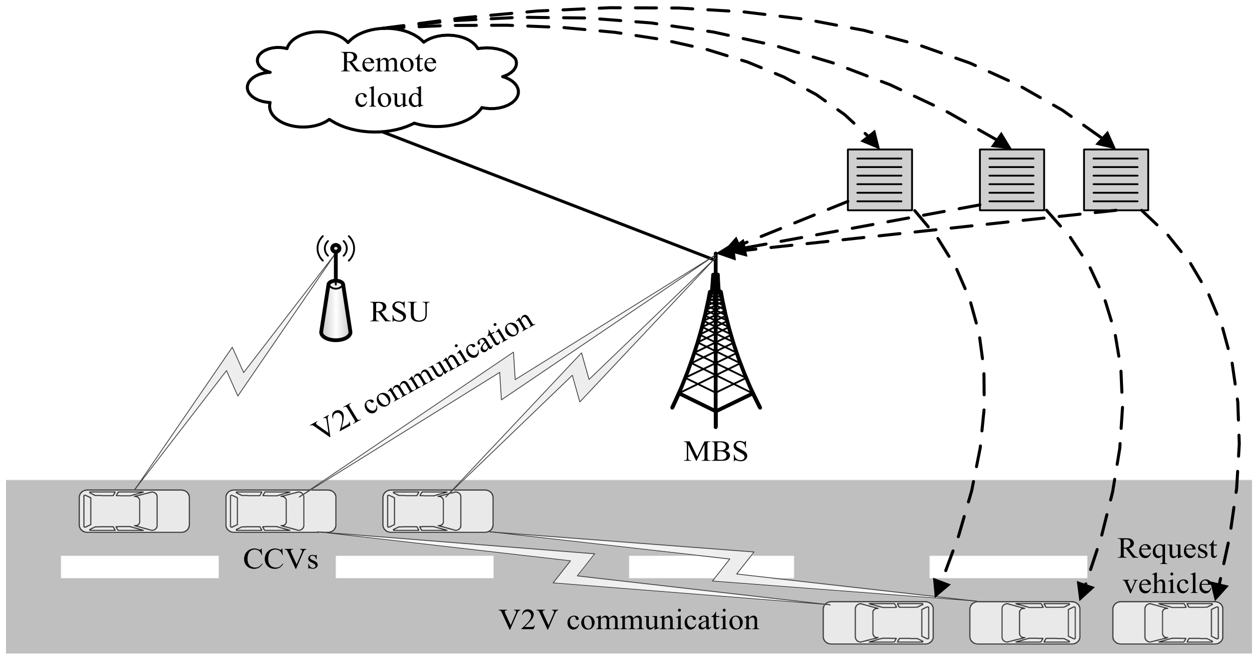

Fig. 1 Schematic diagram of the vehicle-cloud collaboration cache scenario

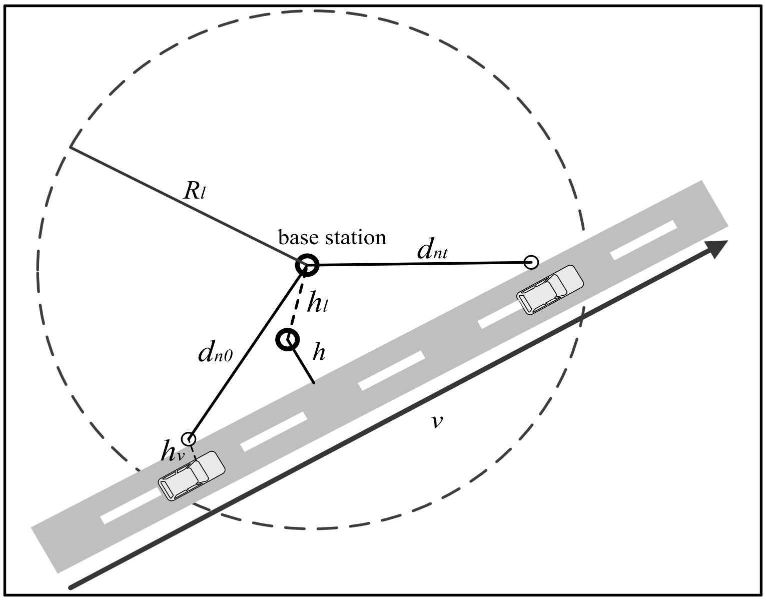

Fig. 2 The relative position model of vehicles and base stations

Let vehicle n be d_{nt} from the base station at time t. The expression that can be calculated for d_{nt} is:

$$d_{nt} = \sqrt{\left(\frac{L}{2} - v_n t\right)^2 + h^2 + (h_t - h_v)^2} \tag{2}$$

Figure 3 shows the location model of vehicles and the oncoming vehicles. Suppose the lane spacing is l. Take the position where base station is perpendicular to the road as the 0 point, and the opposite direction as x-axis direction. Since there are single lanes on both sides, the vehicle position can correspond to coordinate points on several axes. If the coordinates of starting positions for two vehicles are x_{n0} and x_{k0}, and the speeds are v_n and v_k, then the distance between vehicle n and vehicle k at time t is

$$d_{nkt} = \sqrt{[(x_{n0} - x_{k0}) - (v_n + v_k)t]^2 + l^2} \tag{3}$$

Communication model

In this scenario, there are two types of communication methods, V2I communication and V2V communication. It is assumed that the V2I communication and the V2V communication link occupy different frequency bands, and there is no interference between them. Assume that the system allocates a total of B_m and b_m bandwidths for the V2I link and the V2V link respectively. The edge cloud needs to allocate V2I and V2V bandwidth for transmission of each vehicle. It is assumed that the smallest assignable units of bandwidth are B_0 and b_0 respectively, and the noise power is σ^2. Besides, the bandwidth allocation of V2I and V2V links is performed by the edge cloud. The transmission rate formulas for V2I and V2V links are given below.

