Collaborative Edge Computing and Caching in Vehicular Networks

Zhuoxing Qin^1, Supeng Leng^{1,*}, Jihu Zhou^2, Sun Mao^1
^{1}School of Information & Communication Engineering,
University of Electronic Science and Technology of China
^{2}Chongqing Jinmei Communication CO. LTD
*Corresponding author, email: spleng@uestc.edu.cn

***Abstract*—Mobile Edge Computing (MEC) can significantly promote the development of Internet of Vehicles (IoV) for providing a low-latency and high-reliability environment. Nevertheless, a huge amount of sensor data or computing requirements generated by massive vehicles in adjacent area may be duplicated. In order to realize the efficient diffusion of information, we propose a hierarchical end-edge framework with the aid of deep collaboration among data communication, computation offloading and content caching to minimize network overheads. Specially, duplicated perceived data and computation results are cached in advance to decrease repeated data uploading and duplicated computation in offloading process. In addition, the problem is formulated as a mixed integer non-linear programming (MINLP) problem, and the deep deterministic policy gradient (DDPG)-based resource allocation scheme is utilized to obtain a sub-optimal solution with low computation complexity. Performance evaluation demonstrates that the proposed scheme can significantly reduce network overheads compared with other benchmark methods.**

***Index Terms*—Mobile edge computing, content caching, deep deterministic policy gradient**

I. Introduction

In recent years, the construction of smart transportation system has been the trend to improve the road efficiency. A vast amount of perceived data (about 1 GBytes/s) will be generated by these autonomous vehicles [1], requiring vehicles with considerable memory size and computation resources. In order to alleviate the computational burden of vehicles, vehicular edge computation networks (VECNs) have been proposed as a promising computation paradigm to enable vehicles to offload their computation-intensive tasks to edge nodes such as roadside unit (RSU).

When we consider a general scenario with multiple vehicles in the same physical proximity, the perceived data such as current road condition and traffic accidents may be duplicated and thus will result in the repeated transmission. Meanwhile, some computation results of RSU may be valuable to other vehicles, such as the global view of the current road. If these contents are not handled properly, duplicated computation will incur severe network congestion and resources waste in VECNs.

Most of existing works focused on either offloading decisions or caching policy, separately [2]–[6]. In order to improve the resource utilization ratio, there are also some works focusing on studying the collaborative offloading and caching scheme in edge computing networks [7]–[10]. The authors in [7] mainly studied transcoding/computing and caching strategy of requested video for VECNs, which will significantly save backhaul bandwidth resources and reduce execution delay of tasks. Aiming to help construct a smart transportation system, the authors in [8] proposed a two-timescale edge computation and caching resources allocation strategy in vehicular network, considering the mobility of vehicle and the switching across different RSUs. In order to satisfy the increasing demand for data traffic, the authors in [9] studied delay-tolerant data caching and tasks computing in software-defined vehicular networks. The authors in [10] developed a collaborative offloading strategy based on coalitional game for mobile wireless networks with caching enhancement, considering offloaded tasks may be similar in the same area among mobile users.

However, these existing works mainly focused on the issue about duplicated communication, especially for data transmission in backhaul link. For multiple connected vehicles in the near area, the fronthaul link congestion caused by repeated uploading of perceived data and the issue of duplicated computing of offloaded tasks have not been well addressed. Thus, it is essential to investigate the cooperative task offloading and caching strategy for enhancing the efficiency of data delivery and resource utilization.

In this paper, we make a bold step in designing, analyzing and optimizing the cooperative computing and caching strategy by considering the constraints of limited bandwidth, storage space and computation resources of RSUs. Note that contents in our proposed caching scheme consists of input perceived data and output results and they are derived from the offloaded tasks of driving vehicles while in traditional caching paradigm, contents are from Internet Service Provider (ISP). The major contributions of our paper are summarized as follows:

- We propose a collaborative bandwidth allocation, computing and caching strategy to minimize overall network overheads. Specially, the caching strategy for input data

This work was partly supported by National Key R&D Program of China (No.2018YFE0117500), the Science and Technology Program of Sichuan Province, China (No.2019YFH0007), and the EU H2020 Project COSAFE (MSCA-RISE-2018-824019).

