Mobile Networks and Applications (2020) 25:1736–1745
https://doi.org/10.1007/s11036-020-01584-6

Adaptive Task Offloading in Vehicular Edge Computing Networks: a Reinforcement Learning Based Scheme

Jie Zhang[1] · Hongzhi Guo[2] · Jiajia Liu[2]

Published online: 25 June 2020

Abstract

In recent years, with the rapid development of Internet of Things (IoTs) and artificial intelligence, vehicular networks have transformed from simple interactive systems to smart integrated networks. The accompanying intelligent connected vehicles (ICVs) can communicate with each other and connect to the urban traffic information network, to support intelligent applications, i.e., autonomous driving, intelligent navigation, and in-vehicle entertainment services. These applications are usually delay-sensitive and compute-intensive, with the result that the computation resources of vehicles cannot meet the quality requirements of service for vehicles. To solve this problem, vehicular edge computing networks (VECNs) that utilize mobile edge computing offloading technology are seen as a promising paradigm. However, existing task offloading schemes lack consideration of the highly dynamic feature of vehicular networks, which makes them unable to give time-varying offloading decisions for dynamic changes in vehicular networks. Meanwhile, the current mobility model cannot truly reflect the actual road traffic situation. Toward this end, we study the task offloading problem in VECNs with the synchronized random walk model. Then, we propose a reinforcement learning-based scheme as our solution, and verify its superior performance in processing delay reduction and dynamic scene adaptability.

Keywords Vehicular networks · Mobile edge computing · Reinforcement learning

1 Introduction

In the past, vehicular networks were only defined as some massive interactive networks of information such as vehicles' location, speed, and route. However, with the rapid development of 5th generation mobile networks, Internet of Things (IoTs) and artificial intelligence, the explosion of data and computing needs has made vehicular networks begin to enter the intelligent era. Vehicular networks are transforming into integrated networks that enable traffic management, dynamic information services, and vehicle control with intelligence [1–3]. This trend of vehicular networks has attracted the attention of automakers such as Toyota, Volvo, and Mercedes-Benz. The accompanying intelligent connected vehicles (ICVs) with high-intelligent in-vehicle information systems can be connected to the urban traffic information network and smart grid. All the real-time information about road conditions, weather, and emergencies can be obtained at any time, and then the corresponding data can be provided. These features will lead to a new lifestyle shift, such as smart inter-operable ICVs that can improve road safety and traffic efficiency for smoother traffic [4, 5].

In recent vehicular networks, one serious problem that has been elicited is the contradiction between computing-intensive, delay-sensitive applications (e.g., autonomous driving, intelligent navigation, and in-vehicle entertainment services) and resource-limited vehicles [6–8]. To overcome this defect, the use of cloud computing in vehicular networks has been adequately studied in recent years. By offloading part of the computation tasks to the cloud servers, the computing resource pressure of vehicles is somewhat relieved [9]. However, the inherent drawback

✉ Jiajia Liu
liujiajia@nwpu.edu.cn

Jie Zhang
zhangjie_xd@outlook.com

Hongzhi Guo
hongzhi.guo@nwpu.edu.cn

[1] School of Cyber Engineering, Xidian University, Xi'an, Shaanxi, 710071, China

[2] School of Cybersecurity, Northwestern Polytechnical University, Xi'an, Shaanxi, 710072, China

of cloud computing, i.e., the cloud servers are often placed in a distance of several kilometers away from users, so that cloud computing cannot solve the problem fundamentally [10, 11]. In such a case, vehicular edge computing networks (VECNs), which introduce mobile edge computing offloading (MECO) technology has been seen as a promising solution [12]. By deploying edge servers at the road side units (RSUs), VECNs extend computing resources from the cloud to the edge, and provide MECO services to vehicles at the edge of the networks, further reducing the processing delay of vehicles' compute-intensive applications [13–15].

In VECNs, one of the most critical issues is how to obtain the optimal task offloading decisions configuration, to minimize the processing delay of vehicles' computation tasks. In recent years, researchers have conducted extensive research on task offloading techniques in VECNs. By utilizing game theory, convex/non-convex optimization, and other optimization techniques, many well-established methods have been developed [16–18]. For example, Du et al. [16] studied the problem of computing offloading and resource allocation in VECNs, and proposed a Lyapunov optimization-based online algorithm, to minimize the average cost of smart vehicular terminals and MEC enabled roadside units, respectively.

However, the optimization techniques such as traditional convex/non-convex optimization are insufficient for decision-making problem in highly dynamic scenario of vehicular networks. The network nodes of vehicular networks are mainly vehicles with highly dynamic characteristics such as high moving speeds, and thus have a network topology with frequent changes, resulting in a relatively short communication path life. In such case, the usual schemes cannot meet the requirements of the time-changing offloading decisions in VECNs. In response to this situation, we introduce machine learning as our solution. Machine learning, as an important research direction in the field of artificial intelligence, has been applied to many fields [19]. The use of machine learning in vehicular networks can proactively monitor and predict the status of all network components and take appropriate offloading decisions, enabling operators to meet variable requirements in real time to achieve efficient resource utilization [20, 21].

Meanwhile, existing computing offloading research work in VECNs usually only consider the simple assumption of the mesoscopic mobility in which the vehicles move at constant speeds on the road. However, in real life, vehicles have different modes of movement and different speeds due to their different types, road condition, and subjective willingness of drivers. Therefore, this assumption cannot truly reflect the real situation of road traffic. In this paper, we adopted the synchronized random walk model (a widely used traffic model in the highway) to simulate actual traffic conditions [22, 23]. Then, we propose a reinforcement learning-based task offloading scheme to solve the task offloading problem in VECNs. In particular, the main contributions of this paper are as follows:

- We design an SDN enabled vehicular edge computing network architecture, which integrates the computing resources of remote cloud servers and edge servers, to alleviate the computing resources pressure of vehicles.
- In such a scenario, we adopt the synchronized random walk model to simulate real traffic conditions, in which the distribution of the vehicles follows homogeneous Poisson spatial distribution, and the speeds of the vehicles follow the Gaussian speed distribution.
- Given the above assumptions of architecture and system model, we study the task offloading problem, and propose a reinforcement learning-based scheme, to minimize the processing delay of all the vehicles' computation tasks.

