EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

Computing resource allocation scheme of IOV using deep reinforcement learning in edge computing environment

Yiwei Zhang[1], Min Zhang[2], Caixia Fan[1*], Fuqiang Li[1] and Baofang Li[1]

* Correspondence: fancaixia@henau.edu.cn
[1]College of Sciences, Henan Agricultural University, Zhengzhou 450002, Henan, China
Full list of author information is available at the end of the article

Abstract

With the emergence and development of 5G technology, Mobile Edge Computing (MEC) has been closely integrated with Internet of Vehicles (IoV) technology, which can effectively support and improve network performance in IoV. However, the high-speed mobility of vehicles and diversity of communication quality make computing task offloading strategies more complex. To solve the problem, this paper proposes a computing resource allocation scheme based on deep reinforcement learning network for mobile edge computing scenarios in IoV. Firstly, the task resource allocation model for IoV in corresponding edge computing scenario is determined regarding the computing capacity of service nodes and vehicle moving speed as constraints. Besides, the mathematical model for task offloading and resource allocation is established with the minimum total computing cost as objective function. Then, deep Q-learning network based on deep reinforcement learning network is proposed to solve the mathematical model of resource allocation. Moreover, experience replay method is used to solve the instability of nonlinear approximate function neural network, which can avoid falling into dimension disaster and ensure the low-overhead and low-latency operation requirements of resource allocation. Finally, simulation results show that proposed scheme can effectively allocate the computing resources of IoV in edge computing environment. When the number of user uploaded data is 10K bits and the number of terminals is 15, it still shows the excellent network performance of low-overhead and low-latency.

Keywords: Internet of Vehicles, Mobile edge computing, Reinforcement learning, Experience replay method, Resource allocation, Offloading strategy

1 Introduction

In recent years, the automobile industry has brought tremendous changes to people's lives under the impetus transformation of information and communication technology. The applications equipped on vehicles can provide drivers and passengers with more useful help information, such as safety information, surrounding environmental conditions and traffic information [1–3].

The emergence of Internet of Vehicles (IoV) can integrate information provided by multiple applications to solve many problems in transportation [4–6]. The IoV network uses vehicles as basic information unit. Within a certain communication range, road entities such as pedestrians, vehicles and roadside facilities are connected to traffic management network by sensor technology, information acquisition technology, access technology, transmission technology and networking technology. The mobile network is connected to the backup network, which serves applications such as vehicle safety, traffic control, information services and user network access. It aims to establish an intelligent comprehensive network system that improves traffic conditions and travel efficiency, and expands information interaction forms.

The traditional IoV network communication can only meet part of the network needs of vehicle users, and is mainly suitable for some applications with small calculation amount and low delay sensitivity [7]. The intelligent technology of automobiles is gradually being widely used and promoted according to the popularization and development of IoV technology at this stage. With the development and popularization of fifth-generation communication technology, IoV market has also spawned a large number of new service applications (such as unmanned intelligent driving), which have more stringent requirements for network bandwidth, offloading delay, etc. [8, 9]. Therefore, the traditional IoV communication has been unable to meet the current operating requirements, which brings huge challenges to IoV in terms of computing and communication capabilities.

In recent years, Mobile Edge Computing (MEC), as a key 5G technology, is of great significance to alleviating the congestion of cloud network or the core layer of data center in IoV. MEC deploys computing and storage resources at the network edge to provide IT services and cloud computing capabilities for mobile networks. It can greatly accelerate the execution speed of computing tasks [10, 11], solve the problem of insufficient computing resources for vehicle itself, and provide users with ultra-low latency and high bandwidth network service solutions.

Task offloading is one of the key technologies of MEC. Scientifically and rationally, part or all of the computing tasks of in-vehicle devices are handed over to edge computing server for processing, which can effectively solve the problems of in-vehicle devices in terms of resource storage, computing performance and energy efficiency, which reduces communication and calculation delay. This in turn enables real-time operation of IoV network and higher responsiveness [12, 13]. But at the same time, it should be noted that the complex network scenarios of IoV also bring many problems to MEC technology application. The high-speed mobility of vehicles and the diversity of communication quality in IoV make computing task offloading strategies more complicated. Thus, the research on offloading decision-making and execution resource allocation has become a key issue that urgently needs to be solved in vehicle edge computing.

2 Methods

The deep integration of IoV and MEC technology, relying on a new generation of information and communication technology to build a new format of intelligent vehicles, to achieve friendly information interaction between vehicles and the outside world, can support the development needs of next generation for "vehicle connected everything" [14]. However, with the development of intelligent and informatization of IoV, the application of in-vehicle terminals has gradually developed towards multimedia entertainment, which has caused an explosive growth of task data. This has put a heavy pressure on the scarce network resources [15]. Therefore, for the limited resources of IoV, rational allocation of vehicle's own resources can be more effective in realizing that IoV also meets the efficient network computing capabilities when vehicles are running fast, providing the quality of user experience and improving traffic efficiency.

The in-depth integration of IoV and MEC technology relies on a new generation of information and communication technology to build a new format of smart vehicles. This can realize friendly information interaction between vehicles and the outside world, and can support the development needs of next generation for "car-connected everything" [14]. However, with the development of intelligent and informatization of IoV, the application of in-vehicle terminals has gradually developed towards multimedia entertainment, which has caused an explosive growth of task data. This has put a heavy pressure on the scarce network resources [15]. Therefore, for the limited resources of IoV, rational allocation of vehicle's own resources can be more effective in realizing that IoV also meets the efficient network computing capabilities when vehicles are running fast, providing the quality of user experience and improving traffic efficiency.

The joint management of wireless networks and computing resources is the key to achieving high efficiency and low latency in IoV networks. The network architecture in which MEC server and wireless access point coexist promotes the realization of related technologies [16]. For the resource management and offloading decisions of MEC system, scholars have launched corresponding researches. Literature [17] proposed a convex optimization problem to minimize the total energy consumption of mobile devices. The optimal strategy for controlling the size of offloaded data and time allocation had a simple threshold-based structure. The offloading priority function was derived based on channel conditions and local calculation energy consumption, and the full offloading and minimum offloading are performed respectively based on a given threshold. Literature [18] used dynamic voltage and frequency scaling techniques to minimize local execution energy consumption for tasks with strict execution deadlines, and used data transmission scheduling to optimize the energy consumption of computing offload. Literature [19] proposed an end-to-end communication task offloading framework based on network assistance, which can realize resource sharing among mobile users. Literature [20] proposed a cooperative downloading scheme to offload traffic from cellular networks by VANETs. Appropriate data was obtained from cellular network, and the data is distributed to vehicles in an approximately optimal way, and a storage time aggregation graph for planning data transmission was designed. Literature [21] proposed a cloud-edge-based MEC vehicle network offloading framework, which reduces the time consumption of computing tasks and the impact of vehicle mobility.

The existing traditional optimization algorithms are feasible to solve the problems of MEC computing offloading and resource allocation. But it should be noted that the

time slot interval divided by MEC system is very small. Traditional optimization algorithms generally require complicated operations and iterations to obtain optimization results. Thus, traditional optimization algorithms are not very suitable for high real-time MEC systems.