978-1-7281-3106-1/20/$31.00 ©2020 IEEE

and computing results is aided to address duplicated data uploading and computation problems in offloading process, which have not been considered in previous works.

- Different from conventional content updating policy in a large timescale, our proposed content caching strategy can be dynamically adjusted according to the network states and the mobility of vehicles.
- We formulate the joint optimization problem as a mixed integer non-linear programming (MINLP) problem. Based on deep deterministic policy gradient (DDPG) technique, we develop a multi-dimensional resources allocation strategy to obtain a sub-optimal solution of the MINLP problem with low computation complexity.

The rest of this paper is organized as follows. In Section II, we introduce a system model in vehicular edge computing networks. The optimization problem is formulated in Section III. The joint bandwidth allocation, task offloading and content updating scheme based on DDPG algorithm is elaborated in Section IV. The performance evaluation is demonstrated in Section V. Finally, Section VI concludes the whole paper.

II. System Model

A typical vehicular edge computing network is considered in the proposed scheme, which is composed of one BS (base station) and N RSUs with limited bandwidth, computation resources and storage units. Let $\mathcal{N} = \{1, 2, ..., N\}$ be the index set of RSUs. In addition, $\mathcal{M} = \{1, 2, ..., M\}$ is denoted as the index set of vehicles. And the total number of tasks is assumed as K, and $\mathcal{K} = \{1, 2, ..., K\}$ is the index set of different types of tasks. Besides, a parameter tuple $< \mathrm{I}^k, \mathrm{O}^k, \Phi^k, \mathrm{B}^k >$, $k \in \mathcal{K}$ is used to describe the characteristics of the kth task generated by vehicles. I^k is the input size of task k, O^k is the output size of task k, Φ^k is the computation resources required by task k, and B^k is the minimum computation rate.

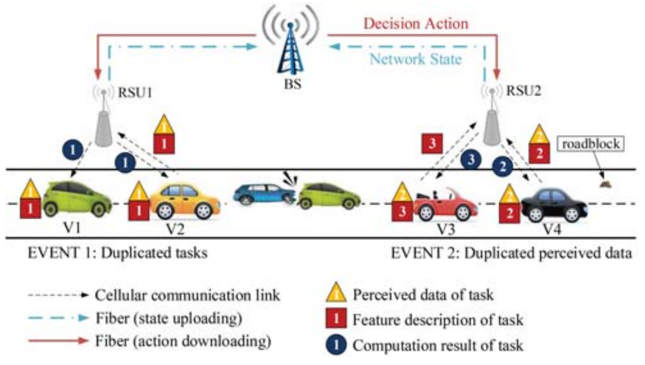
Decision Action
Network State
BS
RSU1
RSU2
roadblock
V1
V2
V3
V4
EVENT 1: Duplicated tasks
EVENT 2: Duplicated perceived data
Cellular communication link
Fiber (state uploading)
Fiber (action downloading)
Perceived data of task
Feature description of task
Computation result of task

Fig. 1. Illustration of collaborative computation and caching scheme.

We assume that each vehicle is willing to send main attributions of offloaded task to RSU and each RSU sends these messages and network states to BS at the beginning of time slot. Then, BS makes decisions according to whole network system states.

Fig. 1 illustrates a typical network scenario of the collaborative computation offloading and content caching. For example, vehicle V1 and V2 may have similar vision about the crash in the same transportation area. The computation result of offloaded task T1 of V2 is valuable for V1 and is decided to be cached in RSU1 considering the redundancy of tasks. V1 with the same task directly receives the result of task T1, which can reduce duplicated computation in RSU. For another important situation, we assume that task T2 of V4 is about speed changing and T3 of V3 is about lanes changing. However, V3 and V4 have similar perceived data about the road such as the roadblock. Thus, the same perceived data of task T2 and T3 can be uploaded one time to RSU2 and is shared between T2 and T3 through being cached in RSU2, which can decrease duplicated data transmission. According to the observations of typical events, a joint computation offloading and caching scheme is necessitated to realize efficient information sharing and improve driving safety.