The rest of the paper is organized as follows. First, we give the related work in Section 2. Then, we present the VECNs architecture and system model covered in this paper in Section 3. In Section 4, the definition of the task offloading problem and the corresponding solution are explained in detail. After that, we present a series of simulation experiments to verify the performance of our scheme. Finally, we summarize the full paper in Section 6.

2 Related work

Mobile edge computing offloading has received widespread attention in both academic and industrial fields over the past decade [12, 24, 25], with the primary goal of reducing task processing delay or energy consumption. For example, the authors in [25] studied the problem of single/multiple user computing offloading in MEC enabled Internet of Things, and formulated it as a mixed-integer linear programming problem. After that, they proposed an iterative heuristic resource allocation algorithm as their solution, to reduce task processing delay. You et al. [26] investigated the resource allocation in a multi-user MEC offloading system, and designed a threshold-based task offloading policy structure. To reduce the energy consumption of mobile devices' computation tasks in multi-access edge computing networks, Guo et al. [27] introduced Fiber-Wireless (FiWi) technology, and further integrated the computing resources of centralized cloud servers and MEC servers. After that, they studied the problem of Cloud-MEC collaborative task offloading in such a scenario, and a game theory-based collaborative task offloading scheme was proposed to solve the problem.

Meanwhile, many researchers have extended MECO to the vehicular networks, and further proposed vehicular edge computing networks (VECNs) [7, 14]. In VECNs, the primary research goal is to reduce tasks' processing delay, and for which there has been a lot of research work. The authors in [7] proposed a FiWi enhanced vehicular edge computing networks and studied the task offloading problem in this scenario. Specifically, they took into account both vehicle-to-vehicle communications and vehicle-to-infrastructures communications to support communication in a dynamic environment of the vehicular networks. To reduce the processing delay of the tasks, they proposed two collaborative task offloading schemes, and the corresponding simulation experiments verified their superior performance. Qiao et al. [14] collaborative task offloading and output transmission mechanism in vehicular, to reduce the processing delay of the tasks. Also, they used 3D scene reconstruction as a research example to demonstrate the performance of the proposed scheme.

However, the research work mentioned above are not for the dynamics of the VECNs, resulting in their lack of scalability. Toward this end, some researchers have begun to pay attention to the applicability of machine learning in VECNs, due to the significant advantages of machine learning in a dynamic environment [20, 28, 29]. For example, Sun et al. [28] designed an adapting learning task offloading scheme based on multi-armed bandit theory, in which the vehicles can learn the offloading delay performance of their neighboring vehicles. To verify the performance of their scheme, they compared it to the existing upper confidence bound based learning algorithm and achieved a 30% average delay reduction. The authors in [29] designed a knowledge-driven task offloading scheme, and utilized reinforcement learning to extract optimal task offloading strategy directly from the environment.

To the best of the authors' knowledge, although the above research work has solved the problem of task offloading in VECNs, they only considered the mesoscopic mobility model in which vehicles move at a constant speed. This model cannot truly reflect the actual road traffic situation, thus leading to the gap between theory and reality. Moreover, existing research work are still insufficient for the highly dynamic feature of vehicular networks, which makes them unable to provide time-varying offloading decisions for dynamic changes in vehicular networks.

3 System model

As shown in Fig. 1, there is a unidirectional straight road with two traffic streams, and N vehicles are moving on the road. For the distribution and the speeds of vehicles, we assume that the vehicles follow a synchronized random walk mobility model, which is a traffic model widely used in highway [22, 23]. Specifically, the model has the following assumptions:

– The vehicles in two traffic streams follow homogeneous Poisson spatial distribution with densities ρ_1 and ρ_2 at the beginning, respectively.