Reinforcement Learning (RL) is very suitable for solving decision-making problems, such as computational offloading decision [22]. The RL algorithm can create experience to learn and complete the optimization goal by a trial-return feedback mechanism that is different from traditional optimization algorithms. The deep learning algorithm can learn the characteristics of historical data, and after the training is completed, it has a great efficiency improvement compared with traditional optimization algorithms. If you use traditional algorithm data for training, you can combine the advantages of two. Literature [23] proposed a distributed wireless resource allocation based on multi-agent theory and reinforcement learning algorithm. This allowed devices to independently select resource blocks and power levels, ensuring that network system had low complexity and signaling overhead. Literature [24] developed an optimal and adaptive vehicle cloud resource allocation model for car networking systems based on Semi Markov Decision Process (SMDP) and reinforcement learning algorithms. It considered the balance between IoV network resource costs and system revenue, make optimization decisions on IoV network service quality and vehicle user experience quality to optimize the total system overhead of IoV network. Literature [25] proposed a new architecture that combined with reinforcement learning algorithms to dynamically orchestrate edge computing and cache resources. It improved the practicability of system and maximized its utility. Literature [26] proposed a task scheduling and resource allocation model based on hybrid ant colony optimization and deep reinforcement learning. This model took the shortest overall task completion time and highest utilization rate of idle resources as goals. The space complexity is reduced and network performance is improved by using weighted values to construct a binary ordered traversal tree and deep reinforcement learning algorithm.

In this paper, oriented to the precise needs of mobility characteristics and task allocation for IoV users, drawing on the existing task management research of MEC, this paper proposes a computing resource allocation scheme using deep reinforcement learning in edge computing environment. The main contributions of this paper are as follows:

1) In order to clarify the mathematical model of MEC task distribution algorithm proposed in this paper, this paper considers the computing power of service nodes and vehicle speed on the basis of determining system network model, computing model and communication model of task offloading and resource allocation. The cache capacity of service nodes is a constraint. Moreover, a mathematical model of task offloading and resource allocation is established with the minimum total computing cost of system as objective function.
2) In order to achieve fast and efficient vehicle network computing resource allocation and avoid the limited dimensions of traditional Q-learning network solving task resource allocation algorithm, this paper proposes a task computing resource allocation scheme based on deep Q network. This scheme uses experience replay method as the training method to solve the instability of Q-learning network due to nonlinear approximation function. It realizes the optimal allocation of task

resources, so that resource allocation can improve the corresponding operating speed while ensuring low overhead.

The rest of this paper is organized as follows. Section 3 introduces the vehicle network resource allocation system model and the corresponding mathematical calculation problem description. Section 4 introduces the task distribution and offloading based on DQN algorithm. Section 5 builds simulation scenarios based on related protocols to verify the performance of proposed method. Section 6 concludes the paper.

3 System model and problem description

3.1 System model

This paper analyzes the corresponding resource allocation scheme based on a vehicle cloud collaborative edge cache model as the network model. The specific vehicle network model is shown in Fig. 1. In this model, there are L RSUs deployed around the road, denoted as $\mathscr{L} = \{\mathscr{M}_1, \mathscr{M}_2, \mathscr{M}_3, \cdots, \mathscr{M}_L\}$, and each RSU is equipped with an MEC server. The Poisson distribution is suitable for describing the number of random events in unit time (or space). Therefore, it is assumed that N vehicles on the road have a Poisson distribution [27], which is expressed as $\mathcal{V} = \{v_1, v_2, v_3, \cdots, v_N\}$. Since both MEC server and neighboring vehicles have computing and caching capabilities, they are collectively referred to as service nodes $\mathcal{W} = \{w_1, w_2, w_3, \cdots, w_M\}$. n vehicles are randomly distributed within the coverage area of each RSU, that is, the set of vehicles within the coverage area of RSU or the service area of $\mathscr{M}_j$ is $\mathcal{V}_j = \{v_1, v_2, \cdots, v_n\}$. The vehicle 802.11p OBU has an 802.11p network interface and a cellular network interface. Vehicles can offload tasks to MEC servers for calculation by RSU, or offload to

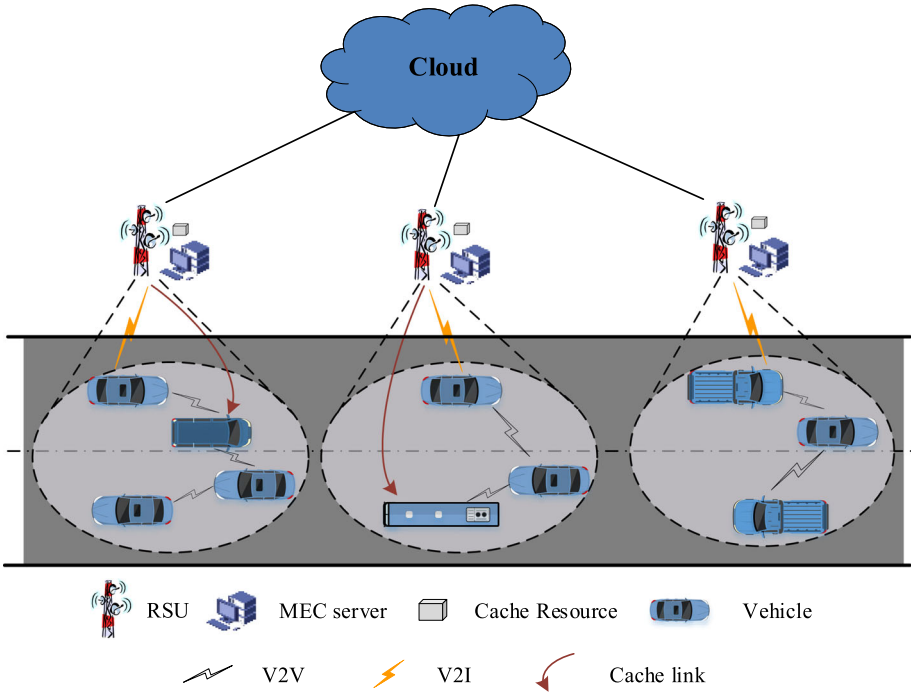

Fig. 1 System model structure diagram

neighboring vehicles for V2V communication. In order to effectively reuse spectrum, V2I mode and V2V mode work in the same frequency band. The spectrum is evenly divided into K sub-channels, denoted as $\mathcal{K} = = \{1, 2, 3, \cdots, K\}$, and the bandwidth of each sub-channel is B Hz. The vehicle offloading strategy set is expressed as $\mathcal{A} = = \{a_1, a_2, a_3, \cdots, a_N\}$, if $a_i = 1$, it means v_i, and the task is offloaded to service nodes for calculation. If $a_i = 0$, it means that v_i will perform computing tasks locally. Assume that at t, there are some tasks in buffer pool. When vehicles have a task request, if the task is cached on service nodes, service nodes inform vehicles that the task exists on service nodes. When the calculation of service nodes is completed, it is directly sent back to vehicles. In this way, the vehicle does not need to perform task offloading operations, which can effectively reduce the energy consumption of mobile devices and the delay of task offloading. If there is no cache for requested tasks on service nodes, the vehicle needs to make an offloading decision and further resource allocation. When the service node completes requested tasks for the first calculation, it considers the cache decision. The cache strategy set of service nodes w_m is denoted as $\mathcal{G}_m = = \{g_{m,1}, g_{m,2}, g_{m,3}, \cdots, g_{m,n1}\}$. If $g_{m,\ n1} = 1$, it means that service node w_m will cache computing task $n1$. This allows the next request to reduce network transmission and reduce calculation delay. The cache collection of all service nodes is denoted as $\mathcal{AG} = = \{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \cdots, \mathcal{G}_M\}$.

3.2 Computing model

Based on the system model built above, it is assumed that each task requesting vehicle has a computing task $\mathcal{Z} = \{d_i, s_i, t_i^{\max}\}$, $i \in N$ to be processed. Where d_i represents the input size of task $\mathcal{Z}_i$. s_i represents the number of CPU cycles required to complete computing task $\mathcal{Z}_i$. $t_i^{\max}$ is the maximum delay that computing task $\mathcal{Z}_i$ can tolerate. The vehicle can offload tasks to MEC servers for calculation by RSU, or offload to neighboring vehicles for processing, or execute on local vehicles.