A. *Task Transmission Model*

Without loss of generality, the vehicular network topology can be regarded as stable during resources allocation period. In the ultra-dense network deployment, orthogonal frequency division multiple access (OFDMA) is applied as a data transmission mode in which M vehicles within the same RSU are separated in the frequency domain. Therefore, there is no interference among vehicles which attach to the same RSU. Let p_m and p_n denote transmission power for the vehicle m ($m \in \mathcal{M}$) and RSU n ($n \in \mathcal{N}$), respectively. The maximum achievable data transmission rate over an additive white Gaussian noise (AWGN) channel from vehicle m to RSU n can be formulated as

$$r_{m,n} = w\log_2\left(1 + \frac{p_m g_{ul}}{\xi_{ul} d^\kappa \sigma^2}\right), \tag{1}$$

where w is the allocated bandwidth, d is the distance between vehicle to RSU, κ is path loss exponent and σ^2 is noise power. In addition, g_{ul} and ξ_{ul} are channel gain and path loss for uplink, respectively.

Similarly, the maximum achievable data transmission rate from RSU n to vehicle m is given by:

$$r_{n,m} = w\log_2\left(1 + \frac{p_n g_{dl}}{\xi_{dl} d^\kappa \sigma^2}\right). \tag{2}$$

B. *Task Execution Model*

In the beginning of task execution period, offloading decisions are made and the task execution is mainly composed of local execution and edge execution.

1) Local Execution: If task k of vehicle m is executed locally, the completion time can be expressed as $t_m^k = \frac{\Phi_m^k}{f_m}$, where f_m denotes CPU rate of vehicle m, and the computation rate is given by:

$$R_m^k = \frac{\mathrm{I}_m^k}{t_m^k} = \frac{\mathrm{I}_m^k f_m}{\Phi_m^k} \geq \mathrm{B}_m^k. \tag{3}$$

2) Edge Execution: If task k of vehicle m is attached to RSU n, total latency of task execution may consist of transmission latency of input data, task computation latency, and transmission latency of output data. Specifically, $t\_s_{m,n}^k = \frac{\mathrm{I}_m^k}{r_{m,n}}$ is task transmission latency from vehicle m to RSU n, $t\_c_{m,n}^k = \frac{\Phi_m^k}{f_n}$ is task computation latency at edge RSU n, where f_n denotes CPU rate of RSU n, and $t\_r_{m,n}^k = \frac{\mathrm{O}_m^k}{r_{n,m}}$ is transmission latency of receiving output data, respectively. Then, the minimum computation rate and total task execution time are expressed as follows:

$$R_{m,n}^k = \frac{\mathrm{I}_m^k}{t\_c_{m,n}^k} = \frac{\mathrm{I}_m^k f_n}{\Phi_m^k} \geq \mathrm{B}_m^k, \tag{4}$$

$$t_{m,n}^k = t\_s_{m,n}^k + t\_c_{m,n}^k + t\_r_{m,n}^k. \tag{5}$$

Note that the objective of content updating is to decrease duplicated data transmission and save computation resources of RSUs for redundant tasks processing, and thus the best content updating policy is not only to cache more popular contents but also to cache contents of those tasks which need higher CPU resource or have massive perceived data. In addition, each vehicle requests to offload task randomly and independently, and the offloading request rate of tasks is time-varying generally. This motivates us to propose a leaning-based caching strategy to improve the utilization of resources and adapt to the dynamic vehicular network environment.

III. Problem Formulation

In this section, a multi-objective optimization problem is presented from the perspective of total task execution time and bandwidth costs and this problem is formulated as a mixed integer non-linear programming problem.