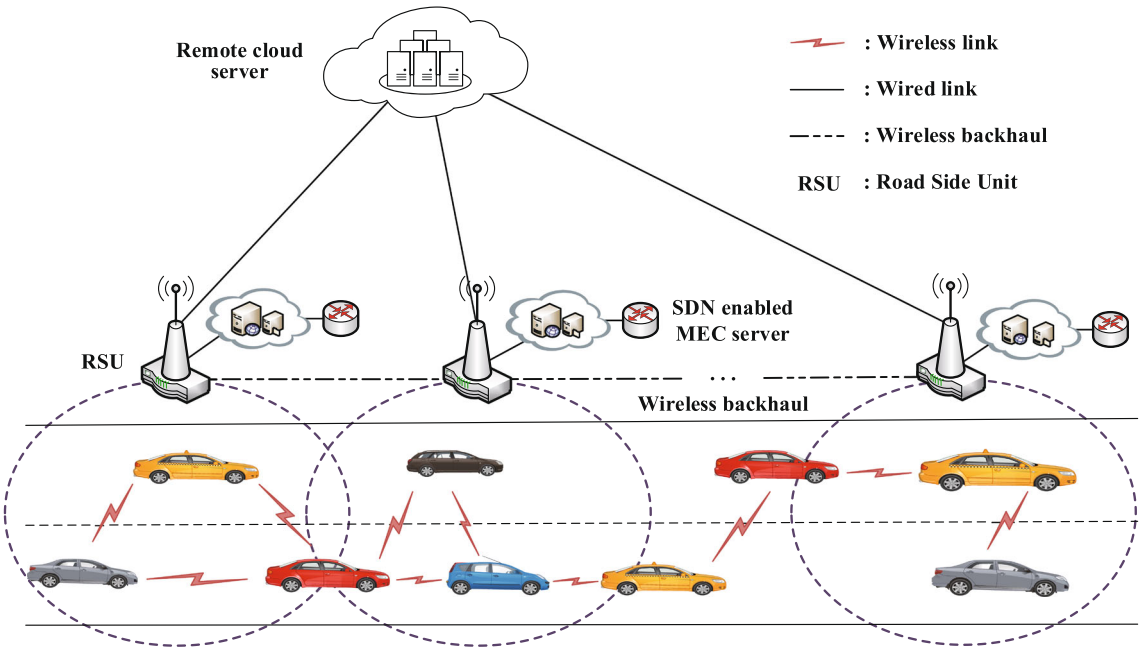

Fig. 1 SDN enabled vehicular edge computing network architecture

- The speeds of the vehicles follow Gaussian speed distribution, i.e., $f_s(v)\ N(\mu_s, \phi_s^2), s \in \{1, 2\}$, where s, μ_s and ϕ_s^2 are the traffic streams, mean speed and variance, respectively.
- We divide time into equally spaced time slots τ. Meanwhile, at the beginning of each time slot, vehicles are independently randomized to their speed, and there is no correlation between the speeds of the vehicles. This assumption is very suitable for free-flow traffic, i.e., the highway scenario in this paper, due to the generally low traffic density of highways [30].

Based on the above assumptions, all the vehicles on the road follow a homogeneous Poisson spatial distribution according to the superposition property of Poisson processes at any time instant.

The characteristics of each vehicle can be described by three variables: the traffic stream index of the vehicle s_i, the initial position of the vehicle p_i, and the speed of the vehicle μ_i. In addition, we assume that only one computation task needs to be accomplished for each vehicle within a period of time, and its characteristics can be described by four variables: the input data size d_i^{in}, the output data size d_i^{out}, the required CPU cycles to accomplish the task c_i, and the maximum permissible processing delay of the task t_i^{max}. So we denote the task of vehicle i as $T_i = \{d_i^{in}, d_i^{out}, c_i, t_i^{max}\}$.

Moreover, M road side units (RSUs) are placed next to the road, and each RSU covers a certain range. The main roles of RSUs are to enable vehicles to communicate with the Internet, and to provide edge computing services by equipping MEC servers on them. The computation capacities of the MEC servers can be denoted by $\mathscr{F}^{\mathscr{MEC}} = \{f_1^{MEC}, f_2^{MEC}, ..., f_M^{MEC}\}$. It is worth noting that the MEC servers we deployed on the RSUs are SDN enabled, which allows us to manage the entire networks centrally. The reason for this deployment is the SDN's features of programmability, separation of control plane and data plane, as well as the centralized control model with network transient state management. Meanwhile, the remote cloud server thousands of miles away acts as a backup server to provide more computation resources for the vehicles, while neither the MEC servers nor the vehicles themselves guarantee the processing delay requirements of the computation tasks. In our scenario, there are two modes for the data transmission, i.e., vehicle-to-vehicle (V2V) communications and vehicle-to-infrastructure (V2I) communications. We denote the communications mode decision of vehicle-i as $g_i \in \{0, 1\}$, where $g_i = 0$ means the decision of is vehicle-i to transmit its data through V2I communications, and $g_i = 1$ means the choice of vehicle-i to is transmit its data through V2V communications.

According to the foregoing, there are $M + 2$ task offloading decisions for each vehicle, i.e., executing its computation task locally on the vehicle's CPU, offloading its task to the MEC server connected to RSU-j, and offloading its task to the remote cloud server. The task offloading decisions of vehicle-i can be denoted as $r_i \in \{0, -1, 1, 2, 3, ..., M\}$, where $r_i = 0$ means that vehicle-i decides to execute its computation task locally on its own CPU, $r_i = -1$ is that vehicle i decides to offload its computation task to the remote cloud server, and $r_i = j\ (1 \le j \le M)$ means that vehicle i decides to offload its computation task to the MEC server connected to RSU-j. After that, for different task offloading decisions, the specific system models are given by the following subsections.

3.1 Executing locally

While $r_i = 0$, vehicle-i decides to accomplish its computation task locally on its CPU. In such a case, the processing time of the computation task is only related to the data size of the task and the computation capacity of vehicle-i, which can be calculated by the following formula:

$$t_i^{Local} = c_i / f_i^{Local}, \tag{1}$$

where f_i^{Local} is the computation capacity of vehicle-i.

3.2 Offloading to the MEC servers connected to the RSUs

If vehicle-i decides to offload its computation task to the MEC server connected to RSU-j, i.e., $r_i = j\ (1 \le j \le M)$, the processing time of the computation task mainly consist of the following parts: the execution time of the task on the MEC server, and the data transmission time between the vehicles and the MEC servers. Then the processing time in such case can be given as

$$t_i^{MEC} = c_i / f_j^{MEC} + t_i^{trans}. \tag{2}$$

Here, the first item is the execution time of the computation task on the MEC server, f_j^{MEC} is the computation capacity of the MEC server connected to RSU-j, and t_i^{trans} is the total transmitting time of both the input data and output data.