V2I communication link Base stations need to transmit the pre-buffered data to requesting vehicles through V2I communication link. The number of assignable bandwidth units is $M_B = B_m/B_0$. Assume that the number of bandwidth units allocated by edge cloud to the n th vehicle is M_n, $M_n \leq M_B$. The transmission rate of its V2I communication link is expressed as:

$$C_n = (M_n B_0) \log(1 + \gamma_n) \tag{4}$$

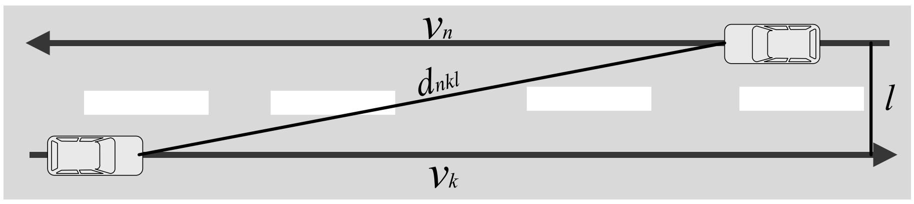

Fig. 3 The relative position model of vehicles and oncoming vehicles

where γ_n is the received signal-to-noise ratio of V2I link and is given by:

$$\gamma_n = \frac{PGG_nh^2h_v^2PL}{\sigma^2} \tag{5}$$

where P is the transmission power of base stations, G is the reception gain of base station antenna, G_n is the reception gain of vehicle antenna, h and h_v are the height of base stations and vehicle antenna. σ^2 is the white Gaussian noise power and the path loss of V2I link is in dB Is defined as

$$PL = 128 + 37.6* \lg(d_{nt}) \tag{6}$$

V2V communication link The opposite vehicle needs to transmit the buffered data to requesting vehicles by V2V communication link. The number of V2V link assignable bandwidth units is $m_b = \frac{b_m}{b_0}$. For vehicle n, if it obtains data through vehicle k, and the number of V2V bandwidth allocated to link by edge cloud is m_n. The transmission rate of its V2V communication link is expressed as:

$$C_n = (M_nB_0)\ \log\left(1 + \gamma_n\right) \tag{7}$$

where γ_{nk} is the received signal-to-noise ratio and is given by:

$$\gamma_{nk} = \frac{p_kG_kG_nh_v^4(d_{nkt})^{-\varsigma}}{\sigma^2} \tag{8}$$

where p_k is the vehicle transmission power, G_k is the vehicle transmitter antenna gain, and ς is the V2V link path loss factor.

Cache model

It is assumed that each type of business data in the system can be divided into z_k parts and transmitted separately through appropriate file encoding technology. The z_k shards will be pre-cached to the edge cloud and an oncoming vehicle, respectively. Assume that the amount of data of the type k service is W_k. The amount of data of each shard is $W_{k0} = W_k/0$, the total cache resource of edge cloud is U, and the proportion of the edge cloud cache occupied by the type k service is η_k. Then the amount of data cached by edge cloud side is $[\frac{U\eta_k}{W_{k0}}]W_{k0}$. The amount of data buffered by oncoming vehicles is after the $[\frac{U(1-\eta_k)}{W_{k0}}]W_{k0}$ oncoming vehicle has cached certain business data in advance. Its business data cache cannot be increased. The edge cloud cache can dynamically adjust the cache amount of its business. In this way, three situations can happen to the amount of cache:

- Cache redundancy: The amount of cache on edge cloud side is higher than its initial cache level, so it overlaps with the oncoming vehicle cache to cause redundancy. Besides, the edge cloud can control requesting vehicles to obtain overlapping caches from opposite vehicles or edge cloud side by adjusting the bandwidth;
- Complete cache: At this time, the amount of edge cloud cache is the amount required by vehicles. And the vehicles can obtain data completely with low latency with sufficient bandwidth and time:
- Insufficient cache: The amount of cache on edge cloud side is lower than its initial cache amount. At this time, requesting vehicles cannot fully obtain all the data. It will have to get the remaining data through the remote cloud within next edge cloud coverage. This greatly increases data acquisition delay and reduces QoS.

Except for the buffer amount, the initial position and speed of vehicles will also affect the final transmission result. For example, when oncoming vehicles are fast, V2V link communication time is lower than the time required for data transmission. When corresponding cache data is not added on the edge cloud side, requesting vehicles cannot obtain complete data. Regardless of insufficient cache or insufficient transmission time, the remaining business data will have to be obtained by requesting vehicles to the remote cloud through the edge cloud of next segment. Since the remote cloud data transmission needs to go by the Internet, the delay of remaining business data transmission will greatly increase. The remaining service data needs to be returned to the next edge cloud by remote cloud: then sent to vehicles by V2I link. Assume that the channel quality of next segment is the same as current segment, and the backhaul link rate from remote cloud to edge cloud is fixed at C, with a fixed request delay τ_r, and the remaining data amount is $\hat{W}_k$:

$$T_n^{'} = \hat{W}_k\left(\frac{1}{\overline{C}_n^{'}} + \frac{1}{C}\right) + \tau_r \tag{9}$$

Here, $\overline{C}_n^{'}$ is the transmission rate of V2I link between vehicles and the next edge cloud.