Four action variables and one indicator variable are defined based on the aforementioned definition of system model, where $\alpha_{m,n}^k$, $x_n^{'k}$, $x_n^{''k}$and $\zeta_{m,n}^k$ are action variables, and φ_n^k is a binary indicator variable. More specifically, $\alpha_{m,n}^k$ is a binary variable of attaching decision. $\alpha_{m,n}^k = 1$ represents that vehicle m is attached to RSU n and RSU n will deliver the computation result of task k, otherwise $\alpha_{m,n}^k = 0$ denotes that task k is processed at vehicle. Variable $x_n^{'k}$ is a binary variable of caching decision for input perceived data of task k at RSU n. $x_n^{'k} = 1$ means input data is decided to be cached in RSU n while $x_n^{'k} = 0$ means the data is not cached. And variable $x_n^{''k}$ is a binary variable of caching decision for computation result of task k at RSU n. The result of task k should be stored at RSU n if variable $x_n^{''k} = 1$, otherwise, the result of task k will not be cached at RSU n. Variable $\zeta_{m,n}^k$ is a continuous action variable which denotes the bid of vehicle m with task k for competing bandwidth resources of RSU n. In addition, variable φ_n^k describes the occupancy state of computation resources for processing the task k at RSU n. $\varphi_n^k = 1$ indicates that task k is being executed at RSU n, which means other vehicles who connect with RSU n currently will not offload the same task to RSU n. In this situation, the perceived data or computation results of task k can be shared among vehicles that own the task k and connect with RSU n. Otherwise, $\varphi_n^k = 0$ represents that task k is not being processed at RSU n.

TABLE I
Execution latency of different system states

Delay	$x_n^{''k}(t-1) = 0$	$x_n^{''k}(t-1) = 1$
$\varphi_n^k(t) = 0$	if $x_n^{'k}(t-1) = 0$, $t_{m,n}^k$; else, $t\_c_{m,n}^k + t\_r_{m,n}^k$	$t\_r_{m,n}^k$
$\varphi_n^k(t) = 1$	$t\_r_{m,n}^k$	

TABLE II
Updating operation of perceived data

Updating operation	$x_n^{'k}(t-1) = 0$	$x_n^{'k}(t-1) = 1$
$x_n^{'k}(t) = 0$	no caching	deleting
$x_n^{'k}(t) = 1$	caching	keeping caching

Task Execution Time: For task k of vehicle m, the total task execution time can be expressed as:

$$\begin{aligned} D_m^k(t) =& (1-\alpha_{m,n}^k(t)) \cdot t_m^k + \alpha_{m,n}^k(t) \cdot \{(1 - x_n^{''k}(t-1)) \\ & \cdot (1-\varphi_n^k(t)) \cdot [(1 - x_n^{'k}(t-1)) \cdot t\_s_{m,n}^k + t\_c_{m,n}^k] \\ & + t\_r_{m,n}^k\}, \end{aligned} \tag{6}$$

If attaching decision variable $\alpha_{m,n}^k = 1$, the detailed task execution time is illustrated as Table I, where $x_n^{''k}(t-1)$ denotes results caching decision at last time slot which is regarded as the current caching state of task k result at RSU n, and $\varphi_n^k(t)$ denotes current computation resources usage state of task k at RSU n. If the result of task k was not cached before (i.e., $x_n^{''k}(t-1) = 0$) and the same task k offloaded by other vehicles is not being executed currently (i.e., $\varphi_n^k(t) = 0$) at RSU n, there are two cases for the completion latency of task. The input data transmission delay can be ignored if the perceived data of task k was cached (i.e., $x_n^{'k}(t-1) = 1$), otherwise task k has to be offloaded to RSU n and total execution latency will be denoted as $t_{m,n}^k$. In other three situations, vehicle m only need to receive the computation result of task k from RSU and the completion time can be shortened greatly.

In addition, the current updating operation of contents (i.e., perceived data and computation result of tasks) at RSU n lies on caching decisions of two time slots, namely [$x_n^{'k}(t-1)$, $x_n^{'k}(t)$] and [$x_n^{''k}(t-1)$, $x_n^{''k}(t)$]. The updating operation of sensor data is defined in detail as Table II.