For t_i^{trans}, since the vehicle-i decides to offload its computation task to the MEC server connected to RSU-j via V2I communications (i.e., $g_i = 0$), its value can be given by

$$t_{i,j}^{V2I} = d_i^{in} / r_{i,j}^{V2I} + d_i^{out} / r_{j,i}^{V2I} + t', \tag{3}$$

where t' is the data transmitting time between RSUs via the wireless backhaul [31]. $r_{i,j}^{V2I}$ is the data transmitting rate between vehicle i and RSU j, and it can be calculated as

$$r_{i,j}^{V2I} = \omega_1 \cdot log_2\left(1 + \frac{p_{i,j} \cdot g_{i,j}^{V2I}}{\sigma^2 + I_{i,j}}\right). \quad (4)$$

Here, ω_1 is the channel bandwidth between vehicle i and RSU j, $p_{i,j}$ is the transmit power of vehicle i, $g_{i,j}^{V2I}$ is the channel gain between vehicle i and RSU j, σ^2 is the background noise power, and $I_{i,j}$ denotes the interference at RSU j. On the other hand, while vehicle-i decides to offload its computation task to the MEC server connected to RSU-j via V2V communications ($g_i = 1$), its value can be calculated by

$$t_{i,j}^{V2V} = d_i^{in}/r_{i,k}^{V2V} + d_i^{in}/r_{k,j}^{V2I} + d_i^{out}/r_{j,l}^{V2I} + d_i^{out}/r_{l,i}^{V2V} \quad (5)$$

where $r_{i,k}^{V2V}$ is the data transmitting rate between vehicle-i and vehicle-k, which can be calculated as

$$r_{i,k}^{V2V} = \omega_2 \cdot log_2\left(1 + \frac{p_{i,k} \cdot g_{i,k}^{V2V}}{\sigma^2 + A_0 \cdot L_{i,k}^{-2}}\right). \quad (6)$$

Here, ω_2 and $g_{i,k}^{V2V}$ are the channel bandwidth and channel gain between vehicle i and vehicle k respectively, σ^2 is the background power noise, A_0 is a constant parameter, and $L_{i,k}$ is the distance between vehicle i and vehicle k.

Therefore,the total transmitting time t_i^{trans} can be given as

$$t_{i,j}^{trans} = \begin{cases} t_{i,j}^{V2I}, & if\ g_i = 0, \\ t_{i,j}^{V2V}, & if\ g_i = 1. \end{cases} \quad (7)$$

3.3 Offloading to the remote cloud server

If $r_i = -1$, the vehicle-i's computation task will be transferred to the remote cloud server for execution. In such case, the processing time of the computation task is mainly composed of the following parts: the transmitting time of the input data from vehicle-i to relay RSU-j, the transmitting time of the input data from relay RSU-j to the remote cloud server, the transmitting time of the output data from the remote cloud server to relay RSU-k, and the transmitting time of the output data from relay RSU-k to vehicle i. Note that different relay RSUs may be used during the data transfer process of the computation task. Also, we ignore the execution time of the computation task, due to the powerful computation capacity of the remote cloud server. After that, the processing time of the task can be calculated as

$$t_i^{RCS} = d_i^{in}/r_{i,j}^{V2I} + (d_i^{in} + d_i^{out})/c' + d_i^{out}/r_{k,i}^{V2I}, \quad (8)$$

where c' is the data transmission rate between the RSUs and the remote cloud server.

According to the aforementioned system model, the processing time of vehicle-i's computation task can be given as

$$t_i(\lambda_i) = \begin{cases} t_i^{Local}, & if\ r_i = 0, \\ t_{i,j}^{MEC}, & if\ r_i = j,\ 1 \le j \le M, \\ t_i^{RCS} & if\ r_i = -1. \end{cases} \quad (9)$$

4 Problem formulation and solution

In this section, we describe the definition of the task offloading problem, and present corresponding solution.

4.1 Problem formulation

In our SDN enabled vehicular edge computing network architecture, the SDN enabled MEC servers can collect the vehicles' location, speeds, computation capacities, as well as the information of the computation tasks. Also, the information on the MEC servers' computation capacities and the communication environment can be collected by the SDN. After that, the above information is sent to the SDN controller for centralized management of the whole network.

Furthermore, as discussed above, each vehicle can choose to execute its computation task on its CPU, offload to the MEC server connected to RSU-j ($1 \le j \le M$), or offload to the remote cloud server thousands of kilometers away, depending on the processing delay of its task in different ways. Therefore, there are $M + 2$ task offloading decisions for each vehicle, and different decisions will lead to different data transmission consumption. While the number of vehicles that choose to execute their tasks on a certain RSU is increased, the communication environment will deteriorate due to the wireless interference. Meanwhile, the density of the vehicles also affects the quality of V2V communications. Taking into account the above factors, the task offloading decisions of the vehicles are crucial for the improvement of system performance. In such a case, our objective is to obtain an optimal task offloading decision profile for all the vehicles, to achieve the minimum total processing delay. Specifically, the problem can be defined as follows:

Definition 1 task Offloading Optimization with Minimal processing Delay (OOMD) : Given the information of the vehicles, MEC servers, remote cloud server, and communications environment obtained by SDN, the OOMD problem is to find an optimal task offloading decisions profile for all the vehicles, so as to minimize the total processing delay of all the computation tasks.

After that, the OOMD problem can be formulated as

$$
\begin{aligned}
\mathbf{P1}: & \min_{\{r_i, g_i\}} \sum_{i=1}^{N} t_i(r_i, g_i) \\
s.t. \ & C1: t_i^{Local} \cdot I_{\{r_i=0\}} + t_i^{MEC} \cdot I_{\{r_i=j\}} + t_i^{RCS} \cdot I_{\{r_i=-1\}} \\
& \quad \leq t_i^{max}, \ \forall\, i \in \mathcal{N}, j \in \mathcal{M}, \\
& C2: \sum_{i=1}^{N} I_{\{r_i=j\}} \leq K, \forall\, i \in \mathcal{N}, j \in \mathcal{M}, \\
& C3: r_i \in \{0, -1, 1, 2, ..., M\}, \ \forall\, i \in \mathcal{N}, \\
& C3: g_i \in \{0, 1\}, \ \forall\, i \in \mathcal{N},
\end{aligned}
$$

where $I_{\{*\}}$ is an indicator function, and $I_{\{*\}} = 1$ while $*$ is true. K is the number constraint of the wireless channels in RSU j, and the value of g_i only affects the selection of the communications modes.