Optimization goal

During the operation of system, the edge cloud will observe the system once every T time period, and take a decision to adjust bandwidth and the number of edge cloud caches. It is assumed that T period is sufficiently small. Within each time period, the channel quality will not change much. The channel quality at the initial time of T period is taken as the average channel quality in period. Assume that the number of time periods during which the vehicle is transmitting arbitrarily on the current road section is j_n, and the time spent by vehicles to obtain data is:

$$\tau_n = j_n T + \tau_n^{'} \tag{10}$$

On the basis of delay, in order to balance efficiency and fairness, this paper defines optimization goal as minimizing the weighted average transmission delay [20].

$$\min \sum_{n=1}^{N} \rho_n \tau_n \tag{11}$$

$$s.t. \sum_{n=1}^{N} M_n B_0 \leq B_m \tag{12}$$

$$\sum_{n=1}^{N} m_n b_0 \leq b_m \tag{13}$$

$$\sum_{n=1}^{N} \rho_n = 1 \tag{14}$$

Here, ρ_n represents the weighting factor of delay.

Model construction

The above problem is first modeled as a Markov decision process model. Its basic five-tuple is: ·.

$$M@\{\tau, S, A_s, R, \rho\} \tag{15}$$

The following details the model building process:

Moment of decision

Let the time at which any one of vehicles in the set N first obtain data as the starting time T_0. Edge cloud observes the state of system every time T passes, and then take a decision. Define the end time of last-vehicle data transmission as the end time of this phase, that is, T_s. This model is a phased model with a termination state. The set τ formed by the model's decision time can be expressed as:

$$\tau = \{T_0, T_0 + T, \mathrm{T}_s\} \tag{16}$$

State space

The decision center needs to observe the state of system at each decision moment. The objectives of its observations include:

- Vehicle instantaneous speed and position distribution: the instantaneous position d_i, $i \in N$ of each vehicle in system;
- Vehicle remaining business data: Each requesting vehicle in system has the most remaining business data. Including the part that should be obtained by oncoming vehicles and the part that should be obtained by edge cloud, it is defined as D_i^V and D_i^I. The state at the moment of decision can be expressed by the following formula:

$$s = \left(d_1, d_2, \mathrm{K}\ d_{N+K}, D_1^V, D_1^I, \mathrm{K}\ D_N^V, D_N^I\right) \tag{17}$$

The set of all possible states is the state space, denoted by S. Obviously, S is continuous space.

Decision space

At $T_0 + iT$, in addition to observing system status, edge cloud will also make a decision to adjust the edge cloud data cache ratio and bandwidth allocation. The decision consists of three parts, namely, 1) V2I link bandwidth allocation amount; 2) V2V link bandwidth allocation amount; 3) edge cloud-side cache ratio, the decision can be expressed as $a = \{B_i, B_v, \eta\}$, where $B_i = \{B_{i1}, \ldots, B_{iN}\}$, $B_v = \{B_{v1}, \ldots, B_{vN}\}$, $\eta = \{\eta_1, \ldots \eta_K\}$. The set of all possible decisions under state s is denoted as A_s, and let $A = U_{s \in S} A_s$ be called the system decision space.

Revenue and state transition

The system's benefits at each stage are defined as:

$$r(s, a) = -\sum_{i=1}^{N} \begin{cases} \rho_i T, d_{it} < R_I \ or \ d_{iKt} < R_V \\ \rho_i \tau_n^{'}, others \end{cases} \tag{18}$$

It can be seen that the return directly reflects the size of objective function. That is, the higher time delay of consumption, the smaller return and the smaller objective function. For a specific state s, after the edge cloud captures decision a, the system state will transition to state $s^{'}$ at next decision moment.

Deterministic strategy

A policy distribution describes the possible decision distributions that the system captures in different states. Define the continuous function $\pi_t(a|s) \geq 0$ as a strategy distribution with $\int_{a \in A_s} \pi_t(a|s) = 1$. In this system, the choice of strategy is completely determined by edge cloud and is not affected by random factors in the environment. That is, edge cloud will choose a decision with probability 1 at each decision moment, and record it as $a = \mu(s)$. Such a strategy is called a deterministic strategy.

Decision value function

The decision value function $Q_\pi(s, a)$ is used to describe the expected discounted income obtained by capturing decision $a = \mu(s)$ in state s and strategy π in future states. According to the definition: ·

$$Q_\pi(s,a) = E_\pi\left\{\sum_{t=T_0}^{T_s} \lambda^{t-1} r(s,a)\right\} \tag{19}$$

Here, λ is a discount factor. Then the original optimization goal can be transformed into finding the optimal strategy $\pi*$ such that $Q_{\pi*}(s,a)$=$\max_\pi Q_\pi(s,a)$.

Edge cooperative caching solution based on deterministic policy iteration

Reinforcement learning is an important branch of machine learning. It maximizes / minimizes cumulative expected value of the return / cost function, enabling the agent to learn a set of optimal or near-optimal solutions from real and dynamic environments. There are two main branches of reinforcement learning: model-based methods and model-free methods [21]. The former is mainly used in the field of automatic control. Generally speaking, Bayesian networks are used for modeling in model-based reinforcement learning, and the optimal solution to the problem is obtained through optimization theory. Model-less reinforcement learning framework is considered as a data-driven learning strategy. It learns and predicts optimal value functions and strategy functions to achieve optimal goals. This chapter will focus on adopting a model-free learning framework based on a large number of experiments and historical experiences to adapt to dynamic environment, and to achieve the optimal decision of edge cache system. There are three main types of model-free reinforcement learning: critic models based on value functions, actor models based on strategy functions, and actor-critic based on value functions and strategy functions model. Based on a deterministic actor-critic model, a deep neural network is used to provide accurate estimates of deterministic policy functions and value functions. The combined architecture can be used to solve the proposed joint optimization of content caching and resource allocation for edge cooperation. In addition, deterministic-based strategies can be well applied to continuous bandwidth allocation decisions. The DDPG framework is mainly composed of a main network, a target network and a memory slot. Among them, the main network and the target network are respectively composed of a deep neural network based on an actor model and a critical model.