For offloading computing, when the limited computing power of vehicle itself is not enough to support the time delay requirement of tasks, the task needs to be offloaded to service nodes for calculation. The task processing process will inevitably bring time delay and energy consumption. Since the data volume of processing results returned is small, the delay and energy consumption of return process are ignored, and only the upload delay, calculation delay and transmission energy consumption are considered [28, 29].

In this paper, the task request vehicle to offload tasks to service node w_j calculation process is defined as the weighted combination of delay and energy consumption, expressed as:

$$u_i^{off} = \alpha t_i^{off} + \beta e_i^{off} \tag{1}$$

where α and β respectively represent the weighting factors of non-negative delay and energy consumption, and satisfy $\alpha + \beta \le 1$. $t_i^{off} = \frac{d_i}{r_{i,j}} + \frac{s_i}{f_j^i}$ represents the sum of offloading delay and calculation delay. f_j^i represents the computing resources allocated by service node w_j to vehicle v_i. $e_i^{off} = p_i \frac{d_i}{r_{i,j}}$ represents the energy consumption of transmission process.

For local calculations, suppose that the computing power of vehicle v_i is F_i^l, and the computing power of different vehicles is different. When vehicle task $\mathcal{Z}_i$ is calculated locally, the cost that vehicle v_i needs to bear is:

$$u_i^l = \alpha t_i^l + \beta e_i^l \tag{2}$$

where $t_i^l = \frac{s_i}{F_i^l}$ is the time delay required for calculation. $e_i^l = \phi s_i (F_i^l)^2$ represents the energy consumption to perform tasks. ϕ is the power coefficient of energy consumed per CPU cycle [30].

3.3 Communication model

When the traditional orthogonal multiple access technology is applied in MEC system, each terminal user has a one-to-one corresponding transmission channel to ensure stable signal transmission. The delay T_v^{OMA} in completing task offloading in this scenario is expressed as follows:

$$T^{OMA} = \frac{S_v}{B \log\left(1 + \frac{p_v^{OMA}|h_v|^2}{p_v}\right)} \tag{3}$$

where p_v^{OMA} represents the transmission power of user v. h_v represents the channel gain between users and edge servers. p_v represents the noise interference power of users. B represents the channel transmission bandwidth of users. Thus, the total time delay T^{OMA} to complete the offloading of all vehicle users is expressed as:

$$T^{OMA} = \sum_{v=1}^{V} T_v^{OMA} \tag{4}$$

In a communication network based on hybrid NOMA-MEC, this system can allow multiple vehicle users to complete task transmission and offloading in the same time slot or frequency band. Suppose there are two car network users m and n requesting task offloading at the same time, $D_n \geq D_m$, $m, n \in \{1, 2, \ldots, v\}$. Thus, in this mode, users m and n can simultaneously offload tasks to MEC servers in time slot D_m. The transmission power of vehicle users m and n are p_m^{OMA} and p_n^{OMA} respectively. It should be pointed out here that if the information of user m is decoded in the second stage of serial interference cancellation, the performance of user m is same as OMA. Therefore, the transmission delay of user m will not be affected [31]. The expression of user n transmission rate R_n in time slot D_m is:

$$R_n \leq B \log\left(1 + \frac{p_{nm}^{NOMA}|h_n|^2}{p_m^{OMA}|h_m|^2 + p_v}\right) \tag{5}$$

where p_{nm}^{NOMA} represents the transmission power of vehicle user n in time slot D_m. h_m and h_n represent the channel gains of vehicle users m and n respectively.

The task offloading of end users by NOMA will generate more energy consumption than OMA mode [32]. Therefore, this paper uses a hybrid NOMA-MEC method to offload the tasks requested by mobile terminal users. The specific steps are: firstly, user m and user n perform task offloading at the same time within time

D_m. Secondly, after user m completes task offloading, user n needs to continue the task offloading in OMA manner. It takes T_n^{re} to complete the offloading of this part of tasks, so total time delay T_n of vehicle user n is:

$$T_n = D_m + \frac{S_n - R_n D_m}{B \log\left(1 + \frac{p_{nn}^{NOMA}}{p_v}|h_n|^2\right)} \tag{6}$$

where p_{nn}^{NOMA} represents the transmission power offloaded by vehicle user n in the second part. The time delay T_m of actual offloading for vehicle user m is expressed as:

$$\begin{aligned} T_m = & \frac{S_m}{B \log\left(1 + \frac{p_m^{OMA}|h_m|^2}{p_v}\right)} \\ & s.t. T_m \le D_m \end{aligned} \tag{7}$$

3.4 Problem description

When a smart vehicle requests a task calculation, it first checks whether there is a content cache in its own buffer pool. If the content is available locally, there is no need to post a task request. Otherwise, scan the surrounding service node to see if there is a content cache, and if it exists, it will be returned after the service node calculation is completed. If it does not exist, you need to consider whether to offload.

After the task is offloaded to service nodes and the calculation is completed, service nodes consider the update of cache. After the content is returned, the service ends. This paper aims to minimize system overhead through proper offloading and caching decisions, as well as the allocation of communication and computing resources. Thus, the optimization goal is expressed as:

$$\begin{aligned} & \min_{\mathcal{A},\mathcal{C},\mathcal{P},\mathcal{F},\mathcal{AG}} U(\mathcal{A},\mathcal{C},\mathcal{P},\mathcal{F},\mathcal{AG}) \\ & = \sum_{i=1}^{N} hit_{j,i} u_i^{cache} + (1 - hit_{i,j}) g_{j,i} \left[(1-a_i)u_i^l + a_i u_i^{off}\right] \\ & = \sum_{i=1}^{N} hit_{j,i} \alpha \frac{s_i}{f_j^i} + (1 - hit_{i,j}) g_{j,i} \left\{ (1-a_i)\left[\alpha \frac{s_i}{F_j^i} + \beta \mathcal{K} s_i \left(f_i^l\right)^2\right] \right. \\ & \qquad \left. + a_i \left[\alpha\left(\frac{d_i}{r_{ij}} + \frac{s_i}{f_j^i}\right) + \beta p_i \frac{d_i}{r_{ij}}\right] \right\} \end{aligned} \tag{8}$$

$$s.t.\ \ C1: a_i \in \{0,1\}, \forall i \in \mathcal{N} \tag{9}$$

$$C2: c_{i,k} \in \{0,1\}, \forall i \in \mathcal{N}, k \in \mathcal{K} \tag{10}$$

$$C3: g_{j,i} \in \{0,1\}, \forall i \in \mathcal{N} \tag{11}$$

$$C4: 0 < p_i < p_{\max}, \forall i \in \mathcal{N} \tag{12}$$

$$C5: f_j^i > 0, \forall i \in \mathcal{N} \tag{13}$$

$$C6: \sum_{i \in N} a_i f_j^i \leq F_j^{\max}, \forall i \in \mathcal{N}, j \in \mathcal{M} \tag{14}$$

$$C7: (1-a_i) t_i^{local} + a_i t_i^{off} \leq \min\left\{ t_i^{\max}, \frac{L_j}{V_u}, \frac{d_{interrupt}}{|V_u - V_v|} \right\}, \forall i \in \mathcal{N} \tag{15}$$

$$C8: \sum_{i=1}^{N} g_{j,i} d_i \leq H_j \tag{16}$$

where $\mathcal{A}$ represents the offloading decision set of all task request vehicles. $\mathcal{C}$ represents the channel allocation status; $\mathcal{P}$ is the task transmission power set of offloaded vehicles. $\mathcal{F}$is the computing resource allocation strategy, and $\mathcal{AG}$ represents the cache decision of service nodes.