4.2 Solution

To solve the above OOMD problem, we need to obtain an optimal task offloading decision profile with the minimum total processing delay for all the vehicles. However, the OOMD problem is NP-hard since it can be transformed into the traditional maximum cardinality bin packing problem. Meanwhile, due to the highly dynamic nature of the vehicular networks, our above OOMD problem requires a dynamic time-varying solution, i.e., the system can adjust the offloading decisions of the vehicles' tasks according to the number of vehicles and the load status of MEC servers. Toward this end, we proposed a deep Q learning-based task offloading scheme as our solution.

The basis of our deep Q learning scheme is the Q learning algorithm, which is one of the reinforcement learning algorithms that have received much attention in recent years. Reinforcement learning is a learning method of machine learning that emphasizes how to act based on the environment, in order to maximize the expected benefits. Take the game as an example; if we adopt a strategy in the game to get a higher score, then we will further strengthen this strategy to achieve consistently better results. This pattern is very similar to the various "performance rewards" in our lives. The core elements of Q learning are environment state, action, and reward. For our scenario, they can be defined as follows:

- state $s_t = \sum_i t_i$: the total processing delay of all the computation tasks, related to the states of the vehicles, MEC servers, and wireless communications environment.
- action a_t: the set of vehicles' offloading decisions and communications decisions.
- reward $R_t = (T_{Local} - T(a_t))/T_{Local}$: related to the processing time T_{Local} while all vehicles decide to execute their tasks locally and the processing time $T(a_t)$ by taking action a_t.

In Q learning, different actions produce different rewards under particular state values. The values of states and actions are used as the rows and columns of a matrix called Q table, and the corresponding value is Q value $Q(s, a)$, which measures how good action a will be under state s.

Algorithm 1 Deep Q Learning based Task Offloading Algorithm (DQLTO)

Require: the initial environment status, i.e., the condition of the vehicles and MEC servers.
Ensure: a task offloading decisions profile for all vehicles and the total processing delay.

Initialize: main Q network with weight θ, target Q network with $\theta^* = \theta$, replay memory D;
for each episode **do**
  obtain state s_1 from the current environment;
  for $t = 1, 2, 3, ..., T$ **do**
    randomly select probability ρ;
    if $\rho \leq \epsilon$ **then**
      randomly select action $a_t = a$;
    else
      select action $a_t = arg\max_a Q(s, a, \theta)$, which means the largest Q-value;
    end if
    get the reward R_t and next state s_{t+1} that correspond to action a_t;
    store (s_t, a_t, R_t, s_{t+1}) in to replay memory D;
    randomly select a batch of samples from replay memory D;
    calculate the target Q value Q^* according to 11;
    update main Q network by minimizing the loss in 12;
    every G steps, update target Q network according to 13;
  end for
end for

However, the representation and access of the Q table lead to large consumption of storage and computation resources, which results in a reduction in system efficiency. Toward this end, we introduce deep neural networks to estimate Q value, i.e., deep Q learning algorithm. In such algorithm, there are two layers of networks for different purposes: main Q network with the weight θ for obtaining the current Q value $Q(s, a, \theta)$, and target Q network with the weight θ^* for getting target Q value $Q^*(s, a, \theta^*)$. Moreover, there is a replay memory D for for storing the obtained samples (s_t, a_t, R_t, s_{t+1}). At the

beginning of the DQLTO algorithm, the main Q network and target Q network are initialized with parameters θ and θ^*, respectively. Then, we select action a_t according to probability ρ in each cycle t, and the selection of the action follows the following equation:

$$a_t(\rho) = \begin{cases} a^-, & if\ \rho \leq \epsilon, \\ \max\limits_{a} Q(s, a, \theta), & if\ \rho > \epsilon, \end{cases} \tag{10}$$

where a^- is a randomly selected value. After that, the corresponding sample (s_t, a_t, R_t, s_{t+1}) is stored to the replay memory. Finally, we calculate the target Q value as:

$$Q_i^* = R_i + \epsilon Q(s_{i+1}, arg\max_{a^*} Q(s_{i+1}, a^*; \theta); \theta^*), \tag{11}$$

and update the main Q network by minimizing the loss:

$$L_i(\theta) = \frac{1}{U} \cdot \sum [Q_i^* - Q(s_i, a_i; \theta)]^2. \tag{12}$$

After repeating the above steps G times, the weight of the target Q network is updated according to the following equation.

$$\theta^* = \alpha\theta + (1 - \theta)\theta^*, \tag{13}$$

The details of the DQLTO algorithm are shown in Algorithm 1.

5 Performance evaluation

In order to verify the feasibility and performance of our proposed scheme, we present some simulation experiments in this section. To make the simulation close to reality, we consider a road with two traffic streams, on which the vehicles follow homogeneous Poisson spatial distribution, and their speeds follow Gaussian speed distribution. For each vehicle, only one computation task needs to be accomplished, whose input data size and output data size are randomly chosen from [500 KB, 1000 KB] and [50KB, 300KB], respectively, and its maximum permissible processing delay is between 0.5 s and 2 s. Meanwhile, the range of CPU cycles required to complete a task is [200 Mc, 1500Mc], and the CPU cycle frequency of MEC servers and vehicles are set to 4 GHz and 2 GHz, respectively. Furthermore, the wireless bandwidth of vehicles and RSUs are 20 MHz and 40 MHz, and the background noise power is -100 dBm. Table 1 lists the parameter settings in our simulation experiments.