Training and update rules for deep critic cache network

The network of deep critics mainly learns the predicted value $Q_\theta(s,a)$ of an action value function with parameter θ, that is,

$$Q_\theta(\boldsymbol{s},\boldsymbol{a}) = \theta^T \cdot \boldsymbol{\Phi}(\boldsymbol{s},\boldsymbol{a}) = \sum_{h=1}^{H}\sum_{h'=1}^{H'}\left(\theta_{hh'\phi qh}(s,a) + \theta_{0h'}\right) \tag{20}$$

For the q th training sample, $\Phi = (\phi_{q1}(s,a), \ldots \phi_{qH}(s,a))$ is a activation function vector, $\boldsymbol{\theta}$ is the neural network parameter matrix, $[\theta_{01}, \ldots \theta_{0H'}]$ is the neural network bias parameter vector, $[\theta_{h1}, \ldots \theta_{hH'}]$ is the neural network weight vector, and H, H' is the number of hidden units in the adjacent neural layer.

In order to solve the problem of deterministic strategy exploration [22, 23], offline learning methods are used to distinguish the sampling strategy from exploration strategy. In order to improve the efficiency of cache learning algorithm, a mini-batch method is used instead of the batch gradient descent method. This method updates the parameters of entire neural network at each content transmission time slot. Define the memory slot size as B and set $D = \{1, 2, \ldots q, \ldots D\}$ as the index set of training samples. In order to obtain the loss function, the following error values are defined:

$$\delta_{TD} = c(s,a) + \gamma Q'_{\boldsymbol{\theta}'}\left(s', a'\right) - Q_{\boldsymbol{\theta}}(s,a) \tag{21}$$

Here, γ discount factor, $Q'_{\theta'}(\boldsymbol{s}', \boldsymbol{a}')$ is the real value of target network $\boldsymbol{s}'$ performing action $\boldsymbol{a}'$ in the state. By randomly selecting a predefined number of samples D from the memory slot, and using a random gradient descent strategy to update parameters of the main network, the update rules are as follows:

$$\boldsymbol{\theta} : \boldsymbol{\theta} - \eta_{\boldsymbol{\theta}} \delta_{TD} \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}(s,a) \tag{22}$$

Here, $\eta_{\boldsymbol{\theta}}$ is the step size of stochastic gradient descent. The target network is used to evaluate true performance of computational migration network, that is, output true $Q'_{\boldsymbol{\theta}'}(\boldsymbol{s}', \boldsymbol{a}')$ value.

Training and updating rules for deep actors' cache network

Instead of learning a random strategy distribution function $\pi(s|a)$, the deterministic actor network learns a deterministic strategy $a = \mu_\varpi(s)$ with a parameter ϖ. Similarly, the function is predicted by introducing a deep neural network, expressed as follows:

$$\mu_\varpi(s) = \boldsymbol{\varpi}^{\boldsymbol{T}} \cdot \boldsymbol{\Psi}(\boldsymbol{s},\boldsymbol{a}) = \sum_{h=1}^{H}\sum_{h'=1}^{H'}\left(\varpi_{hh'}\phi_{qh}(s,a) + \varpi_{0h'}\right) \tag{23}$$

For the q th training sample, $\Psi = (\phi_{q1}(s,a), \ldots \phi_{qH}(s,a))$ is the activation function and $\boldsymbol{\varpi}$ is the parameter of neural network. $[\varpi_{01}, \ldots \varpi_{0H'}]$ is the bias parameter of neural network, and $[\varpi_{h1}, \ldots \varpi_{hH'}]$ is the weight parameter of neural network.

The goal of edge cooperative buffering is to minimize joint optimization problem defined in the formula using stochastic gradient descent. The literature has proved that solving the gradient of objective function is equivalent to solving the gradient of $Q(s, a)$. The gradient of parameter is:

$$\nabla_{\varpi} Q_{\theta,\varpi}(s,a) = \frac{\partial \mu_{\varpi}(s)}{\partial \varpi} \frac{\partial Q_{\theta}(s,a)}{\partial a}\bigg|_{a=\mu_{\varpi}(s)} \tag{24}$$

Globally or locally optimal solutions can be found by minimizing Q. The update rule for parameter ϖ can be expressed as:

$$\boldsymbol{\varpi} : \boldsymbol{\varpi} - \eta_{\boldsymbol{\varpi}} \nabla_{\boldsymbol{\varpi}} Q_{\boldsymbol{\theta},\boldsymbol{\varpi}}(s,a) \tag{25}$$

Here, $\eta_{\varpi} > 0$ is the learning rate.

Caching strategy selection method and target neural network update rule

Most studies have shown that if an agent pays too much attention to exploration and utilization strategies, it will cause poor network performance. To solve this problem, the strategy selection method based on Ornstein-Uhlenbeck noise obtains better cache strategy and training performance [24]. The Ornstein-Uhlenbeck method uses the following formula to select current cache strategy. The expression is as follows:

$$\mu(s) = \mu_{\varpi}(s) + \varsigma N(a_{din}) \tag{26}$$

Here, ς is a normal number, a_{din} is the dimension of the cache action space, and N is a standard normal random process.