Due to strict latency constraints of tasks and limited bandwidth resources of RSUs, vehicles are necessitated to bid for bandwidth. Specifically, our bandwidth allocation scheme is based on an auction-based model. Thus, for vehicle m, the bid of bandwidth can be expressed as:

$$C_m^k(t) = \alpha_{m,n}^k(t) \cdot \zeta_{m,n}^k(t), \tag{7}$$

Moreover, the allocated bandwidth of RSU n for vehicle m can be expressed as:

$$w_{m,n}(t) = \frac{C_m^k(t)}{\sum\limits_{i=1}^{M} C_i^k(t)} \cdot W_n, \tag{8}$$

where W_n denotes available bandwidth resources of RSU n.

The caching resources evolution of RSU n is expressed as follows:

$$\begin{aligned} Z_n(t+1) =& Z_n(t) + \sum_{k=1}^{K} \{ \mathrm{I}^k \cdot [x_n^{'k}(t-1) \cdot (1 - x_n^{'k}(t)) \\ & - (1 - x_n^{'k}(t-1)) \cdot x_n^{'k}(t)] + \mathrm{O}^k \cdot [x_n^{''k}(t-1) \\ & \cdot (1 - x_n^{''k}(t)) - (1 - x_n^{''k}(t-1)) \cdot x_n^{''k}(t)] \}, \end{aligned} \tag{9}$$

where $Z_n(t)$ denotes current available storage space of RSU n. Specially, if $x_n^{'k}(t-1) \cdot (1 - x_n^{'k}(t)) = 1$, it means that the perceived data of task k is decided to be deleted and can release storage space while $(1 - x_n^{'k}(t-1)) \cdot x_n^{'k}(t) = 1$ means that the data is decided to be cached and will occupy storage space.

The computation resources evolution of RSU n can be expressed as follows:

$$\begin{aligned} G_n(t+1) =& G_n(t) + \sum_{k=1}^{K} \Phi^k \cdot \varphi_n^k(t-1) - \sum_{m=1}^{M} \Phi^k \cdot \\ & [\alpha_{m,n}^k(t) \cdot (1 - x_n^{''k}(t-1)) \cdot (1 - \varphi_n^k(t))], \end{aligned} \tag{10}$$

where $G_n(t)$ denotes current available computation resources of RSU n, and if $\varphi_n^k(t-1) = 1$, it means that task k occupied computation resources at last time slot and should be released at time t. In addition, if $\alpha_{m,n}^k(t) \cdot (1 - x_n^{''k}(t-1)) \cdot (1 - \varphi_n^k(t)) = 1$, it means that vehicle m is connecting with RSU n and task k is not cached at RSU n and task k is not being computed now. Thus, task k of vehicle m will occupy computation resources of RSU n.

Then the optimization problem with bandwidth bidding, offloading scheduling and content updating decisions $(\alpha, x^{'}, x^{''}, \zeta)$ is formulated as:

$$\min_{\alpha, x^{'}, x^{''}, \zeta} \quad U(t) = \sum_{m=1}^{M} \sum_{n=1}^{N} [\omega_d \cdot D_m^k(t) + \omega_c \cdot C_m^k(t)], \tag{11}$$

$$\text{s.t.} \quad \sum_{n=1}^{N} \alpha_{m,n}^k(t) \leq 1, m = 1, 2, \cdots, M, k \in \mathrm{K}, \tag{12}$$

$$\sum_{n=1}^{N} [\alpha_{m,n}^k(t) \cdot R_{m,n}^k + (1 - \alpha_{m,n}^k(t)) \cdot R_m^k] \geq \mathrm{B}_m^k, \quad m = 1, 2, \cdots, M, k \in \mathrm{K}, \tag{13}$$

$$G_n(t+1) \geq 0, n = 1, 2, \cdots, N, \tag{14}$$

$$Z_n(t+1) \geq 0, n = 1, 2, \cdots, N, \tag{15}$$

where (12) denotes each vehicle associates with a single RSU at most. (13) denotes the realistic computation rate should be greater than the threshold of computation rate of task k. (14) and (15) are computation resources constraint and caching space constraint, respectively.

As this mixed integer non-linear programming (MINLP) problem is NP-hard, so deep reinforcement learning (DRL) is adopted to solve this joint optimization problem in given feasible zone.

