Integrated Networking, Caching, and Computing for Connected Vehicles: A Deep Reinforcement Learning Approach

Ying He, *Student Member, IEEE*, Nan Zhao, *Senior Member, IEEE*, and Hongxi Yin

***Abstract*—The developments of connected vehicles are heavily influenced by information and communications technologies, which have fueled a plethora of innovations in various areas, including networking, caching, and computing. Nevertheless, these important enabling technologies have traditionally been studied separately in the existing works on vehicular networks. In this paper, we propose an integrated framework that can enable dynamic orchestration of networking, caching, and computing resources to improve the performance of next generation vehicular networks. We formulate the resource allocation strategy in this framework as a joint optimization problem, where the gains of not only networking but also caching and computing are taken into consideration in the proposed framework. The complexity of the system is very high when we jointly consider these three technologies. Therefore, we propose a novel deep reinforcement learning approach in this paper. Simulation results with different system parameters are presented to show the effectiveness of the proposed scheme.**

***Index Terms*—Caching, deep reinforcement learning, edge computing, vehicular networks.**

I. Introduction

IN RECENT years, there are great interests in connected vehicles, which use advanced technologies to connect vehicles with various infrastructures, devices, users, services, etc. It is envisioned that connected vehicles provide key enabling technologies to improve safety, enhance efficiency, reduce accidents, and decrease traffic congestion in transportation systems.

The developments of connected vehicles are heavily influenced by information and communications technologies, which have fueled a plethora of innovations in various areas, including *networking*, *caching* and *computing*. Recent advances in these areas have been extensively studied in the developments of connected vehicles. In the area of networking, *software-defined networking* (SDN) has attracted great interests in both academia and industry. The basic principle of SDN is to separate the control plane from the data plane, enabling the ability of programming the network via a centralized software-defined controller with a global view of the network. Another related concept is *network function virtualization* (NFV), which enables abstraction of physical networking resources and flexible sharing of resources by multiple users through isolating each other [1]–[3]. It is beneficial to extend SDN and NFV to vehicular networks [4], [5]. *Software defined and virtualized vehicular networks* enable direct programmability of vehicular network controls and abstraction of the underlying infrastructure for a variety of applications of connected vehicles, with improved efficiency and great flexibility in vehicular network management [6].

Another promising technology, *in-network caching*, as one of the key features of *information-centric networking* (ICN), can efficiently reduce the duplicate content transmission in networks [7]. Particularly, caching content (e.g., videos) at network edge nodes (e.g., base stations (BSs) and road side units (RSUs)) has been proposed as one of the key enablers in next generation vehicular networks [8], [9]. The investigation on exploiting caching in vehicular networks has shown that access delays, traffic loads, and network costs can be significantly reduced by caching contents in vehicular networks [10].

Recent advances in computing (e.g., cloud/fog/edge computing) will have profound impacts on vehicular networks as well. *Cloud computing* has become very popular to enable access to a shared pool of computing resources [11], [12]. Nevertheless, as the distance between the cloud and the end device is usually large, cloud computing services may not provide guarantees to low latency applications in vehicular networks. To address these issues, *mobile edge computing* (MEC) [13], [14] have been studied to deploy computing resources closer to end vehicles, which can efficiently improve the quality of service (QoS) for applications that require intensive computations and low latency [15], [16].

While some excellent works have been done on networking, caching and computing in vehicular networks, these three important enabling technologies have traditionally been studied separately in the existing works on vehicular networks. Jointly considering these new technologies for connected vehicles has been largely ignored in the existing research. In this paper, we

Manuscript received March 13, 2017; revised August 12, 2017 and September 7, 2017; accepted September 12, 2017. Date of publication October 6, 2017; date of current version January 15, 2018. This work was supported in part by the Xinghai Scholars Program, in part by the Fundamental Research Funds for the Central Universities under Grant DUT17JC43, and in part by the National Natural Science Foundation of China under Grants 61771089 and 61671101. The review of this paper was coordinated by the Guest Editors of the Connected Vehicle Series. *(Corresponding author: Nan Zhao.)*

Y. He is with the School of Information and Communications Engineering, Dalian University of Technology, Dalian 116024, China, and also with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada (e-mail: heying@sce.carleton.ca).

N. Zhao and H. Yin are with the School of Information and Communication Engineering, Dalian University of Technology, Dalian 116024, China (e-mail: zhaonan@dlut.edu.cn; hxyin@dlut.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2017.2760281

jointly consider networking, caching and computing to enhance the performance of vehicular networks. The distinct features of this paper are as follows.

1) Based on the programmable control principle originated from SDN and caching originated from ICN, we propose an integrated framework that can enable dynamic orchestration of networking, caching and computing resources to improve the performance of next generation vehicular networks.
2) In this framework, we formulate the resource allocation strategy as a joint optimization problem, where the gains of not only networking but also caching and computing are taken into consideration in the proposed framework.
3) The complexity of the system is very high when we jointly consider three technologies. Therefore, we propose a novel deep reinforcement learning approach in this paper. Deep reinforcement learning is an advanced reinforcement learning algorithm that uses deep Q network to approximate the Q value-action function [17], and has been exploited in wireless networks to achieve automatic resource allocation [18], [19]. In this paper, deep reinforcement learning is used to obtain the resource allocation policy in vehicular networks with integrated networking, caching, and computing.
4) Simulation results with different system parameters are presented to show the effectiveness of the proposed scheme. It is illustrated that the performance of vehicular networks can be significantly improved with integrated networking, caching, and computing.