The update of target network can be viewed as an "approximate replication" of the main network. In fact, for the deep target network structure, such as the number of layers and the number of hidden units, it needs to be consistent with the main network structure. In addition, in order to improve the robustness of learning performance, the update speed of target network parameters should be less than that of the main network. An Exponentially Weighted Moving Average (EWMA) strategy is used here to update the parameters $\boldsymbol{\theta}'$ and $\boldsymbol{\varpi}'$ of target network. The update rules are as follows:

$$\boldsymbol{\theta}' : \tau_{\theta'} \boldsymbol{\theta}' + \left(1 - \tau_{\theta'}\right) \boldsymbol{\theta} \tag{27}$$

$$\boldsymbol{\varpi}' : \tau_{\varpi'} \boldsymbol{\varpi}' + \left(1 - \tau_{\varpi'}\right) \boldsymbol{\varpi} \tag{28}$$

Here, $\tau'_{\theta} \in [0, 1]$ and $\tau'_{\varpi} \in [0, 1]$ are the weights of parameters $\boldsymbol{\theta}'$ and $\boldsymbol{\varpi}'$. $\boldsymbol{\theta}'$ and $\boldsymbol{\varpi}'$ can be regarded as approximate averages of $1/(1-\tau'_{\theta})$ and $1/(1-\tau'_{\varpi})$.

The main parameters of DDPG-based edge cooperative cache algorithm include the total number of simulation experiments, the state space of system, the attributes of content, the learning rate, and the size of memory slot. After completing the initialization of system parameters and hyper parameters, intelligently implement edge caching strategies at the beginning of each experiment, including content placement / update strategies based on large time scales and content transmission strategies based on small time scales. During each experiment, the main network uses Ornstein-Uhlenbeck method and stochastic batch gradient descent method to update the parameters of critical network and the parameters of actor network. Based on the preset number of experiments, EWMA mechanism is used to update the parameters θ' of critical network in the target network and the parameters ϖ' of actor network. Finally, the iteration of algorithm ends after the maximum number of trials.

Experimental results and analysis

In order to verify the performance of our algorithm, the simulation experiment is designed in this section to compare with two cooperative caching strategies in reference [15, 19], and the results are analyzed.

Simulation environment and parameter settings

In this section, Python scripting language is used to build the simulation environment for edge cooperative cache proposed in this paper. And we use deep learning framework of TensorFlow and open source DDPG module to verify the performance of edge cooperative cache. In order to simulate the traffic network environment, this paper uses the Geohash algorithm to implement the division of traffic network. The road network in some areas of the city is divided into 16 transportation areas. The macro base station is set in the center of road network to cover all traffic areas. Deploy 5 RSUs in traffic areas with high traffic flow. By analyzing real traffic data, the destination is predicted under the given current position of vehicles, and the driving path of the starting and ending points of vehicles is simulated. As the number of trials increases, the training performance based on DDPG has the characteristic of violent oscillation within a certain range. This paper uses statistical averages to calculate the cumulative average of system overhead.

Table 1 shows the parameters of edge cache system and DDPG neural network parameters. In order to verify the performance of our proposed edge cooperative cache strategy, two benchmark schemes are proposed: a random edge cache strategy and a non-cooperative edge cache strategy. For the random cache strategy, content of different process levels is randomly placed on vehicles and roadside units. For the non-cooperative edge cache scheme, the joint optimization algorithm proposed in this paper is used to directly deploy the content on the roadside unit.

Table 1 Parameter settings of DDPG neural network

Number of layers in the neural network	2
Number of hidden units	[20, 15]
Maximum number of experiments	3000
Maximum number of content transfer cycles	10
Learning rat	$[10^{-4}, 10^{-6}]$

Vehicle destination prediction and driving path simulation

This section analyzes a set of real vehicles driving data. The data consists of 8 features: sample index, vehicle index, start time, end time, start latitude, start longitude, end latitude and end longitude. By mining these historical traffic data and using different machine learning algorithms to train the model, given the vehicle index and starting point latitude and longitude, accurate prediction of vehicle destination is achieved. Compared with a longer character string to characterize the traffic area where the vehicle is located, using a shorter character string can significantly improve the accuracy of vehicle destination prediction. The heuristic results show that in the case of limited data features, the length of the Geohash characters can be flexibly controlled to control the accuracy of the corresponding traffic area.

Figure 4(a) and (b) are the simulation results of space-time migration of CRV and CCV respectively. In Fig. 4, the abscissa represents an index value of a content transmission slot based on a time domain, and the ordinate represents an index of a traffic area based on an airspace. Simulated 16 road network traffic areas and 10 content transmission time slots. Based on the driving route of CRV2, since CRV2 passes through different traffic areas in each time slot, it can be inferred that the route has a higher traffic efficiency. In addition, a useful result was observed from the CRV9's driving trajectory, the vehicle frequently appeared in a certain traffic area. The reason may be that CRV9 travels to some hot spots, such as subway stations or commercial centers.

Figure 4(b) shows that CCV16 stayed on traffic area-6 for a long time. Therefore, when CRV2 passes through the traffic area, the two can mutually share resources and information through V2V. Besides, CCV20 meets CRV2, CRV5 and CRV9 in different time slots. Therefore, content with higher popularity can be deployed on CCV20 in advance to improve the hit rate of content acquired by CRV and reduce the transmission delay of content. It can be seen from the simulation results shown in Fig. 4 that the content cache strategy is designed based on the vehicle density and the driving path. This will provide richer a-priori information for the edge cooperative cache system, thereby improving cache performance.