IV. DDPG-BASED COMPUTING AND CACHING SCHEME

Recently, deep reinforcement learning (DRL) has been a promising technology to tackle the complicated optimization problem in dynamic network environment. Conventional DRL such as DQN can only handle discrete action space while the action space in our proposed scheme is continuous because the bandwidth bidding decision should be a continuous variable. In this case, deep deterministic policy gradient (DDPG)-based algorithm is used to solve the MINLP problem with low computation complexity.

Deep deterministic policy gradient [11] is a strategy learning method using convolutional neural network as the function approximator of policy function and Q function, namely actor network and critic network. The actor network is used to select action deterministically in continuous action space with policy gradient, while the critic network provides a value based on the performance of actor to guide the actor learning policy.

1) State: $S = \{S_\mathrm{N}, S_\mathrm{M}, S_\mathrm{K}\}$ is denoted as state space reflecting the situation of environment, where S_N is the state subspace of N RSUs which includes computation resources, caching space, bandwidth resources, computation capacity, location and so on, and S_M is the state subspace of M vehicles which includes velocity, location, requested tasks and local computation capacity, and S_K is the state subspace of K tasks like a content database consisting of input size, output size, requested computation resources and minimum computation rate of tasks.

2) Action: $A = \{a_1, a_2, \cdots, a_N\}$ is denoted as action space, where a_n means action subspace of RSU n and a_n is expressed as follows:

$$\begin{aligned} a_n = [& \alpha_{1,n}^{k^{(1)}}, \alpha_{2,n}^{k^{(2)}}, \cdots, \alpha_{M,n}^{k^{(M)}}, \zeta_{1,n}^{k^{(1)}}, \zeta_{2,n}^{k^{(2)}}, \cdots, \zeta_{M,n}^{k^{(M)}}, \\ & x_n^{'1}, x_n^{'2}, \cdots, x_n^{'K}, x_n^{''1}, x_n^{''2}, \cdots, x_n^{''K}], \end{aligned} \tag{16}$$

where $\alpha_{m,n}^{k^{(m)}}$ is attaching decision action which means that RSU n provides service for vehicle m with task $k^{(m)}$ if $\alpha_{m,n}^{k^{(m)}} = 1$, otherwise, the task $k^{(m)}$ will be executed locally. $x_n^{'k}$ and $x_n^{''k}$ are caching decision actions of RSU n for the perceived data and result of task, respectively. $\zeta_{m,n}^{k^{(m)}}$ is bidding action of bandwidth resources that RSU n allocates to vehicle m who owns task $k^{(m)}$.

3) Reward: r is denoted as current stage reward function, $r = -U$, where U is the objective function, if the action satisfies conditions, otherwise, $r = -P$, where P is a penalty which is far larger than U.

The proposed collaborative bandwidth allocation, computing and content updating strategy based on DDPG algorithm is demonstrated in Algorithm 1. Specifically, Line 1-3 initialize parameters of neural networks and experience replay buffer.

Algorithm 1 DDPG based collaborative computation and caching policy

```
1: Randomly initialize critic network $Q(s,a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
2: Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$.
3: Initialize replay buffer $R$.
4: for (each episode) do:
5:   Initialize observation state $s_1$
6:   for (each time slot) do:
7:     Choose action $a_t = \mu_\theta(s_t)+\varepsilon$ according to the current policy and exploration noise.
8:     Execute the action and obtain reward $r$.
9:     Update the current state to the next state $s_{t+1}$ based on Equation (7) and (8).
10:    Store transition $(s_t, a_t, s_{t+1}, r_t)$ in $R$.
11:    Sample a random minibatch of N transitions form $R$.
12:    Set$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$.
13:    Update critic by minimizing the loss:
           $L = \frac{1}{N}\sum_i \left(y_i - Q(s_i, a_i|\theta^Q)\right)^2$
       .
14:    Update the actor using policy gradient:
           $\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s=s_t,a=\mu(s_t)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_t}$
15:    Update the target networks:
           $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
           $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
16:  end for
17: end for
```

At the beginning of each episode, our initial observation state will be reset. At each time slot, Line 7 selects a correct action based on current policy and adds an exploration noise to encourage exploration since the policy is deterministic. Line 8-9 receive reward and move to next state, and then the transition information is stored in an experience replay buffer in Line 10. Line 11-15 sample transitions from the replay buffer and update networks.