First, we compared the total processing delay of all the vehicles' computation tasks by adopting three different schemes, to verify that our proposed DQLTO scheme is valid for solving the OOMD problem, i.e., an optimal or suboptimal solution can be obtained. The first two schemes we adopted are the enumeration algorithm that can obtain the optimal solution and the game theory-based algorithm [7]. Note that game theory is a technology that is widely used to solve the decision-making problems in vehicular networks and shows great applicability. Fig. 2 shows the total processing delay obtained by these two schemes and our DQLTO scheme. From the figure, we can see that our DQLTO scheme can achieve results approximate to the game theory-based scheme, which shows that our DQLTO scheme can effectively solve the OOMD problem. Also, we compared the running time of these three schemes, and the corresponding results are shown in Fig. 3. One undeniable fact is that the overall running time of our DQLTO algorithm is much less than the enumeration algorithm, slightly more than the game theory-based algorithm, and does not grow sharply as the number of vehicles increases. The above

Table 1 Parameter settings

notation	description	value
d_i^{in}	input data size	[500 KB, 1000 KB]
d_i^{out}	output data size	[50KB, 300KB]
t_i^{max}	maximum permissible processing delay	[0.5 s, 2 s]
c_i	CPU cycles required to complete a task	[200 Mc, 1500 Mc]
f_j^{MEC}	CPU cycle frequency of MEC servers	4 GHz
f_j^{Local}	CPU cycle frequency of vehicles	2 GHz
ω_1	wireless bandwidth of vehicles	20 MHz
ω_2	wireless bandwidth of RSUs	40 MHz
σ	background noise power	-100 dBm

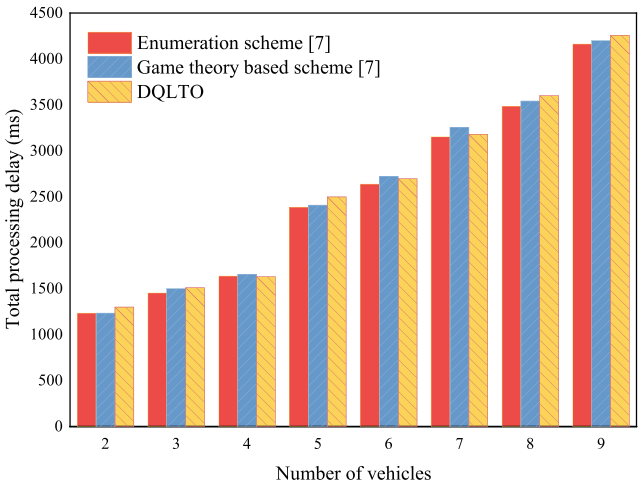

Fig. 2 Comparisons of processing delay of all the vehicles' computation tasks with three different offloading schemes, i.e., enumeration scheme, game theory based scheme, and our DQLTO scheme

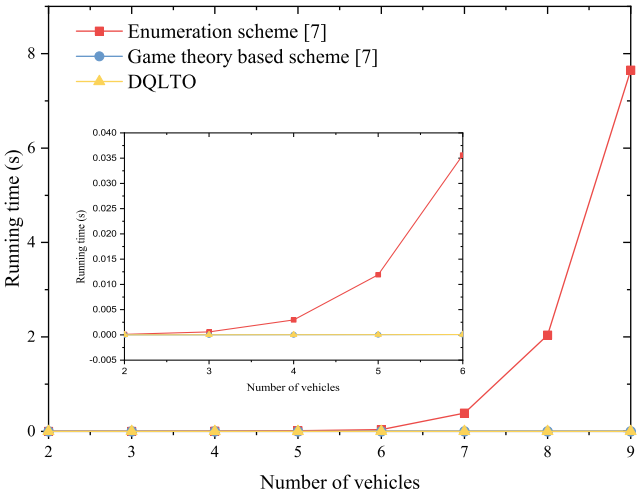

Fig. 3 Comparisons of the running time with three different offloading schemes, i.e., enumeration scheme, game theory-based scheme, and our DQLTO scheme (same coordinate titles for both small and large figure)

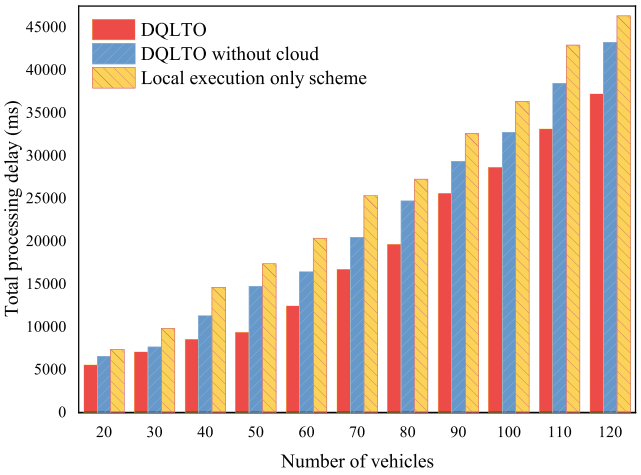

Fig. 4 Comparisons of the total processing delay with three different offloading schemes, i.e., DQLTO scheme, DQLTO scheme without the cloud, and local execution only scheme

experimental results undoubtedly validate that our DQLTO scheme is extremely efficient.

After that, we set up an experiment in order to verify the role of MEC servers and remote cloud servers in reducing the processing delay of vehicles' computation tasks. Accordingly, a comparison of the total processing delay obtained by three different offloading schemes (i.e., DQLTO scheme, DQLTO scheme without the cloud, and local execution only scheme) is shown in Fig. 4. From the figure, we can see that the processing delay obtained by the DQLTO scheme and DQLTO scheme without cloud are smaller than those obtained by local execution only scheme, and the results show that MECO has a significant impact on reducing the processing time of vehicles' tasks. Furthermore, the processing delay obtained by the DQLTO scheme are smaller than that obtained by the DQLTO scheme without the cloud, so it is necessary to introduce the remote cloud server as a backup server for MECO.

In order to investigate the impact of the computation capacities of the MEC servers on the tasks' offloading decision profile, we studied the total processing delay at different CPU cycle frequency of MEC servers, and the corresponding experiment results are shown in Fig. 5. From the figure, one observed phenomenon is that the computation capacities of the MEC servers have a decisive influence on the offloading decisions of the vehicles' computation tasks. Specifically, the total processing time of the tasks did not change with the CPU cycle frequency of the MEC servers, while we adopted local execution only scheme. However, the total processing delay of the DQLTO scheme and DQLTO scheme without cloud decrease with the increasing CPU cycle frequency of the MEC servers. The reason for this phenomenon is that the vehicles prefer to offloading their computation tasks to MEC servers or remote cloud servers to reduce the processing delay, as the computation capacities of the MEC servers increase. Also, we can see that the rate of decrease in the total processing time increases first and then decreases because the impact of the MEC servers on the tasks' offloading decisions is reduced while their computation capacities are large enough.