Numerical analysis of cache performance

Figure 5 shows the comparison results of system overhead under different cache strategies in an edge cooperative cache system with CRVs = 30 and CCVs = 20. As the number of simulation experiments increases, all

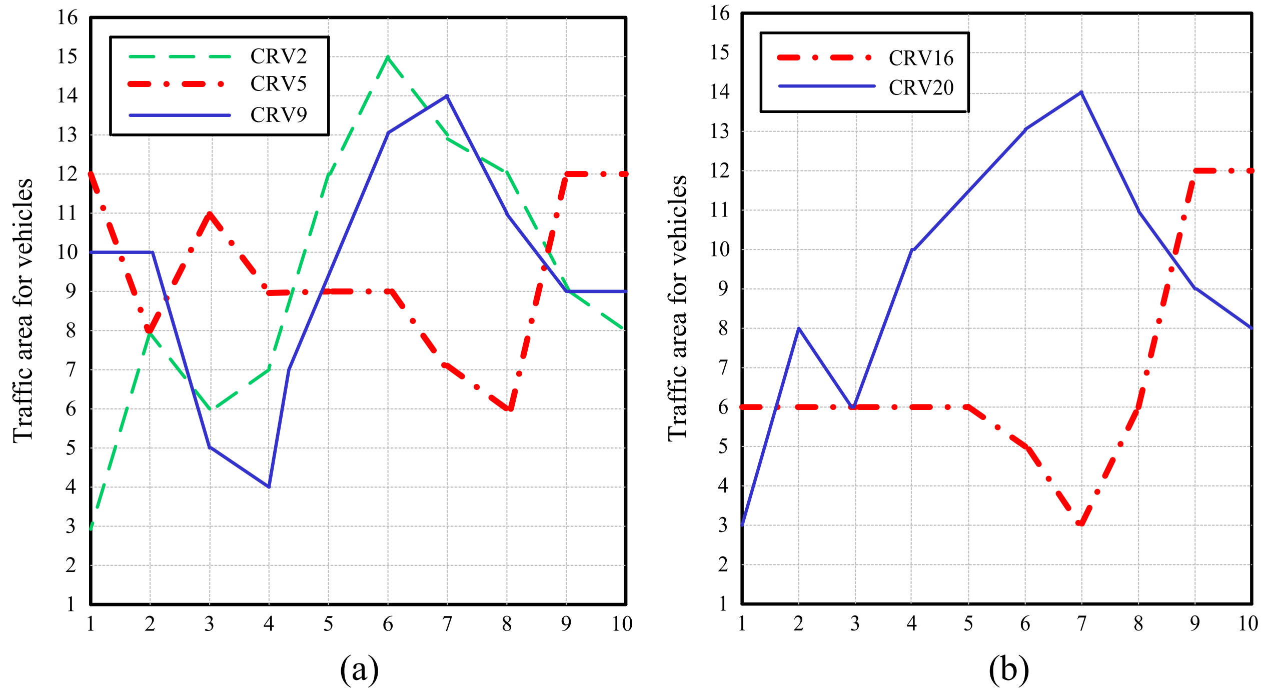

Fig. 4 Simulation of driving path based on vehicle starts and stop positions

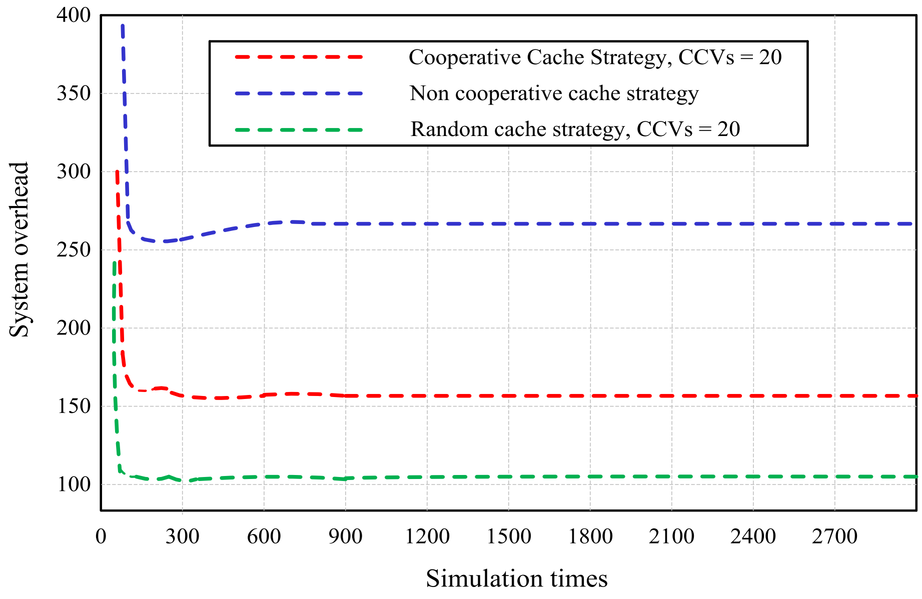

Fig. 5 Comparison of the cumulative average system cost under different cache strategies

three content caching strategies can reach a stable cumulative average. The following results can be obtained from Fig. 5: Firstly, the non-cooperative cache scheme in reference [15] has the highest average system overhead. The main reason is that CRV needs to get more content fragments from the centralized cache node in a single content transmission slot. This results in higher content acquisition costs. Secondly, since CCVs can provide more mobile storage resources, even if reference [19] adopted a random cache strategy, the V2V cooperative cache strategy can effectively reduce system overhead. Thirdly, compared with the two benchmark schemes, the cooperative cache scheme proposed in this paper can minimize system overhead. The simulation results prove the effectiveness of joint content placement and content transmission optimization algorithm from the perspective of reducing system overhead.