V. Performance evaluation

In this section, we will evaluate the performance of the proposed collaborative bandwidth allocation, computing and content updating strategy based on DDPG algorithm. The experiment is conducted based on the Python simulator and TensorFlow platform. We consider 5 RSUs uniformly located along a unidirectional road with 1500 m. For vehicle side, 50 vehicles run at the speed 5m/s and the initial location of each vehicle follows exponential distribution. Parameters of vehicular environment in our system are listed in Table III for the ease of reference. For evaluating the performance of our proposed learning-based computing and caching scheme, we also plot the following benchmark methods:

TABLE III
SIMULATION PARAMETERS OF VEHICULAR NETWORK ENVIRONMENT

Parameter description	Value
Length of the unidirectional road	1500 m
Number of RSUs	5
Number of vehicles	50
Types of tasks	30
Computation resource of RSU	50 units
Computation capability of RSU	2 GHz
Caching space of RSU	300 MB
Bandwidth resource of RSU	10 Mbps
Speed of vehicles	5 m/s
Computation capability of vehicles	0.5 - 1.2 GHz
Input size of tasks	30 - 90 MB
Output size of tasks	10 - 50 MB
Requested computation resource of tasks	3 - 9 units
Minimal computation rate of tasks	0 - 1
Value of penalty	50

1) Task offloading with request rate-based caching scheme: perceived data or computation results are cached according to the offloading request rate of tasks.
2) Task offloading with random caching scheme: contents are randomly cached in RSU when tasks are offloaded to RSU.
3) Task offloading with non-caching scheme: contents caching is not considered in this offloading scheme.

Fig. 2 plots cumulative average rewards of different schemes. Since the defined rewards are weighted sum of latency of task execution and bandwidth bids, the final convergence of overall rewards is balance of tradeoff between latency and bandwidth costs. Our proposed scheme has the highest rewards compared to the other three schemes in this vehicular network scenario. In our proposed computing and caching scheme, each RSU will update contents at every time slot in order to adapt to the dynamic network environment. Thus, caching perceived data or computation results of tasks in RSU can significantly reduce the latency of task execution and increase rewards. Specifically, we have following observations:

1) For task offloading with request rate-based caching scheme, since the request rate is changing dynamically and caching space of RSU is limited, only caching contents with high popularity is not suitable and those tasks with high CPU resources or massive input data should also be considered.
2) For task offloading with random caching scheme, due to perceived data or computation results are randomly cached in each RSU and have no relevance to offloaded tasks, rewards are not as high as ours.

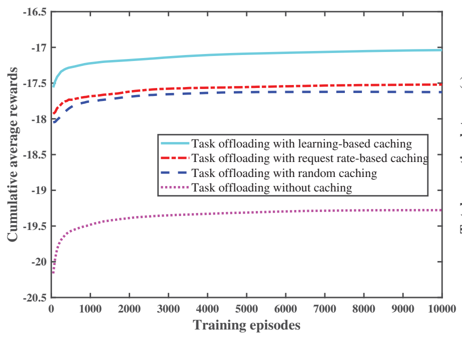

Fig. 2. Comparison of cumulative average rewards on different schemes.

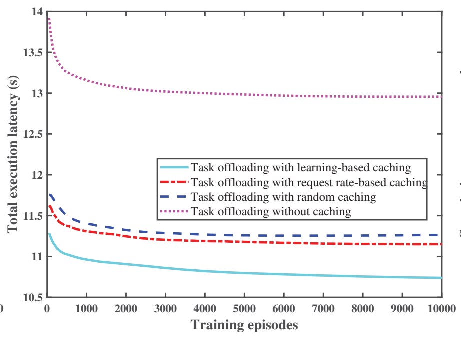

Fig. 3. Comparison of total execution latency on different schemes.