The rest of this paper is organized as follows. In Section II, the system model is presented. The deep reinforcement learning algorithm is presented in Section III. In Section IV, the cache-enabled opportunistic IA network is formulated as a deep reinforcement learning process. Simulation results are discussed in Section V. Finally, conclusions are presented in Section VI.

II. System Description

In this section, we first introduce vehicular networks, followed by recent advances in software-defined and virtulaized vehicular networks. Then, information-centric networking and mobile edge computing are introduced to vehicular networks. Next, we present the proposed framework of integrated networking, caching and computing for connected vehicles.

A. Vehicular Networks

Vehicular networks deliver information and entertainment contents to drivers and vehicles by interconnecting various services, content, and application providers through mobile wireless networks. Typical applications over vehicular networks include advance safety warning, managing and playing audio content, utilizing navigation for driving, delivering entertainment such as movies, games, social networking, listening to incoming and sending outgoing SMS text messages, making phone calls, and accessing Internet-enabled or smartphone-enabled content, as well as more complex applications, such as cooperative driving (e.g., lane-merging assistance and platooning) and autonomous driving (e.g., driverless vehicles) [20].

Different vehicular applications have different preconditions. For example, some of them are exclusively rely on Internet (e.g., web surfing, online gaming, video downloads, and email access), and others are supported by vehicle-to-vehicle communications (e.g., exchanging information and data between vehicles). Most safety applications are based on data dissemination in a neighboring geographical area, while most non-safety applications require a network-layer mobility solution to give vehicles Internet access. In general, vehicular communications are based on two main technologies: dedicated short range communications (DSRC) and cellular networks [21]. By using these two techniques, a vehicle can provide others with information acquired from its own sensors, other neighboring vehicles, road side units (RSUs) or direct access to the Internet.

B. Software-Defined and Virtualized Vehicular Networks

It is beneficial to extend SDN principles (e.g., flexibility, programmability, and centralized control) to manage networking and communication resources in vehicular networks [6], e.g., to optimize channel allocation and network selection and reduce interference in multi-channel, multi-radio environment (e.g., wireless local area networks and cellular networks [22], [23]), to improve packet routing decisions in multi-hop environments, and to effectively handle mobility in high-speed scenarios.

In addition, wireless network virtualization has been considered as an effective and promising approach in the management of network architecture and network resources (e.g., spectrum and infrastructure) [1]. With different QoS, service functions or network requirements, a common physical wireless network can be virtualized into several virtual ones by the hypervisor. Moreover, since virtualization enables the sharing of network infrastructure and resources, the capital expenses (CapEx) as well as operation expenses (OpEx) of wireless access networks can be reduced significantly.

Fig. 1 shows a framework of integrated networking, caching and computing with software-defined and virtualized vehicular networks, where the infrastructure (i.e., the substrate network) is abstracted and virtualized into multiple customized virtual vehicular networks with different QoS requirements.

C. Information-Centric Vehicular Networks

In the connected vehicle ecosystem, huge amounts of information-rich and safety-critical data will be exchanged by RSUs and onboard tranceivers in a variety of vehicular applications. Due to poor-quality wireless links and high vehicle mobility, it is challenging to deliver huge amounts of data using the traditional host-centric IP-based approach in vehicular networks [10]. Recent advances in ICN can be extended to vehicular networks to address this issue. A typical representative example of the ICN architectures is named-data networking (NDN) [24], which uses content name to route and retrieve data. In NDN, the hourglass model of IP is retained, but the waist layer is changed with a hierarchical content naming structure. In

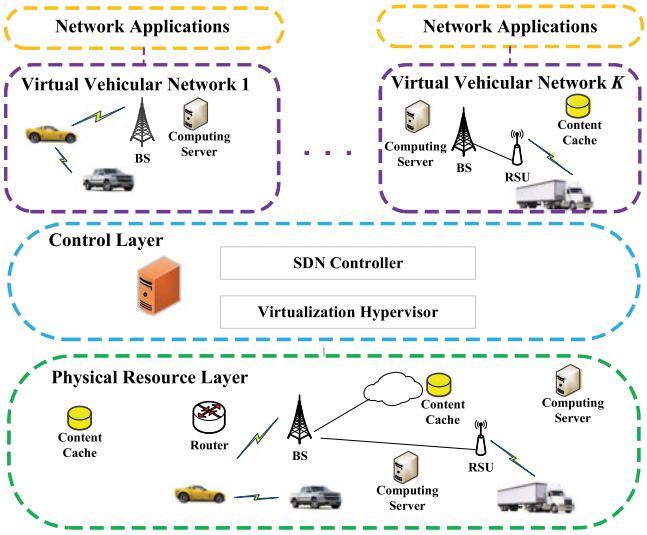


Fig. 1. A framework of integrated networking, caching and computing for connected vehicles.

addition, NDN integrates security into data itself by cryptographically signing every data packet to ensure integrity and authenticity. Moreover, in-network caching, as one of the key features of ICN, can efficiently reduce the duplicate content transmission in networks [25].