Figure 6 shows the comparison results of average transmission delay of content under different cache strategies. In order to reduce resource usage overhead

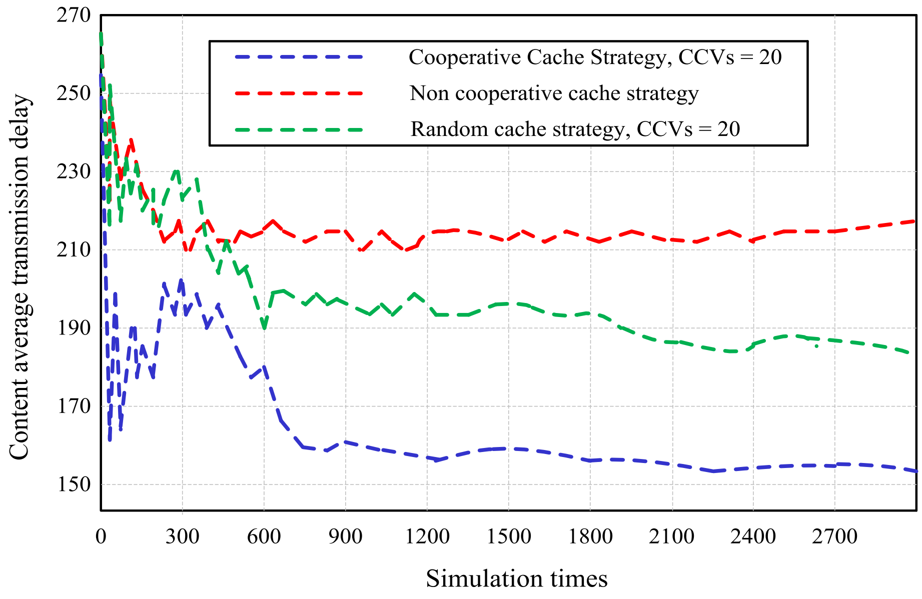

Fig. 6 Comparison of average content transmission delay under different cache strategies

during content transmission, CRVs should look for an access mode with lower resource costs. It gets all the content it needs with the maximum tolerable content transfer latency. For the non-cooperative caching strategy in reference [15], since the static roadside unit does not achieve full coverage of the road network, some CRVs content meets its own performance requirements. For V2V-assisted caching strategies, CCVs can provide richer mobile bandwidth and storage resources. In this case, based on the vehicle's movement path, the chances of CRVs and CCVs establishing V2V connections will be significantly increased. Therefore, reference [19] adopts a random storage strategy to reduce the content transmission delay. Compared with the random caching strategy, the edge cooperative caching strategy proposed in this paper performs joint optimization on content placement / update and content transmission. Therefore, the minimum content transmission delay can be obtained.

Conclusion

In this paper, the problem of joint allocation based on edge cloud communication and cache resources in high-speed free-flow scenarios for vehicle-cloud collaborative caching is studied in depth. The above research proves that the vehicle's driving trajectories can be accurately predicted and the edge cloud cache placement method can be determined. Furthermore, the data can be cached in advance to oncoming vehicles and the edge cloud and transmitted to requesting vehicles at the same time, so as to reduce the user delay for obtaining data. In the above scenario, this paper studies the bandwidth allocation strategy of V2I and V2V communication links and the strategy of adjusting cache ratio between base stations and oncoming vehicles with the goal of minimizing weighted average delay. Besides, the content caching strategy of Actor-Critic (AC) learning model based on deep neural network is adopted. We predict vehicle destinations accurately based on real traffic data and the vehicle travel path between starting point and destination is simulated. The proposed caching strategy can not only improve the hit rate by placing and updating content in advance, but also improves the convergence speed of our algorithm based on deterministic strategy and the small batch gradient descent method. Finally, the simulation results show that the performance of our algorithm is better than the benchmark strategy under different draw speeds, traffic volumes and cache numbers.

Due to the complexity of Internet of Vehicles and the limitations of our research capabilities, it is impossible to fully study the resource management issues in various scenarios and levels in Internet of Vehicles. Therefore, the research in this paper still has certain deficiencies and limitations. From the perspective of Internet of Vehicles resource management, this paper designs computing migration and edge caching strategies to meet users' business needs and performance requirements. The high-speed mobility of vehicles will cause the spatial and temporal migration of road network traffic flow and heterogeneous services of Internet of Vehicles, and then change the vehicle density and vehicle resource requirements in each transportation area. It can be seen that the edge network resource status and road network traffic situation have a certain degree of coupling and interchangeability, which requires coordinated optimization of resource management and road network traffic control behaviors in order to maximize the utilization of communication, computing and storage resources.

Acknowledgements
Not applicable

About the authors
Mu Zhang, received the Master of Software Engineering degree from University of Electronic Science and Technology of China in 2014. He is currently a Senior Engineer and a Master's Supervisor of Information Technology Center, Chengdu Sport University. His research interests include cloud computing,edge computing and computer networks.
Song Wang, Bachelor of Computer Science and Technology, Lecturer. Graduated from Southwest Jiaotong University in 2003.Worked in Sichuan Vocational and Technical College of Communications. His research interests include Internet of Vehicles and Vehicle to Cloud.
Qing Gao, received the M.S. degree from Beijing Normal University in 2002, and the Ph. D degree in biomedical engineering from the University of Electronic Science and Technology of China in 2010. She is currently a Professor and a doctoral supervisor with the School of Mathematical Science, University of Electronic Science and Technology of China. Her current research interests include graph theoretical analysis and pattern recognition of human brain networks.

Authors' contributions
The main idea is proposed by Qing Gao, and the polishing of English language is completed by Qing Gao. The program design and coding are mainly completed by Mu Zhang and Song Wang. The arrangement of the experimental data was completed by Mu Zhang. The author(s) read and approved the final manuscript.

Funding
This work was supported by the National Natural Science Foundation of China (No. 61573085), the Science and Technology Program of Sichuan Province (No. 2017JY0198, No. 2018JY0562), the Project of Sichuan Network Culture Research Center (No. WLWH14–46).