In equations (9) to (16), constraints C1 and C3 indicate that the offloading decision is a 0-1 decision. C2 indicates that the channel allocation matrix is a binary variable. C4 ensures that the power distribution is non-negative and does not exceed the range of uplink transmission power. C5 and C6 indicate that the computing resource allocation does not exceed the maximum computing capacity of service nodes. C7 represents the delay constraint, where L_j is the coverage of RSU_j and V_u is the moving speed of vehicle requested by tasks. V_v is the moving speed of service vehicles, and $d_{interrupt}$ is the maximum interruption distance. C8 indicates that the cache content of service nodes cannot exceed its maximum cache capacity.

4 Offloading decision based on deep reinforcement learning

As an optimization problem, IoV network resource allocation problem is essentially a mixed integer nonlinear programming model. Traditional optimization algorithms are used to solve the model has the problem of obtaining sub-optimal solutions [33, 34]. In order to achieve fast and efficient mathematical model solving, this paper uses deep Q network to calculate nonlinear mathematical problems. This can avoid the danger of traditional Q-learning network easily falling into a dimensional disaster, so that the vehicle network resource allocation can improve the corresponding operating speed while ensuring low overhead.

4.1 *Q*-learning Network

Q-learning is a classic reinforcement learning algorithm, that is a method of recording Q-value. Each state and action group has a value $Q(s, a)$. For each step, the agent calculates and stores $Q(s, a)$ in Q table. This value can be regarded as the expectation of long-term return, $Q(s, a)$ update formula can be expressed as:

$$Q(s,a) = r(s,a) + \gamma^* \max Q\left(s', a'\right) \tag{17}$$

where (s, a) is the current state and action; (s', a') is the state and action of next time slot. This paper defines γ as the learning rate, and γ is a constant that satisfies $0 \leq \gamma \leq 1$. It is worth noting that if γ tends to 0, it means that the agent mainly considers current instantaneous return. If γ tends to 1, it means that the agent is also very concerned about future returns. For each step, iterate the value of $Q(s, a)$. In this way, we can get the optimal $\mathcal{A}$.

Algorithm 1 shows the corresponding operation process of Q-learning algorithm.

Algorithm 1 : *Q*-learning algorithm

Step 1 Gives the learning rate parameter γ and return matrix

Step 2 Initializes $Q(s,a)=0$

Step 3 For each episode

3.1 Randomly select an initial state s

3.2 If the target state is not reached, perform the following steps

(1) Among all the possible behaviors in current state s, select action a with the greatest return

(2) Execute the selected action a to get the next state s'

(3) Q (s, a) is calculated according to formula (18)

(4) $s=s'$

4.2 Offloading decision algorithm based on DQN

In order to further reduce the amount of calculation of IoV network computing resource allocation and improve the real-time performance of algorithm, Deep Q-learning Network (DQN) approximate estimation $Q(s,a)$ is used. It realizes the traversal of enough sample states to make the algorithm meet the needs of actual engineering environment.

DQN algorithm enables V-UEs to dynamically make the best offloading decision based on their behavior and the behavior of edge cloud. This process is formulated as a limited Markov Decision Process (MDP). It is defined as a tuple $\boldsymbol{M}=(\boldsymbol{S},\boldsymbol{A},\boldsymbol{R})$, where $\boldsymbol{S}$ and $\boldsymbol{A}$ represent state and behavior spaces. $\boldsymbol{R}(s,a)$ represents the timely reward for performing action a in state s. π is a strategy that matches a behavior a from a state s, such as $\pi(s)=a$. The main goal of V-UEs is to find the optimal strategy π^* to minimize the utility obtained by users, thereby minimizing energy consumption and delay.

State space $\boldsymbol{S}$ is the number of task offloading requests Q^u of V-UEs and the size of remaining tasks in edge cloud Q^c. The distance D between V-UEs and the edge cloud consists of three parts, which are defined as follows:

$$\boldsymbol{S}=\{s=(Q^u,Q^c,D)\} \tag{18}$$

Behavior space $\boldsymbol{A}$ is expressed as:

$$\boldsymbol{A} = \{a = (a_0, ..., a_x, ..., a_X) | a_x \in (0, 1, ..., a_{\max})\} \tag{19}$$

where a_0 represents the task sequence processed locally; a_x represents the sequence offloaded to edge cloud. $a_{\max}$ is the maximum number of tasks that are processed locally or offloaded to the cloud in each decision cycle. The total number of tasks for each behavior is less than or equal to the number of tasks currently staying in user queue.

The instant return is the cost of V-UEs making the optimal offloading decision in each system state. Thus, the instant reward matrix $\boldsymbol{R}(s, a)$ for a given behavior a in state s is:

$$\boldsymbol{R}(s, a) = \boldsymbol{U}(s, a) - \boldsymbol{C}(s, a) \tag{20}$$

where $\boldsymbol{U}(s, a)$ and $\boldsymbol{C}(s, a)$ are instant utility matrix and cost matrix respectively. For immediate utility, it can be expressed as:

$$\boldsymbol{U}(s, a) = \rho\left(O_{i,j}^{L} + O_{i,j}^{C}\right) \tag{21}$$

where ρ is the utility constant. Correspondingly, $\boldsymbol{C}(s, a)$ cost matrix can be expressed as:

$$\boldsymbol{C}(s, a) = \eta_1 \boldsymbol{E}(s, a) + \eta_2 \boldsymbol{T}(s, a) \tag{22}$$

where η_1 and η_2 are constants. $\boldsymbol{E}(s, a)$ and $\boldsymbol{T}(s, a)$ are energy consumption and delay matrices respectively, expressed as follows:

$$\boldsymbol{E}(s, a) = a_0 e_{i,j}^{L}(s, a) + \sum_{x=1}^{X} a_d e_{i,j}^{C}(s, a) \tag{23}$$

$$\boldsymbol{T}(s, a) = a_0 t_{i,j}^{L}(s, a) + \sum_{x=1}^{X} a_d t_{i,j}^{C}(s, a) \tag{24}$$

$\boldsymbol{Q}$ matrix is an online learning scheme of model-free deep learning algorithm In this scheme, V-UEs select behavior a_t for potassium planting in state s_t at time step t to minimize the immediate future return [35]. $\boldsymbol{Q}$ matrix can be expressed as:

$$\boldsymbol{Q}^{*}(s, a) = \text{-}\max E\left[r_t + \sum_{k=1}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a, \pi\right] \tag{25}$$

where r_t is the minimum reward for adopting an offloading strategy π after performing behavior a in state s at time step t. $E[\cdot]$ represents the expectation function; γ is the attenuation coefficient. $\boldsymbol{Q}$ matrix is a neural network approximator $\boldsymbol{Q}(s, a; \theta)$, θ is a weighting factor. In each decision cycle, state vector $\boldsymbol{S} = (Q^{\mu}, Q^{c}, D)$ taken by V-UEs for the first time is used as the input of $\boldsymbol{Q}$ matrix, and all possible behaviors $\boldsymbol{A}$ are used as the output of $\boldsymbol{Q}$ matrix. Then V-UEs select the behavior according to $\varepsilon - greedy$ method. In addition, $\boldsymbol{Q}$ matrix is iteratively adjusted θ to minimize the loss function. Therefore, the loss function at time step t can be defined as:

$$L_t(\theta_t) = -E\left[\left(r_t + \gamma \max_{a'} \boldsymbol{Q}\left(s_{t+1}, a'; \theta_{t-1}\right) - \boldsymbol{Q}(s_t, a_t, \theta_t)\right)^2\right] \tag{26}$$

In other words, given a converted $\langle s_t, a_t, r_t, s_{t+1}\rangle$ weight factor θ, $\boldsymbol{Q}$ matrix is updated by minimizing the square error between the current predicted $\boldsymbol{Q}$ value $\boldsymbol{Q}(s_t, a_t)$ and the target $\boldsymbol{Q}$ value $r_t + \gamma \max_{a'} \boldsymbol{Q}(s_{t+1}, a')$.