Moreover, we studied the selection of the communications modes (i.e., V2I communications and V2V communications) in the task offloading process. The numerical results in Fig. 6 show the comparisons of the number of V2I or V2V communications selection, as the number of vehicles changes from 20 to 120. We can see that both the number of V2I communications selection and V2V communications

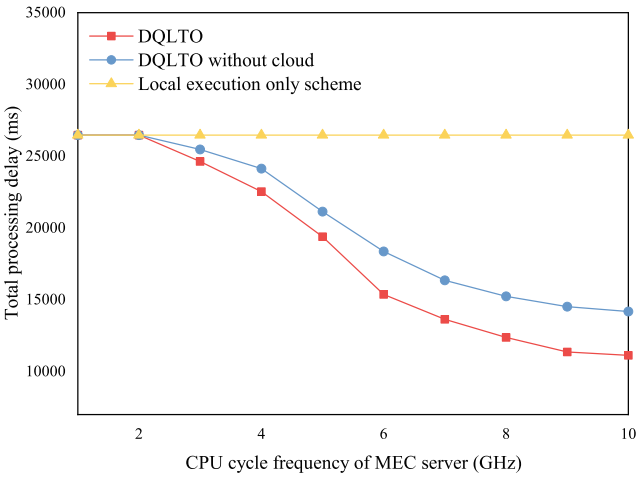

Fig. 5 Comparisons of the total processing time obtained by three schemes, i.e., DQLTO scheme, DQLTO scheme without the cloud, and local execution only scheme, as the computation capacity of the MEC servers changes

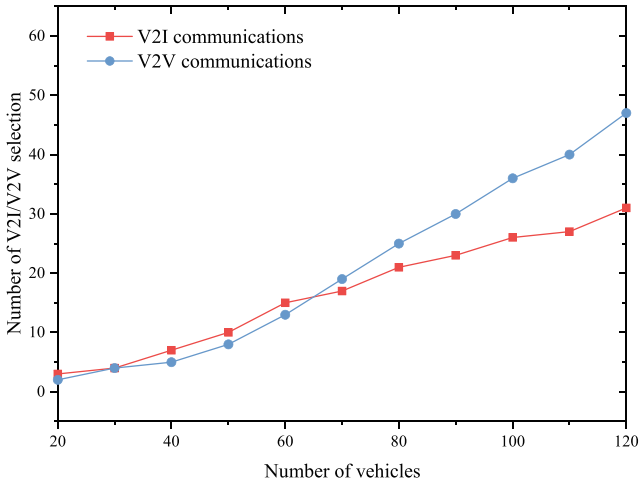

Fig. 6 Comparisons of the number of V2I and V2V communications selection, as the number of vehicles changes from 20 to 120

selection increase with increasing vehicles. Meanwhile, the growth rate of the V2V communications selection is more significant than that of the V2I communications selection, which indicates that the number of vehicles has a more significant influence on V2V communications.

6 Conclusions

In this paper, we studied the task offloading decision optimization problem in VECNs with the goal of minimizing the total processing delay of all the vehicles' tasks. In our scenario, we consider a synchronized random walk model to simulate real traffic conditions. To solve the problem, a reinforcement learning-based task offloading scheme is proposed. Final numerical results corroborated the superior performance of our algorithm in processing delay reduction and dynamic scene adaptability. For future work, it should be meaningful to study the mobile edge computing offloading problems in a complex urban vehicular networking environment.

Acknowledgments This work was supported by National Natural Science Foundation of China (61771374, 61771373, 61801360, and 61601357), in part by Natural Science Basic Research Program of Shaanxi (2020JC-15 and 2020JM-109), in part by Fundamental Research Funds for the Central Universities (3102019PY005, 31020190QD040, and 31020200QD010), in part by Special Funds for Central Universities Construction of World-Class Universities (Disciplines) and Special Development Guidance (06390-20GH020114), and in part by China 111 Project (B16037).