It is beneficial to extend this innovative ICN approach to vehicular networks, which natively privilege the information (e.g., trusted road traffic information in a specific proximity of a hazard and a specific time period) rather than the node identity. In addition, with in-network caching, ICN helps to address the mobility and sporadic connectivity issues in vehicular networks.

D. Vehicular Networks With Mobile Edge Computing

Recently, cloud computing has been applied in diverse domains, where user data needs to be transmitted to and processed in the data centers. However, since data centers are usually far away from the end users, cloud computing services may not provide guarantees to low latency applications for connected vehicles, and transmitting a large amount of data (e.g., in big data analytics) from the vehicle to the cloud may not be feasible or economical.

To address these issues, MEC has been proposed to deploy computing resources closer to end vehicles, which can efficiently improve the quality of service (QoS) for applications that require intensive computations and low latency [16].

E. Proposed Framework of Integrated Networking, Caching and Computing for Connected Vehicles

In most existing works on vehicular networks, networking, caching and computing are studied separately. However, from the vehicular application's point of view, network, cache and compute are underlying resources enabling vehicular applications. How to abstract, allocate and optimize these resources can have significant impacts on the performance of vehicular applications. In addition, although this traditional approach provides simplicity of the system design, it could result in suboptimal performance. Therefore, it is desirable to design an integrated framework for connected vehicles that enables network, cache and compute to be abstracted as a common resource pool for different vehicular applications. Moreover, this framework could enable the resources to be dynamically allocated based on the time-varying requirements of different vehicular applications.

Therefore, in this paper, we propose a novel framework of integrated networking, caching and computing in a systematic for connected vehicles. Fig. 1 shows this framework. Based on the programmable control principle originated from SDN, we incorporate the ideas of information centricity originated from ICN. This integrated framework can enable dynamic orchestration of networking, caching and computing resources to meet the requirements of different applications.

One example use case in this framework is as follows. Assume that a vehicle issues a video content request to its associated virtual BS (could be a BS or RSU in the physical wireless network). First of all, according to the description of the video content and the information about the vehicle, the virtual BS will check whether or not its associated cache has the requested content. If yes, the cache will further examine if the version of its cached video content can be played and match with the vehicle. If still yes, the virtual BS will directly send the requested content to the vehicle from the cache. If no, the virtual BS will extract the current video content and construct a computation task according to the size of the content involving the input data, codes and parameters, as well as the number of CPU cycles that is needed to accomplish the computing/transcoding task. Then the virtual BS will transmit the task to the MEC server to execute the computation. After the computation is finished, the virtual BS sends the transcoded video content to the vehicle. If the cache cannot find the matched video content, the virtual BS has to retrieve the content from Internet, and this will inevitably occupy some of the backhaul resources. The procedure is shown in Fig. 2.

When we consider networking, caching and computing jointly, the complexity of the system is very high, which makes it difficult solve the resource allocation problem using traditional approaches [19], [26]. In this paper, we used a novel deep reinforcement learning approach to solve the optimization problem in vehicular networks with integrated networking, caching and computing. Before we describe recent advances of deep reinforcement learning in Section IV, we first present the system model considered in this paper as follows.

III. System Model

In this section, we first present the network model, followed by the communication model. Then, the computation model and caching model are presented.

A. Network Model

We consider a software-defined and virtualized vehicular network with multiple vehicles requesting for video-concerned

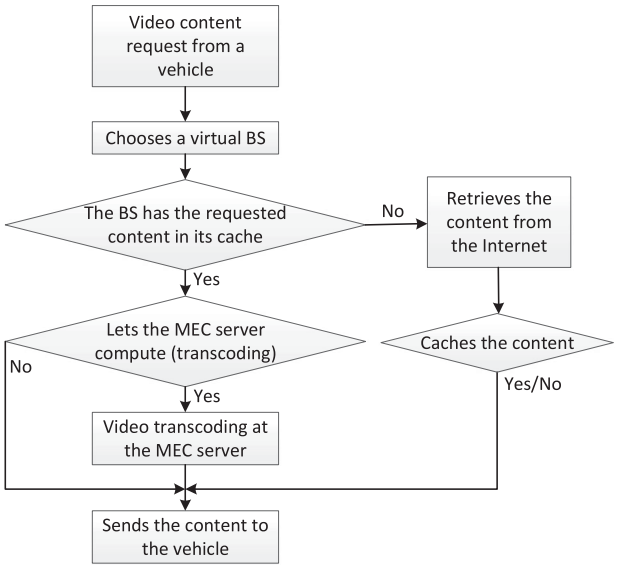


Fig. 2. The procedure of video services in a vehicular network with MEC and caching.

contents. The physical network infrastructures mainly include multiple base stations (BSs), road side units (RSUs), MEC servers, content caches, vehicles and routers, which are operated by the infrastructure providers (InPs). Let $\mathcal{K}_m$ and $\mathcal{K}_s$ be the sets of BSs and RSUs, respectively. $\mathcal{K} = \mathcal{K}_m \cup \mathcal{K}_s = \{1, \ldots, K\}$ and $\mathcal{U} = \{1, \ldots, U\}$ are the sets of all the BSs, RSUs, and vehicles, respectively. The sets of the MEC servers and content caches are denoted as $\mathcal{M} = \{1, \ldots, M\}$ and $\mathcal{C} = \{1, \ldots, C\}$, respectively.