In addition, the empirical replay method is used as a training method to solve the instability of Q network due to the nonlinear approximation function in DQN. More specifically, user experience $e_t = \langle s_t, a_t, r_t, s_{t+1} \rangle$ is stored in the memory $\Omega = \{e_{t-\psi}, \ldots, e_t\}$. At each time step t, a random mini-batch conversion is selected from memory to train Q network instead of the most recent conversion e_t.

Figure 2 shows the corresponding DQN-based offloading decision algorithm flow chart. From Fig. 2, we can see that the algorithm steps 2-4 are recursion. $\boldsymbol{Q}$ value is

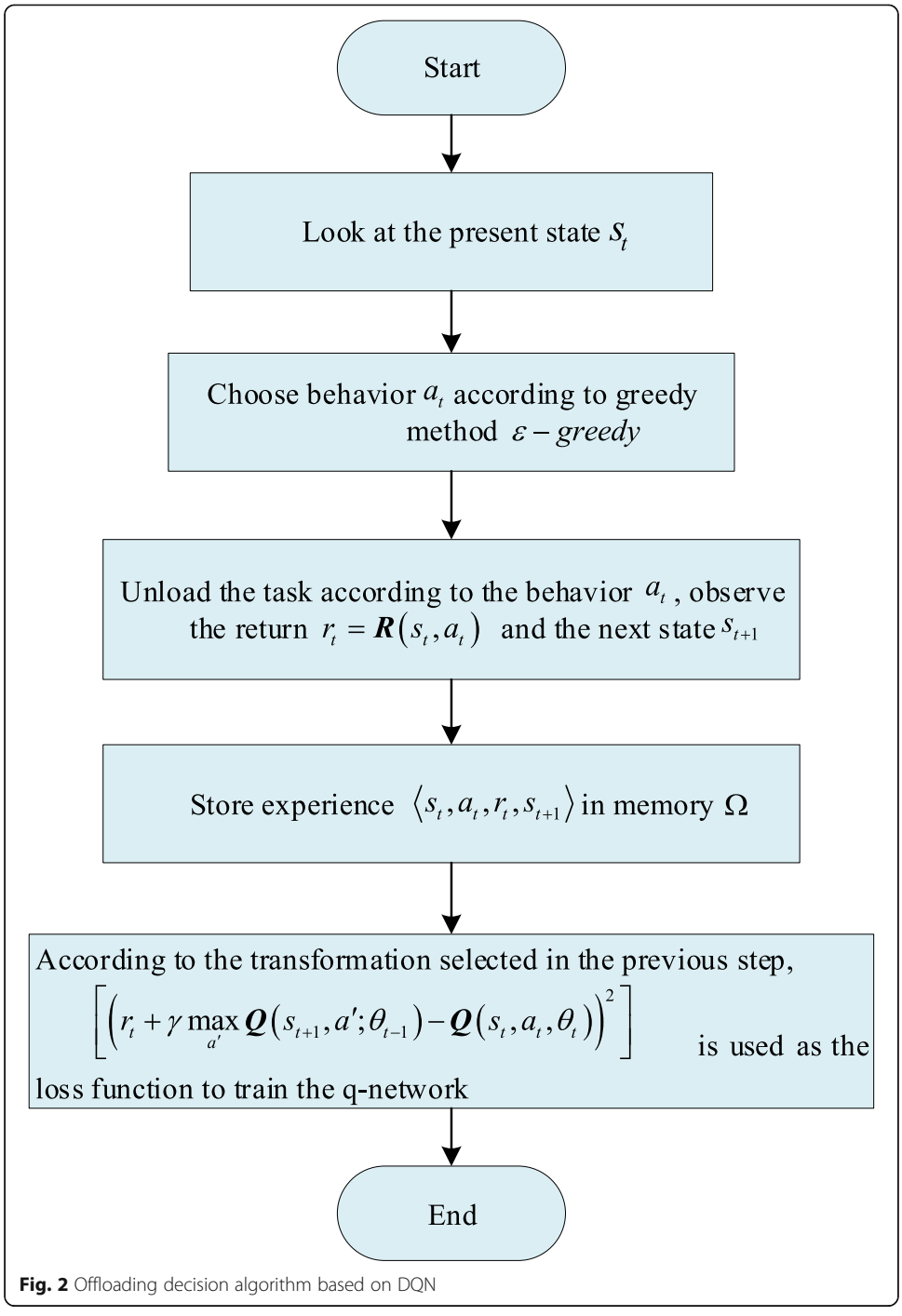

Fig. 2 Offloading decision algorithm based on DQN

estimated according to Q network, and the offloading decision action made by users at the beginning of each decision period is presented. Steps 5-7 use the experience replay method to train Q network.

5 Experimental

5.1 Simulation setting

In this section, MATLAB simulation platform is used to verify the efficient performance of proposed resource allocation mechanism in DQN algorithm-based vehicle network under edge computing environment. This experiment is carried out in the context of IEEE 802.11p vehicle network scene standard and MEC white paper, using the channel gain model proposed in 3GPP standardization.

The simulation scenario is set to a one-way straight road, and vehicles running on the road can communicate with roadside base stations as well as vehicle-to-vehicle communication. The purpose is to simulate proposed MEC task distribution algorithm based on deep reinforcement learning and evaluate the performance in different situations. This paper mainly considers 3 communities along the roadside. Each cell is equipped with RSU and MEC server, and the coverage radius of RSU is 500 meters. The specific simulation parameters are shown in Table 1.

5.2 Algorithm sensitivity analysis

In order to verify the superiority of proposed method for the allocation of computing resources in IoV tasks, a discussion and analysis are carried out from two aspects: the total system computing overhead and time delay. Then it achieves the superior performance of proposed method in this paper with low overhead and high real-time in task allocation.

Table 1 Experimental simulation parameter setting

Parameter	Numerical Value
Maximum transmitting power of vehicle	25 dbm
Task calculation size	22.5 MHz
CPU weeks required for computing tasks	15-25 MB
Gaussian white noise power	1500~3000 Megacycles
Weight factor setting	-106 dbm
Coverage radius of RSU	0.75
Vehicle computing power	1-2GHz
Computing power of MEC server	11.2GHz
Number of uplink transmission channels	10
The number of days of vehicles in a single community	15
Distributed parameter	0.59
Vehicle cache capability	115 MB
Cache capability of MEC server	512 MB
Vehicle moving speed	40 Km/h, 60Km/h
Maximum interruption distance	345 m

5.2.1 Sensitivity analysis of total system overhead

In this paper, two basic methods, "full local calculation" and "full offload calculation", are compared and verified with proposed method. Discuss and analyze the relationship between number of users, the computing capacity of servers, and the volume of uploaded data and the total computing overhead of system. "Full local calculation" means that all users choose local calculation. "Full offload calculation" means that all users choose to offload calculation. At this time, the computing resources of MEC servers are equally distributed to each user.

Figure 3 is a graph showing the relationship between total expenditure and the number of users. On the whole, when the number of users continues to increase, the total cost of the three methods is on the rise.