Availability of data and materials
The data will not be shared because it is not applicable to this article.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Information Technology Center, Chengdu Sport University, Chengdu 610041, Sichuan, China. [2]Department of Rail Transit Engineering, Sichuan Vocational and Technical College of Communications, Chengdu 611130, Sichuan, China. [3]School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu 611731, Sichuan, China.

Received: 23 February 2020 Accepted: 8 June 2020
Published online: 18 June 2020

References

1. Woo H, Lee M (2018) A hierarchical location service architecture for VANET with aggregated location update [J]. Comput Commun 125:38–55. https://doi.org/10.1016/j.comcom.2018.05.001
2. Abbasi M, Rafiee M, Khosravi MR, Jolfaei A, Menon VG, Koushyar JM (2020) An efficient parallel genetic algorithm solution for vehicle routing problem in cloud implementation of the intelligent transportation systems [J]. J Cloud Comput 9(1):6
3. Ghane S, Jolfaei A, Kulik L, Ramamohanarao K, Puthal D (2020) Preserving privacy in the internet of connected vehicles. IEEE Transact Intell Transport Syst:1–10. https://doi.org/10.1109/TITS.2020.2964410
4. Qi L, Zhang X, Dou W, Ni Q (2017) A distributed locality-sensitive hashing based approach for cloud service recommendation from multi-source data [J]. IEEE J Select Areas Commun 35(11):2616–2624
5. Jolfaei A, Kant K, Shafei H (2019) Secure data streaming to untrusted road side units in intelligent transportation system [C]. In: 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). IEEE, Piscataway, pp 793–798
6. Abid H, Phuong LTT, Wang J et al (2011) V-cloud: vehicular cyber-physical systems and cloud computing [C]// International Symposium on Applied Sciences in Biomedical & Communication Technologies. ACM, New York, pp 1–5
7. Yu R, Zhang Y, Gjessing S et al (2013) Toward cloud-based vehicular networks with efficient resource management [J]. IEEE Netw 27(5):48–55
8. Cordeschi N, Amendola D, Shojafar M et al (2015) Distributed and adaptive resource management in cloud-assisted cognitive radio vehicular networks with hard reliability guarantees [J]. Vehicular Commun 2(1):1–12
9. Zhang W (2016) Duan, Pengcheng, gong, Wenjuan, et al. a load-aware pluggable cloud framework for real-time video processing [J]. IEEE Transact Indust Inform 12(6):2166–2176
10. Wu Y, Ni K, Zhang C et al (2018) NOMA assisted multi-access Mobile edge computing: a joint optimization of computation offloading and time allocation [J]. IEEE Trans Veh Technol 67(12):12244–12258
11. Cordeschi N, Amendola D, Baccarelli E (2015) Reliable adaptive resource Management for Cognitive Cloud Vehicular Networks [J]. IEEE Trans Veh Technol 64(6):2528–2537
12. Lin CC, Deng DJ, Yao CC (2017) Resource allocation in vehicular cloud computing systems with heterogeneous vehicles and roadside units [J]. IEEE Internet Things J 5(5):3692–3700
13. Zheng K, Meng H, Chatzimisios P et al (2015) An SMDP-based resource allocation in vehicular cloud computing systems [J]. IEEE Trans Ind Electron 62(12):7920–7928
14. Liang H (2012) An SMDP-based service model for Interdomain resource allocation in mobile cloud networks [J]. IEEE Trans Veh Technol 61(5):2222–2232
15. Shojafar M, Cordeschi N, Baccarelli E (2019) Energy-efficient adaptive resource management for real-time vehicular cloud services [J]. IEEE Transact Cloud Comput 7(1):196–209
16. Glass S, Mahgoub I, Rathod M (2017) Leveraging MANET based cooperative cache discovery techniques in VANETs: a survey andAnalysis [J]. IEEE Commun Surv Tutorials 19(4):2640–2661
17. Zheng J, Tong H, Wu Y (2017) A cluster-based delay tolerant routing algorithm for vehicular Ad Hoc Networks [C]// 2017 IEEE 85th vehicular technology conference: VTC2017-spring. IEEE, Piscataway, pp 1–6
18. Kumar N, Lee JH (2014) Peer-to-peer cooperative caching for data dissemination in urban vehicular communications [J]. IEEE Syst J 8(4):1136–1144
19. Huang W, Song T, Yang Y et al (2019) Cluster-based cooperative caching with mobility prediction in vehicular named data networking [J]. IEEE Access 7:23442–23458
20. Zhou YF, Chen N (2019) The LAP under facility disruptions during early post-earthquake rescue using PSO-GA hybrid algorithm [J]. Fresenius Environ Bull 28(12A):9906–9914
21. Bradley A, van der Meer R, McKay C (2019) Personalized pancreatic cancer management: a systematic review of how machine learning is supporting decision-making [J]. Pancreas 48(5):598–604
22. Qi L, Dou W, Hu C, Zhou Y, Yu J (2015) A context-aware service evaluation approach over big data for cloud applications. IEEE Transact Cloud Comput. https://doi.org/10.1109/TCC.2015.2511764
23. Qi L, Dou W, Wang W, Li G, Yu H, Wan S (2018) Dynamic Mobile crowdsourcing selection for electricity load forecasting. IEEE Access 6: 46926–46937
24. Chen F, Mamon R, Nkurunziza S (2018) Inference for a change-point problem under a generalised Ornstein–Uhlenbeck setting. Ann Inst Stat Math 70:807–853. https://doi.org/10.1007/s10463-017-0610-4

Publisher's Note