Virtual networks are established logically. Each virtual network includes BSs, RSUs, MEC servers, and content caches, which can provide the capabilities of video content caching and high-speed computing for vehicles. The MEC server and the cache associated with BS k are given by m_k and c_k, $m_k \in \mathcal{M}$ and $\forall c_k \in \mathcal{C}$. It is assumed that each virtual network belongs to a different InP, and the licensed spectrum of different InPs is orthogonal so that there exits no interference among virtual networks. The service providers (SPs) manage the virtual networks, and let $\mathcal{S} = \{1, \ldots, S\}$ be the set of SPs. For an SP s, the set of vehicles is denoted as $\mathcal{U}_s$, among which one allocated vehicle is represented by u_s. All the vehicles are managed by SPs, and one vehicle only subscribes to one SP at a given time period, i.e., $\mathcal{U} = \cup_s \mathcal{U}_s$ and $\mathcal{U}_s \cap \mathcal{U}_{s'} = \phi, \forall s' \neq s$. Let $a_{u_s,k}(t)$ denote the association between vehicle u_s and BS k at time instant t, where $a_{u_s,k}(t) = 1$ means that vehicle u_s connects to BS k to request for the video content; otherwise $a_{u_s,k}(t) = 0$. Practically, each vehicle can only access to one BS or RSU at one time, thus

$$\sum_{k \in \mathcal{K}} a_{u_s,k}(t) = 1, \forall s \in \mathcal{S}, u_s \in \mathcal{U}_s. \quad (1)$$

Since the network model is closely related to communication, computation and caching aspects, next the communication model, computation model, and caching model are described separately in detail.

B. Communication Model

Consider that the wireless channels between the vehicles and their connected BSs/RSUs are realistic time-varying channels, and they are modeled as finite-state Markov channels (FSMC). FSMC models have been widely accepted in the literature as an effective approach to characterize the correlation structure of the fading process [27], [28]. Considering FSMC models may enable substantial performance improvement over the schemes with memoryless channel models.

In FSMC models, the received SNR is a proper parameter that can be used to reflect the quality of a channel. The received SNR of the wireless channel linking vehicle u_s and BS k is modelled as a random variable $\gamma_{u_s}^k$. The value range of $\gamma_{u_s}^k$ can be partitioned and quantized into L discrete levels: $\mathcal{L}_0$, if $\gamma_0^* \leqslant \gamma_{u_s}^k < \gamma_1^*$; $\mathcal{L}_1$, if $\gamma_1^* \leqslant \gamma_{u_s}^k < \gamma_2^*$; $\ldots$; $\mathcal{L}_{L-1}$, if $\gamma_{u_s}^k \geqslant \gamma_{L-1}^*$. Each level corresponds to a state of a Markov chain, thus forms a L-element state space $\mathcal{D} = \{\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{L-1}\}$. The channel state realization of $\gamma_{u_s}^k$ at time instant t can be denoted as $\Upsilon_{u_s}^k(t)$. We assume there are T time slots during the period of the whole communication, which starts when vehicle u_s issues a request and terminates when the requested content is obtained. Let $t \in \{0, 1, 2, \ldots, T-1\}$ denote the time instant. According to certain transition probabilities, the received SNR $\Upsilon_{u_s}^k(t)$ varies from one state to another when one time slot elapses. The transition probability that $\Upsilon_{u_s}^k(t)$ jumps from one state g_s to another state h_s can be denoted as $\psi_{g_s h_s}(t)$ at time slot t. The $L \times L$ wireless channel state transition probability matrix for the link between vehicle u_s and BS k is defined as:

$$\Psi_{u_s}^k(t) = [\psi_{g_s h_s}(t)]_{L \times L}, \quad (2)$$

where $\psi_{g_s h_s}(t) = \Pr\left(\Upsilon_{u_s}^k(t+1) = h_s \mid \Upsilon_{u_s}^k(t) = g_s\right)$, and $h_s, g_s \in \mathcal{D}$.

We assume that the whole available spectrum bandwidth is B Hz, and BS k is allocated with B_k Hz. The backhaul capacity of the vehicular network is Z bps, and BS k is assigned to Z_k bps. Let $b_{u_s,k}$ be the spectrum bandwidth orthogonally assigned from BS k to vehicle u_s, thus there is no interference to vehicle u_s from other vehicles that are also connected to BS k. The achievable spectrum efficiency of vehicle u_s associated with BS k can be denoted as $v_{u_s,k}(t)$, and based on Shannon bound it can be easily achieved. The communication rate of vehicle u_s can be expressed as

$$R_{u_s,k}^{\text{comm}}(t) = a_{u_s,k}(t) b_{u_s,k}(t) v_{u_s,k}(t), \forall s \in \mathcal{S}, u_s \in \mathcal{U}_s. \quad (3)$$