In Fig. 3, the performance of proposed DQN method is relatively stable and can achieve the best results. When the number of users reaches 15 cars, the total system overhead can still be kept at a low level compared with the comparison method. Among them, there is almost no difference between the curve of full offloading method and DQN when the number of users is 4. But when the number of vehicles increases, the total cost increases rapidly. The analysis believes that when the number of users increases and all of them choose to offload computing, MEC servers with limited computing resources cannot provide sufficient computing resources for each user, which reduces the overall performance.

Figure 4 is an analysis diagram of the influence of computing capacity for MEC servers on weighted total overhead. It can be seen from the figure that as the computing capacity of servers increases, for the total system overhead, the method proposed in this paper can always maintain a lower level than the comparison method, and has obvious advantages in computing performance.

It can be seen from Fig. 4 that the more special one is the all local calculation curve, and the weighted total overhead does not change with the calculation capacity of MEC

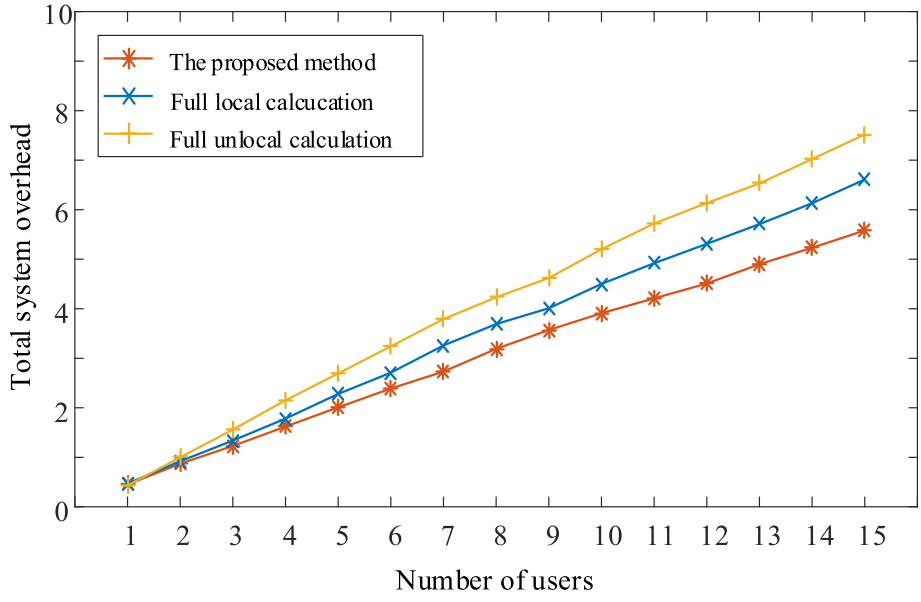

Fig. 3 Relationship between total cost and number of users

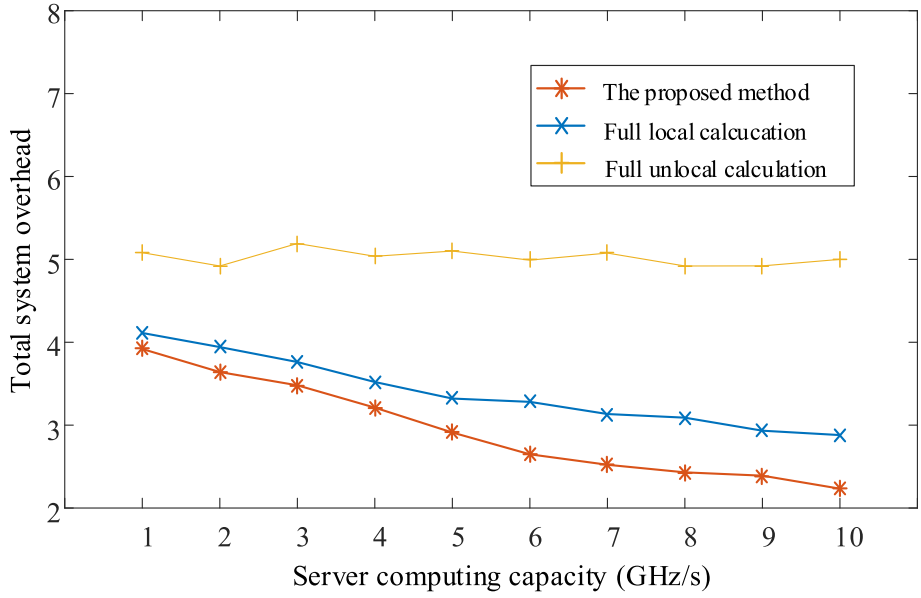

Fig. 4 Relationship between total cost and server computing capacity

servers. Obviously, this is because the number of computing resources of MEC servers has no effect on the local computing process. The other two curves show a downward trend as F increases. This is because the larger F is, the server can allocate more computing resources to users, thereby reducing processing time and energy consumption. The curve of DQN method proposed in this paper is always at the bottom and performs best.

Figure 5 shows the performance of various algorithms under different upload data volume conditions. It can be seen from Fig. 5 that as the size of uploaded data increases, the curves of all algorithms show an upward trend. Because a larger amount of data means more time to upload and process data, this process also increases energy consumption correspondingly, leading to an increase in the total system overhead. According to Fig. 5, DQN method we proposed has the best effect because it rises the slowest among these three lines. The upward trend of all locally calculated curve is much higher than other two curves, and the performance gap with other two algorithms is getting bigger and bigger.

5.2.2 Sensitivity analysis of system time delay

For the distribution of computing tasks in IoV, the delay is also an important indicator to measure the quality of resource allocation. In order to prove that proposed algorithm can further meet the needs of practical engineering applications, the algorithm of literature [25] and the algorithm of literature [26] are selected here as a comparison method and the method proposed in this paper is compared and verified.

Figure 6 is a simulation result of the number of users requesting task offloading and the total time delay of task offloading. Compared with literature [25] and literature [26], DQN algorithm proposed in this paper has a slower increase in time delay.

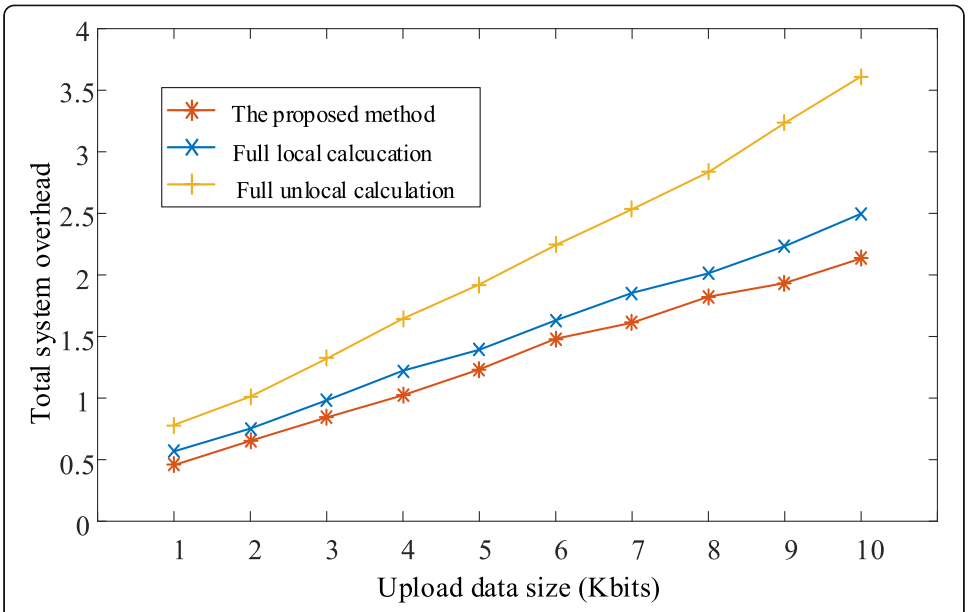

Fig. 5 Relationship between total cost and the size of uploaded data

Besides, when the number of users reaches 15 and the offloading delay reaches the upper limit of 235ms, it has obvious advantages in fast calculation.