References

1. Al-fuqaha A, Guizani M, Mohammadi M et al (2015) Internet of things: a survey on enabling technologies, protocols, and applications. IEEE Commun Surv Tutorials 17(4):2347–2376. https://doi.org/https://doi.org/10.1109/COMST.2015.2444095
2. Siegel JE, Erb DC, Sarma SE (2018) A survey of the connected vehicle Landscape-Architectures, enabling technologies, applications, and development areas. IEEE Trans Intell Transp Syst 19(8):2391–2406. https://doi.org/10.1109/TITS.2017.2749459
3. Khelifi H, Luo S, Nour B et al (2019) Named Data Networking in Vehicular Ad hoc Networks: State-of-the-Art and Challenges
4. Zhang J, Wang F, Wang K et al (2011) Data-Driven Intelligent transportation systems: a survey. IEEE Trans Intell Transp Syst 12:1624–1639. https://doi.org/10.1109/TITS.2011.2158001
5. Han S, Huang Y, Meng W et al (2019) Optimal power allocation for SCMA downlink systems based on maximum capacity. IEEE Trans Commun 67(2):1480–1489. https://doi.org/10.1109/TCOMM.2018.2877671
6. Zhang K, Mao Y, Leng S et al (2016) Delay constrained offloading for mobile edge computing in Cloud-Enabled vehicular networks. In: Proc 2016 8th Int Work Resilient Networks Des Model RNDM, vol 2016, pp 288–294, https://doi.org/10.1109/RNDM.2016.7608300
7. Guo H, Zhang J, Liu J (2019) Fiwi-enhanced Vehicular Edge Computing networks: Collaborative Task Offloading. IEEE Veh Technol Mag 14(1):45–53. https://doi.org/10.1109/MVT.2018.2879537
8. Bisio I, Garibotto C, Lavagetto F et al (2019) Blind detection: Advanced Techniques for WiFi-Based Drone Surveillance. IEEE Trans Veh Technol 68(1):938–946. https://doi.org/10.1109/TVT.2018.2884767
9. Sun F, Hou F, Cheng N et al (2018) Cooperative task scheduling for computation offloading in vehicular cloud. IEEE Trans Veh Technol 67(11):11049–11061. https://doi.org/10.1109/TVT.2018.2868013
10. Guo H, Liu J, Zhang J (2018) Computation offloading for Multi-Access mobile edge computing in Ultra-Dense networks. IEEE Commun Mag 56(8):14–19. https://doi.org/10.1109/MCOM.2018.1701069
11. Guo H, Liu J, Zhang J (2018) Mobile-edge Computation Offloading for Ultradense IoT Networks. IEEE Internet Things J 5(6):4977–4988. https://doi.org/10.1109/JIOT.2018.2838584
12. Mach P, Becvar Z (2017) Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun Surv Tutorials 19(3):1628–1656. https://doi.org/10.1109/COMST.2017.2682318
13. Han S, Li Y, Meng W et al (2019) Indoor localization with a single Wi-Fi access point based on OFDM-MIMO. IEEE Syst J 13(1):964–972. https://doi.org/10.1109/JSYST.2018.2823358
14. Qiao G, Leng S, Zhang K, He Y (2018) Collaborative task offloading in vehicular edge Multi-Access networks. IEEE Commun Mag 56(8):48–54. https://doi.org/10.1109/MCOM.2018.1701130
15. Liu Y, Wang S, Huang J, Yang F (2018) A computation offloading algorithm based on game theory for vehicular edge networks
16. Du J, Yu FR, Chu X et al (2019) Computation offloading and resource allocation in vehicular networks based on Dual-Side cost minimization. IEEE Trans Veh Technol 68(2):1079–1092. https://doi.org/https://doi.org/10.1109/TVT.2018.2883156
17. Wang K, Yin H, Quan W, Min G (2018) Enabling collaborative edge computing for software defined vehicular networks. IEEE Netw 32(5):112–117. https://doi.org/10.1109/MNET.2018.1700364
18. Zhao J, Li Q, Gong Y, Zhang K (2019) Computation offloading and resource allocation for cloud assisted mobile edge computing. IEEE Trans Veh Technol 68(8):7944–7956. https://doi.org/https://doi.org/10.1109/TVT.2019.2917890
19. Serra J, Sanabria-russo L, Pubill D, Verikoukis C (2018) Scalable and Flexible IoT Data Analytics: When Machine Learning Meets SDN and Virtualization. In: 2018 IEEE 23rd Int Work Comput Aided Model Des Commun Links Networks 1–6

20. Ye H, Liang L, Li GY et al (2018) Machine learning for vehicular networks: recent advances and application examples. IEEE Veh Technol Mag 13(2):94–101. https://doi.org/10.1109/MVT.2018.2811185
21. Taherkhani N, Pierre S (2016) Centralized and localized data congestion control strategy for vehicular ad hoc networks using a machine learning clustering algorithm. IEEE Trans Intell Transp Syst 17(11):3275–3285. https://doi.org/10.1109/TITS.2016.2546555
22. Zhang Z, Mao G, Member S et al (2014) Stochastic Characterization of Information Propagation Process in Vehicular Ad hoc Networks. IEEE Trans Intell Transp Syst 15(1):122–135. https://doi.org/https://doi.org/10.1109/TITS.2013.2274274
23. Zarei M, Rahmani AM (2017) Analysis of vehicular mobility in a dynamic Free-Flow highway. Veh Commun 7:51–57. https://doi.org/https://doi.org/10.1016/j.vehcom.2016.12.001
24. Wang S, Zhang X, Zhang Y et al (2017) A survey on mobile edge networks: convergence of computing, caching and communications. IEEE Access 5:6757–6779. https://doi.org/https://doi.org/10.1109/ACCESS.2017.2685434
25. Ning Z, Dong P, Kong X, Xia F (2019) A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things. IEEE Internet Things J 6(3):4804–4814. https://doi.org/https://doi.org/10.1109/JIOT.2018.2868616
26. You C, Huang K, Chae H, Kim BH (2017) Energy-Efficient Resource allocation for Mobile-Edge computation offloading. IEEE Trans Wirel Commun 16(3):1397–1411. https://doi.org/https://doi.org/10.1109/TWC.2016.2633522
27. Guo H, Liu J (2018) Collaborative computation offloading for multiaccess edge computing over Fiber-Wireless networks. IEEE Trans Veh Technol 67(5):4514–4526. https://doi.org/10.1109/TVT.2018.2790421
28. Sun Y, Member S, Guo X, Song J (2019) Adaptive Learning-Based task offloading for vehicular edge computing systems. IEEE Trans Veh Technol 68(4):3061–3074. https://doi.org/https://doi.org/10.1109/TVT.2019.2895593
29. Cao Y, Zhang L, Liao J (2019) Knowledge-Driven Service offloading decision for vehicular edge computing: a deep reinforcement learning approach. IEEE Trans Veh Technol 68(5):4192–4203. https://doi.org/10.1109/TVT.2019.2894437
30. Yousefi S, Altman E, El-Azouzi R, Fathy M (2008) Analytical model for connectivity in vehicular Ad Hoc networks. IEEE Trans Veh Technol 57(6):3341–3356. https://doi.org/10.1109/TVT.2008.2002957
31. Han S, Zhang Y, Meng W et al (2019) Full-Duplex Relay-Assisted Macrocell with millimeter wave backhauls: framework and prospects. IEEE Netw 33(5):190–197. https://doi.org/10.1109/MNET.2019.1800305