The sum rate of all the vehicles associated with BS k cannot exceed its backhaul capacity, thus the following requirement must be met,

$$\sum_{s \in \mathcal{S}} \sum_{u_s \in \mathcal{U}_s} R_{u_s,k}^{\text{comm}}(t) \leq Z_k, \forall k \in \mathcal{K}. \quad (4)$$

Similarly, the data rate of all the vehicles in the whole virtual network cannot exceed the total backhaul capacity, thus the following condition should be met,

$$\sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}} \sum_{u_s \in \mathcal{U}_s} R_{u_s,k}^{\text{comm}}(t) \leq Z. \quad (5)$$

C. Computation Model

In the computation model, we focus on the constructed computation task $Q_{u_s} = \{o_{u_s}, q_{u_s}\}, \forall s \in \mathcal{S}, u_s \in \mathcal{U}_s$, which is executed on vehicle u_s's associated MEC server m_k. The first parameter o_{u_s} stands for the size of the video content, and the second q_{u_s} denotes the required number of CPU cycles that is needed to complete the task. After the computation, BS k sends the transcoded video content back to vehicle u_s.

We denote the computation capability of BS k allocated to vehicle u_s as $f^k_{u_s}$, which can be measured by CPU cycles per second. In our virtualized vehicular network, different computation-speed MEC servers are assigned to BSs dynamically. In addition, multiple vehicles may access to the same BS and share the same MEC server at a given time. The above two points result in the fact that we don't know exactly the computation capability for vehicle u_s at the next time instant. Thus, the computation capability $f^k_{u_s}$ can be modelled as a random variable, and divided into discrete levels denoted by $\mathcal{E} = \{\mathcal{E}_0, \mathcal{E}_1, \ldots, \mathcal{E}_{N-1}\}$, where N is the number of available computation capability states. Assume the computation capability realization of $f^k_{u_s}$ to be $F^k_{u_s}(t)$ at time slot t. We model the transition of the computation capability level of a BS for one allocated vehicle in this virtualized vehicular network as a Markov chain. The computation capability state transition probability matrix of BS k for vehicle u_s is defined as:

$$\Theta^k_{u_s}(t) = [\iota_{x_s y_s}(t)]_{N\times N}, \tag{6}$$

where $\iota_{x_s y_s}(t) = \Pr\left(F^k_{u_s}(t+1) = y_s \mid F^k_{u_s}(t) = x_s\right)$, and $x_s, y_s \in \mathcal{E}$.

The computation execution time of the task Q_{u_s} at BS k can be obtained as $T_{u_s,k} = \frac{q_{u_s}}{F^k_{u_s}(t)}$. We are more concerned with the metric of computation rate (bits computed per second), which can be given by

$$r^{comp}_{u_s,k}(t) = a_{u_s,k}(t)\frac{o_{u_s}}{T_{u_s,k}} = a_{u_s,k}(t)\frac{f^k_{u_s}(t)o_{u_s}}{q_{u_s}}. \tag{7}$$

Since the computation capacity of each MEC server is limited, the following condition should be satisfied

$$\sum_{s\in\mathcal{S}}\sum_{u_s\in\mathcal{U}_s} a_{u_s,k}(t)o_{u_s} \le O_k, \forall k \in \mathcal{K}, \tag{8}$$

where O_k is the maximum content size simultaneously executed on the MEC server m_k of BS k.

D. Caching Model

Assume that there are I contents in the server for the time interval we concern. For content i, $i \in 1, 2, \ldots, I$, the average request rate for content i at time t can be denoted as

$$\lambda_i(t) = \frac{\beta}{\rho i^\alpha}. \tag{9}$$

Here, the index i represents the ith most popular content and it is assumed that the requests for these I contents arrive according to a Poisson process with rate β [29]. The probability of requesting a content is determined by a Zipf-like distribution [30], thus the probability of content i being selected is $1/\rho i^\alpha$, where $\rho = \sum_{i=1}^{I} 1/i^\alpha$ and α is the Zipf slope such that $0 < \alpha < 1$.

In our system model, the content caches periodically store the contents from the server. Whether or not a vehicle's requested content i is in the cache can be viewed as a random variable ξ_i, and the state of the cache can be modeled using a two-state Markov chain [29]. State 0 corresponds to when content i is not in the cache. State 1 corresponds to when content i is in the cache. The state space can be expressed as $\mathcal{G} = \{0, 1\}$, where $\xi_i \in \mathcal{G}$. The cache state transition probability matrix of content i is defined as:

$$\Gamma_i(t) = [\delta_{\nu_s\omega_s}(t)]_{2\times 2}, \tag{10}$$

where $\delta_{\nu_s\omega_s}(t) = \Pr\left(\xi_i(t+1) = \omega_s \mid \xi_i(t) = \nu_s\right)$, and ν_s, $\omega_s \in \mathcal{G}$.