It can be seen from Fig. 6 that as the number of users increases, the total delay of task offloading also gradually increases. At the same time, the total delay gap of task offloading under different modes has gradually increased. The reason for the above phenomenon is that when the number of users requesting task offloading is small, the channel resources in the three modes are relatively sufficient, which can satisfy users to perform offloading at the same time. However, with the further increase in number of users, the problem of insufficient channel resources has gradually emerged. The users

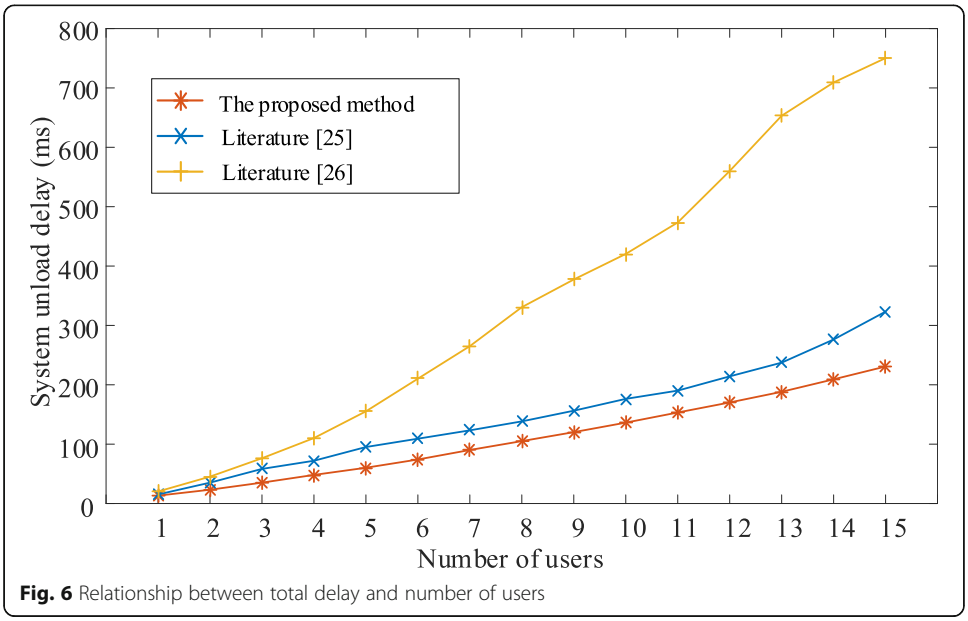

Fig. 6 Relationship between total delay and number of users

in literature [25] and literature [26] need to perform task offload sequence, and wait for other users to complete tasks before offloading. The offloading strategy method proposed in this paper can satisfy more users to offload tasks at the same time under limited channel resources.

Figure 7 is a simulation diagram of task offloading delay and data size for a single user in different modes. According to the simulation results, it can be found that the data size of user task offloading is linearly positively correlated with the offloading delay. In the three offloading modes, when the size of offloading tasks is the same, there is no big difference in offloading delay. The reason for the above simulation results is that when a single user requests task offloading, the channel resources of communication network model are abundant, which can ensure that offloading requests are transmitted with the optimal channel bandwidth.

In summary, compared with other current task resource allocation methods, DQN algorithm-based task resource allocation method for IoV proposed in this paper has a good performance in edge computing environment. The algorithm not only guarantees the low-overhead computing performance of system, but also realizes lower-latency communication, which provides a better service experience for users in IoV.

6 Conclusion

The high-speed mobility of vehicles and diversity of communication quality in current IoV make offloading strategies for computing tasks more complicated. To solve the problem, this paper proposes a computing resource allocation scheme based on deep reinforcement learning network in MEC scenarios. Considering the computing power of service nodes and vehicle moving speed as constraints, the scheme builds a task resource allocation model in edge computing scenario with the minimum total system computing cost as objective function. In addition, deep Q learning network is used to solve the mathematical model of resource allocation, experience replay method is used to avoid dimension disaster and ensure the low-

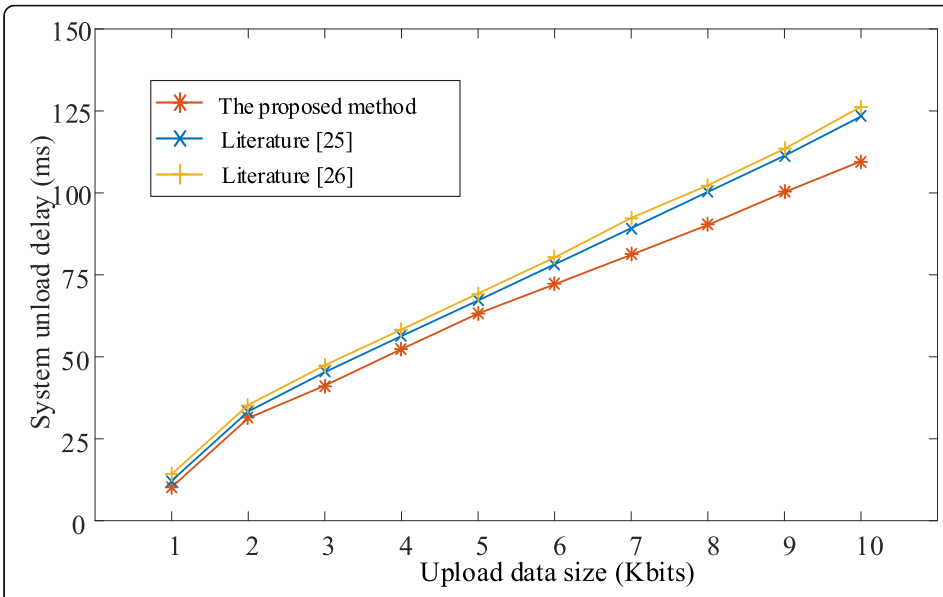

Fig. 7 Relationship between time delay and the size of uploaded data

overhead and low-latency operation requirements of resource allocation. Simulation results prove that the proposed scheme still shows excellent network performance with low overhead and low latency when the amount of user upload data is 10K bits and the number of terminals is 15.

The future research will be to explore the platformization of our proposed method and strive to realize its commercialization.

Acknowledgements
We wish to express their appreciation to the reviewers for their helpful suggestions which greatly improved the presentation of this paper.

Authors' contributions
The main idea of this paper is proposed by Caixia Fan. The algorithm design and experimental environment construction are jointly completed by Yiwei Zhang and Min Zhang. The experimental verification was completed by all the five authors. The writing of the article is jointly completed by Baofang Li and Fuqiang Li. And the writing guidance, English polish, is completed by Caixia Fan. All authors read and approved the final manuscript.

Funding
This work was supported by National Natural Science Foundation of China [No. 61703146]; Scientific and Technological Project of Henan Province [No. 202102110126]; Backbone teacher project of Henan Province [No. 2020GGJS048] and key scientific research projects of colleges and universities in Henan Province [No. 19B413002].

Availability of data and materials
The data included in this paper are available without any restriction.

Declarations

Ethics approval and consent to participate
Our manuscript does not involve research manuscripts of human participants, human data, or human tissues, so our manuscript does not require the statement of ethical approval and ethical consent.

Consent for publication
Our manuscript does not contain any individual person's data in any form, so we do not need the consent of others.

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

Author details
[1]College of Sciences, Henan Agricultural University, Zhengzhou 450002, Henan, China. [2]State Grid Henan Skills Training Center, Zhengzhou, Henan, China.