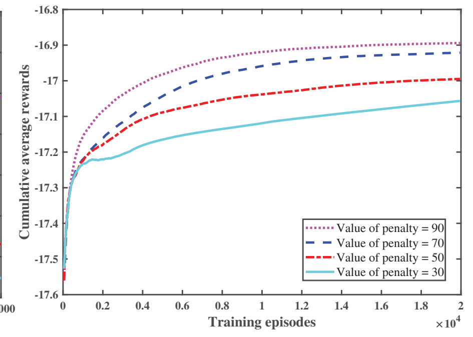

Fig. 4. Impact of penalty factor on the cumulative average rewards.

3) For task offloading without caching, it is obvious that the average rewards are worst since duplicated contents of requested tasks have to be offloaded to the RSU.

Fig. 3 illustrates the total execution latency of these four schemes. It is clear that latency decreases with the number of training episodes increases and gradually remains fairly constant. Compared to the other three schemes, latency of our proposal is the lowest due to adaptive learning-based caching scheme because multiple factors such as request rate, size, computation resources needed of tasks are considered. For offloading request rate based caching scheme, perceived data or results of those tasks with large-size data or high computation resources may be not cached for their low request rate. For random caching scheme, RSU may randomly caches contents with large size and much lower popularity which results in the waste of storage space and longer execution latency tasks. Moreover, the latency in offloading with no caching scheme is the highest because the redundancy of perceived data or tasks is not considered.

In Fig. 4, we show the impact of penalty factor on the average reward. Generally, cumulative average rewards increase as the value of penalty increases, because the agent constantly learns knowledge according to the rewards or punishments received in the interaction with the environment aiming to be more adaptable to the dynamic network states. The learning outcome will be better if the value of punishment is large.

VI. Conclusion and Future Work

In this paper, joint bandwidth allocation, task offloading and content updating for vehicular edge networks has been studied to improve the efficiency of information delivery. We proposed an optimal task computing and content caching scheme with bandwidth bidding enhancement to minimize network overheads in multi-RSU and multi-vehicle scenario. This model was formulated as a mixed integer non-linear programming problem. Because of high dimension and complexity of action space, DDPG-based algorithm was applied to tackle it and got a sub-optimal solution. Compared to three alternative solutions in literature, our proposed approach achieved the best performance. In the future work, we will study the distributed joint computing and content updating scheme in vehicular edge network.

References

[1] A. PIERONI, N. Scarpato, and M. BRILLI, "Industry 4.0 revolution in autonomous and connected vehicle a non-conventional approach to manage big data.," *Journal of Theoretical & Applied Information Technology*, vol. 96, no. 1, 2018.

[2] G. Qiao, S. Leng, K. Zhang, and Y. He, "Collaborative task offloading in vehicular edge multi-access networks," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 48–54, 2018.

[3] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *arXiv preprint arXiv:1908.06849*, 2019.

[4] S. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Energy efficiency and delay tradeoff for wireless powered mobile-edge computing systems with multi-access schemes," *IEEE Transactions on Wireless Communications*, 2019.

[5] L. Yao, A. Chen, J. Deng, J. Wang, and G. Wu, "A cooperative caching scheme based on mobility prediction in vehicular content centric networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 5435–5444, 2017.

[6] Y. Zhang, C. Li, T. H. Luan, Y. Fu, W. Shi, and L. Zhu, "A mobility-aware vehicular caching scheme in content centric networks: Model and optimization," *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 3100–3112, April 2019.

[7] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 44–55, 2017.

[8] L. T. Tan, R. Q. Hu, and L. Hanzo, "Twin-timescale artificial intelligence aided mobility-aware edge caching and computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 3086–3099, April 2019.

[9] M. Li, F. R. Yu, P. Si, H. Yao, and Y. Zhang, "Software-defined vehicular networks with caching and computing for delay-tolerant data traffic," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2018.

[10] S. Yu, R. Langar, X. Fu, L. Wang, and Z. Han, "Computation offloading with data caching enhancement for mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 11098–11112, 2018.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.