Based on the cache capacity, two cases can be considered: infinite and finite cache capacity. In the infinite case, the cache can store all the requested contents, and the stored content is only eliminated from the cache when it exceeds the expiry time. The lifetime of content i follows exponential distribution with mean $1/\mu_i$. The cache state transition probability matrix can be derived from the Markov chain flow matrix as [29]

$$\Lambda^0_i = \begin{bmatrix} -\lambda_i & \lambda_i \\ \mu_i & -\mu_i \end{bmatrix} \tag{11}$$

If finite cache capacity is assumed, the cache state transition probability matrix can be derived according to different cache replacement strategies, among which the least recently used (LRU) cache replacement policy is commonly used. The transition probability matrix can be derived using the following Markov chain flow matrix [29],

$$\Lambda^1_i = \begin{bmatrix} -\lambda_i & 0 & \cdots & 0 & 0 & \lambda_i \\ \zeta_i + \mu_i & -\beta-\mu_i & \cdots & 0 & 0 & \lambda_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mu_i & 0 & \vdots & \zeta_i & -\beta-\mu_i & \lambda_i \\ \mu_i & 0 & \vdots & 0 & \zeta_i & -\zeta_i-\mu_i \end{bmatrix} \tag{12}$$

where $\zeta_i = \beta - \lambda_i$.

IV. Deep Reinforcement Learning

In order to better understand state-of-the-art reinforcement learning agent, Deep Q-Network (DQN), a brief review of reinforcement learning and Q-learning are first described. Then recent advances of DQN are presented, and double DQN and dueling DQN that go beyond DQN are also given.

A. Reinforcement Learning

Reinforcement learning is an important branch of machine learning, where an agent learns to take actions that would yield the most reward by interacting with the environment. Different from supervised learning, reinforcement learning cannot learn from samples provided by an experienced external supervisor. Instead, it has to operate based on its own experience despite that it faces with significant uncertainty about the environment.

Reinforcement learning is defined not characterizing learning methods, but by characterizing a learning problem. Any method that is suitable for solving that problem can be considered as a reinforcement learning method [31]. A reinforcement learning problem can be described as an optimal control of a Markov Decision Process (MDP), however, state space, explicit transition probability and reward function are not necessarily required [32]. Therefore, reinforcement learning is powerful in handling tough situations that approach real-world complexity [17].

There are two outstanding features of reinforcement learning: trial-and-error search and delayed reward. Trial-and-error search means making trade-off between exploration and exploitation. The agent prefers to exploit the effective actions that have been tried in the past to produce rewards, but it also has to explore better new actions that may yield higher rewards in the future. The agent must try various actions and progressively favor those that earn the most rewards. The other feature of reinforcement learning is that the agent looks into a big picture, not just considering the immediate reward, but also the cumulative rewards in the long run, which is specified as value function.

Generally, reinforcement learning can be divided into model-free and model-based reinforcement learning, which is based on whether or not the environment elements are already known. Model-free reinforcement learning has recently been successfully applied to handle the deep neural network and value functions [17], [33]–[35]. It can learn policies for tough tasks using the raw state representation directly as the input to the neural networks [36]. Contrastively, model-based reinforcement learn a model of the system with the help of supervised learning and optimize a policy under this model [36]–[38]. Recently, elements of model-based RL have been incorporated into model-free deep reinforcement learning to accelerate the learning rate without losing the strengths of model-free learning [36]. One of the model-free reinforcement learning algorithms is Q-Learning. The most important component of Q-learning algorithms is a method for properly and efficiently estimating the Q value. The Q function can be implemented simply by a look-up table, or by a function approximator, sometimes a nonlinear approximator, such as a neural network, or even more complicated deep neural network. Q-learning, combined with deep neural network, is the so-called deep Q-learning.

B. Deep Q-Learning

The agent that uses a neural network to represent Q function is called Q-network, which is denoted as $Q(x, a; \theta)$. The parameter θ stands for the weights of the neural network, and the Q-network is trained by updating θ at each iteration to approximate the real Q values. While neural networks allow for great flexibility, they do so at the cost of stability when it comes to Q-Learning, which is interpreted in [17]. Deep Q-network that uses deep neural networks instead of approximating the Q-function is proposed recently, and it is proven to be more advantageous with greater performance and more robust learning [17]. To transform an ordinary Q-network into a deep Q-network, three improvements have been implemented.

1) Replacing ordinary neural networks with advanced multi-layer deep convolutional networks, which utilize hierarchical layers of tiled convolutional filters to exploit the local spatial correlations, and make it possible to extract high-level features from raw input data [32], [39].
2) Utilizing Experience Replay, which stores its interaction experience tuple $e(t) = (x(t), a(t), r(t), x(t+1))$ at time t into a replay memory $D(t) = \{e(1), \ldots, e(t)\}$, and then randomly samples batches from the experience pool to train the deep convolutional network's parameters rather than directly using the consecutive samples as in Q-learning. This allows the network to learn from more various past experiences, and restrains the network from only focusing on what it is immediately doing.
3) Adopting a second network, to generate the target Q values that are used to calculate the loss for each action during the training procedure. One network for both estimations of Q values and target Q values would result in falling into feedback loops between the target and estimated values. Thus, in order to stabilize the training, the target network's weights are fixed and periodically updated.