Received: 19 March 2021 Accepted: 21 June 2021
Published online: 30 June 2021

References
1. J. Zhang, K.B. Letaief, Mobile Edge Intelligence and Computing for the Internet of Vehicles. Proc. IEEE **108**(2), 246–261 (2020)
2. S.S. Shah, M. Ali, A.W. Malik, et al., vFog: A Vehicle-Assisted Computing Framework for Delay-Sensitive Applications in Smart Cites. IEEE ACCCESS **7**(1), 1–10 (2019)
3. D.J. He, S. Chan, M. Guizani, Security in the Internet of Things Supported by Mobile Edge Computing. IEEE Commun. Mag. **56**(8), 56–61 (2018)
4. A. Nanda, D. Puthal, J.J.P.C. Rodrigues, et al., Internet of Autonomous Vehicles Communications Security: Overview, Issues, and Directions. IEEE Wirel. Commun. **26**(4), 60–65 (2019)
5. H. Lu, Q. Liu, D. Tian, et al., The Cognitive Internet of Vehicles for Autonomous Driving. IEEE Netw. **33**(3), 65–73 (2019)
6. B. Vaidya, H.T. Moufta, IoT Applications and Services for Connected and Autonomous Electric Vehicles. Arab. J. Sci. Eng. **45**(4), 2559–2569 (2019)
7. Y. Yang, K. Hua, Emerging technologies for 5G-enabled vehicular networks. IEEE Access **7**(1), 181117–181141 (2019)
8. L. Guevara, F.A. Cheein, The Role of 5G Technologies: Challenges in Smart Cities and Intelligent Transportation Systems. Sustainability **12**(16), 1–15 (2020)
9. X. Zhu, F. Qi, Y. Feng, Deep-Learning-Based Multiple Beamforming for 5G UAV IoT Networks. IEEE Netw. **34**(5), 32–38 (2020)
10. H. Ji, O. Alfarraj, A. Tolba, Artificial Intelligence-Empowered Edge of Vehicles: Architecture, Enabling Technologies, and Applications. IEEE Access **8**(1), 61020–61034 (2020)
11. Y. Cao, H. Song, O. Kaiwartya, et al., Mobile Edge Computing for Big-Data-Enabled Electric Vehicle Charging. IEEE Commun. Mag. **56**(3), 150–156 (2018)
12. G. Hong, W. Su, Q. Wen, et al., RAVEC: An Optimal Resource Allocation Mechanism in Vehicular MEC Systems. J. Inf. Sci. Eng. **36**(4), 865–878 (2020)

13. J. Zhou, F. Wu, K. Zhang, et al., *Joint optimization of Offloading and Resource Allocation in Vehicular Networks with Mobile Edge Computing* (2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)., 2018)
14. C. Yang, Y. Liu, X. Chen, et al., Efficient Mobility-Aware Task Offloading for Vehicular Edge Computing Networks. IEEE Access **7**(1), 26652–26664 (2019)
15. H. Wang, X. Li, H. Ji, et al., in *2018 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*. Dynamic Offloading Scheduling Scheme for MEC-enabled Vehicular Networks (IEEE, 2018)
16. J. Feng, Z. Liu, C. Wu, et al., Mobile Edge Computing for the Internet of Vehicles: Offloading Framework and Job Scheduling. IEEE Veh. Technol. Mag. **14**(1), 28–36 (2019)
17. C. You, K. Huang, H. Chae, et al., Energy-efficient Resource Allocation for Mobile Computation Offloading. IEEE Trans. Wirel. Commun. **16**(3), 1397–1411 (2016)
18. Y. Zhao, V.C.M. Leung, H. Gao, et al., in *2018 IEEE International Conference on Communications (ICC 2018)*. Uplink Resource Allocation in Mobile Edge Computing-Based Heterogeneous Networks with Multi-Band RF Energy Harvesting (IEEE, 2018), pp. 1–6
19. M. Liu, Y. Richard, Y. Teng, et al., Computation Offloading and Content Caching in Wireless Blockchain Networks With Mobile Edge Computing. IEEE Trans. Veh. Technol. **67**(11), 11008–11021 (2018)
20. Y. Sun, L. Xu, Y. Tang, et al., Traffic Offloading for Online Video Service in Vehicular Networks: A Cooperative Approach. IEEE Trans. Veh. Technol. **67**(8), 7630–7642 (2018)
21. K. Zhang, Y. Mao, S. Leng, et al., Mobile-Edge Computing for Vehicular Networks: A Promising Network Paradigm with Predictive Off-Loading. IEEE Veh. Technol. Mag. **12**(2), 36–44 (2017)
22. K. Wang, X. Wang, X. Liu, et al., Task Offloading Strategy Based on Reinforcement Learning Computing in Edge Computing Architecture of Internet of Vehicles. IEEE ACCESS **8**(1), 173779–173789 (2020)
23. S. Xu, S. Zheng, Wireless resource allocation algorithm based on Multi-Agent Reinforcement Learning in M2M Communication. Journal of Beijing Jiaotong University **42**(05), 1–9 (2018)
24. H. Liang, X. Zhang, J. Zhang, et al., A Novel Adaptive Resource Allocation Model Based on SMDP and Reinforcement Learning Algorithm in Vehicular Cloud System. IEEE Trans. Veh. Technol. **68**(10), 10018–10029 (2019)
25. Y. Dai, D. Xu, S. Maharjan, et al., Artificial Intelligence Empowered Edge Computing and Caching for Internet of Vehicles. Wireless Communications, IEEE Wireless Communications **26**(3), 12–18 (2019)
26. U. Rugwiro, C. Gu, W. Ding, Task Scheduling and Resource Allocation Based on Ant-Colony Optimization and Deep Reinforcement Learning. Journal of Internet Technology **20**(5), 1463–1475 (2019)
27. R. Jin, X. Du, K. Zeng, et al., Delay Analysis of Physical-Layer Key Generation in Dynamic Roadside-to-Vehicle Networks. IEEE Trans. Veh. Technol. **66**(3), 2526–2535 (2019)
28. J. Zhang, W. Xia, F. Yan, et al., Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing. IEEE Access **6**(1), 19324–19337 (2018)
29. C. Wang, C. Liang, F.R. Yu, et al., Computation Offloading and Resource Allocation in Wireless Cellular Networks With Mobile Edge Computing. IEEE Trans. Wirel. Commun. **16**(8), 4924–4938 (2017)
30. L. Tianze, W. Muqing, Z. Min, *Consumption considered optimal scheme for task offloading in mobile edge computing* (International Conference on Telecommunications. IEEE, 2016)
31. F. Wang, J. Xu, Z. Ding, Multi-Antenna NOMA for Computation Offloading in Multiuser Mobile Edge Computing Systems. Communications. IEEE Trans. Commun. **67**(3), 2450–2463 (2019)
32. Z. Ding, P. Fan, H.V. Poor, Impact of Non-orthogonal Multiple Access on the Offloading of Mobile Edge Computing. IEEE Trans. Commun. **67**(1), 375–390 (2019)
33. Y. Zhou, H. Yu, Z. Li, et al., Robust Optimization of a Distribution Network Location-Routing Problem under Carbon Trading Policies. IEEE Access **8**(1), 46288–46306 (2020)
34. Y. Zhou, B. Zheng, J. Su, et al., The joint location-transportation model based on grey bi-level programming for early post-earthquake relief. Journal of Industrial and Management Optimization (2020). https://doi.org/10.3934/jimo.2020142
35. L.D. Van, C.K. Tham, *A Deep Reinforcement Learning based Offloading Scheme in Ad-hoc Mobile Clouds* (IEEE Conference on Computer Communications Workshops. IEEE, 2018), pp. 760–765

Publisher's Note