The deep Q-function is trained towards the target value by minimizing the loss function $L(\theta)$ at each iteration, and the loss function can be written as

$$L_i(\theta_i) = E[(y_i - Q(x, a; \theta_i)^2)], \tag{13}$$

where $y_i = r + \varepsilon \max_{a'} Q(x', a'; \theta_i^-)$. Here, the weights θ_i^- are updated as $\theta_i^- = \theta_{i-G}$, i.e., the weights update every G time steps in the deep Q-networks instead of $\theta_i^- = \theta_{i-1}$.

C. Beyond Deep Q-Learning

Since deep Q-learning is first proposed, great efforts have been made for even greater performance and higher stability. Here, we briefly introduce two recent improvements: Double DQN [40] and Dueling DQN [33].

1) Double DQN: In regular DQN, both choosing an action and evaluating the chosen action use the max over the Q values, which would lead to over-optimistic Q value estimation. To relieve the over-estimation problem, the target value in double DQN is designed and updated as

$$y_i^{\text{double}} = r + \varepsilon Q(x', \arg\max_{a'} Q(x', a'; \theta_i); \theta_i^-), \tag{14}$$

where the action choice is decoupled from the target Q-value generation. This simple trick makes the over-estimation significantly reduced, and the training procedure runs faster and more reliably.

2) Dueling DQN: The intuition behind Dueling DQN is that it is not always necessary to estimate the value of taking each available action. For some states, the choice of action makes no influence on what happens. Thus, in dueling DQN, the state-action value $Q(x, a)$ is decomposed into two components as follows,

$$Q(x, a) = V(x) + A(a). \tag{15}$$

Here, $V(x)$ is the value function, which simply represents how good it is to be in a given state x. $A(a)$ is the advantage function, which measures the relative importance of a certain action compared with other actions. After $V(x)$ and $A(a)$ are separately computed, their values are combined back into a single

Q-function at the final layer. This improvement would lead to better policy evaluation.

Combining both the above two techniques can achieve better performance and faster training speed. In the following section, the deep Q-network with the two improvements is utilized to find the optimal policy for a practical network application.

V. Problem Formulation

In this section, we formulate the resource allocation optimization problem as a deep reinforcement learning process.

In our system model, there are K base stations, M MEC servers and C content caches that are virtualized and managed by a mobile virtual network operator (MVNO), and provide services for vehicles. Since we consider the realistic scenarios, where the base stations' downlink channel conditions, the MEC servers' computation abilities and the caches' states are all dynamically changing, the MVNO faces with large amount of system states, and it has to make an decision on which virtualized resources will be assigned to a specific vehicle according to the system's current state. It is barely possible to solve this complicated task by using a traditional method. Deep Q-learning is a recent advance that is capable of receiving complex high-dimensional data as input, and yielding an optimal action for each input data. By taking the advantages of Deep Q-network, the MVNO can manage the system in an effective and efficient way.

Here, the MVNO is responsible for collecting the status from each base station, MEC server and content cache, and then it assembles the whole information into a system state. Then, the MVNO sends the constructed system state to the agent, (i.e., the deep Q-network) and gets a feedback of the optimal policy a^* for arranging which resources for a certain vehicle. After obtaining the action, the MVNO will send a notice to inform the vehicle to which virtualized network it can access.

Inside the deep Q-network, the replay memory stores the agent's experience of each time slot. The Q network parameters θ is updated at every time instant with samples from the replay memory. The target Q network parameters θ^- is copied from the Q network every N time instants. The ε-greedy policy is utilized to balance the exploration and exploitation, i.e., to balance the reward maximization based on the knowledge already known with trying new actions to obtain knowledge unknown. The training algorithm of the deep Q network is described in Algorithm 1.

In order to obtain the optimal policy, it is necessary to identify the actions, states and reward functions in our deep Q learning model, which will be described in the next following subsections.

Algorithm 1: Double-Dueling-Deep Q-network algorithm in the virtualized vehicular network.

Initialization.

Initialize the experience replay buffer.
Initialize the main deep-Q network with weights θ.
Initialize the target deep-Q network with weights $\theta^- = \theta$.

For episode $k = 1, \ldots, K$ **do**:

Receive the initial observation s_1, and pre-process s_1 to be the beginning state x_1.

For $t = 1, 2, 3..., T$ **do**:

Choose a random probability p.
Choose a_t as,
if $p \leq \varepsilon$
randomly select an action a_t,
otherwise,
$a_t = \arg\max_a Q(x, a; \theta)$.
Execute action a_t in the system, and obtain the reward r_t, and the next observation s_{t+1}.
Process s_{t+1} to be the next state x_{t+1}.
Store the experience (x_t, a_t, r_t, x_{t+1}) into the experience replay buffer.
Get a batch of U samples (x_i, a_i, r_i, x_{i+1}) from the replay memory.
Calculate the target Q-value y_t^- from the target deep-Q network,

$$y_i^- = r_i + \varepsilon Q(x_{i+1}, arg\max_{a'} Q(x_{i+1}, a'; \theta); \theta^-)$$

Update the main deep Q-network by minimizing the loss $L(\theta)$,

$$L(\theta) = \frac{1}{U}\sum_i (y_i^- - Q(x_i, a_i; \theta))^2,$$

and perform a gradient descent step on $L(\theta)$ with respect to θ.
Every G steps, update the target deep Q-network parameters with rate σ,

$$\theta^- = \sigma\theta + (1-\sigma)\theta^-.$$

End for
End for

A. System State

The state of an available base station $k \in \{1, 2, \ldots, K\}$, an available MEC server $m \in \{1, 2, \ldots, M\}$, and an available cache $c \in \{1, 2, \ldots, C\}$ for vehicle u_s in time slot $t \in \{0, 1, \ldots, T-1\}$ is determined by the realization of the states $\Upsilon_{u_s}^k(t)$ of the random variables $\gamma_{u_s}^k$, the realization of the states $F_{u_s}^m(t)$ of the random variables $f_{u_s}^m$, and the realization of the states $\Xi_{u_s}^c(t)$ of the random variables ξ_c. Consequently, the state vector can be described as follows.

$$\begin{aligned}\chi_{u_s}(t) = \big[&\Upsilon_{u_s}^1(t), \Upsilon_{u_s}^2(t), \ldots, \Upsilon_{u_s}^K(t),\\ &F_{u_s}^1(t), F_{u_s}^2(t), \ldots, F_{u_s}^M(t),\\ &\Xi_{u_s}^1(t), \Xi_{u_s}^2(t), \ldots, \Xi_{u_s}^C(t)\big]. \end{aligned} \quad (16)$$

Here, $\Xi_{u_s}^c(t) = [\xi_1, \xi_2, \ldots, \xi_I]$, for $\forall i \in \{1, 2, \ldots, I\}$, $\xi_i \in \{0, 1\}$.

B. System Action

In the system, the agent has to decide which BS is assigned to the vehicle, whether or not the requested content should be cached in the BS, and whether or not the computation task should be offloaded to the MEC server.

The current composite action $a_{u_s}(t)$ is denoted by

$$a_{u_s}(t) = \{a_{u_s}^{\text{comm}}(t), a_{u_s}^{\text{comp}}(t), a_{u_s}^{\text{cache}}(t)\}, \tag{17}$$

where $a_{u_s}^{\text{comm}}(t), a_{u_s}^{\text{comp}}(t), a_{u_s}\,\text{cache}(t)$ are defined and interpreted as follows:

1) Define row vector $a_{u_s}^{\text{comm}}(t) = [a_{u_s,1}^{\text{comm}}(t), a_{u_s,2}^{\text{comm}}(t), \ldots, a_{u_s,K}^{\text{comm}}(t)]$, where $a_{u_s,k}^{\text{comm}}(t)$ represents the communication control of the kth base station for vehicle u_s, and each element $a_{u_s,k}^{\text{comm}}(t) \in \{0,1\}$, where $a_{u_s,k}^{\text{comm}}(t) = 0$ means the base station k is passive (not connected) at time slot t, and $a_{u_s,k}^{\text{comm}}(t) = 1$ means it is active (connected). Note that $\sum_{k\in\mathcal{K}} a_{u_s,k}^{\text{comm}}(t) = 1$.
2) Define row vector $a_{u_s}^{\text{comp}}(t) = [a_{u_s,1}^{\text{comp}}(t), a_{u_s,2}^{\text{comp}}(t), \ldots, a_{u_s,M}^{\text{comp}}(t)]$, where $a_{u_s,m}^{\text{comp}}(t)$ represents the offloading control of the mth MEC server for vehicle u_s, and each element $a_{u_s,m}^{\text{comp}}(t) \in \{0,1\}$, where $a_{u_s,m}^{\text{comp}}(t) = 0$ means the task is not offloaded to the mth MEC server at time slot t, and $a_{u_s,m}^{\text{comp}}(t) = 1$ means it is offloaded. Note that $\sum_{m\in\mathcal{M}} a_{u_s,m}^{\text{comp}}(t) = 1$.
3) Define row vector $a_{u_s}^{\text{cache}}(t) = [a_{u_s,1}^{\text{cache}}(t), a_{u_s,2}^{\text{cache}}(t), \ldots, a_{u_s,C}^{\text{cache}}(t)]$, where $a_{u_s,c}^{\text{cache}}(t)$ represents the cache control of the cth content cache for vehicle u_s, and each element $a_{u_s,c}^{\text{cache}}(t) \in \{0,1\}$, where $a_{u_s,c}^{\text{cache}}(t) = 0$ means the content is not cached at the cth content cache at time slot t, and $a_{u_s,c}^{\text{cache}}(t) = 1$ means it is cached. Note that $\sum_{c\in\mathcal{C}} a_{u_s,c}^{\text{cache}}(t) = 1$.

C. Reward Function

In this paper, we set the comprehensive revenue of the MVNO as our system's reward. MVNO rents the wireless spectrum and the backhaul bandwidth from InPs, and assigns them to virtual SPs. MVNO needs to pay for the usage of spectrum to InPs, which is defined as δ_k per Hz for BS k. Furthermore, MVNO also needs to pay the computation fee as long as a computation task is executed on the MEC server. The unit price for the energy consumption at mth MEC server is defined as η_m per Joule. The fee for caching the content is denoted as ζ_c per unit space.

On the other hand, MVNO charge the vehicles for accessing to the virtual networks, which is defined as τ_{u_s} per bps. The vehicles, who have already paid the fee, can access to the virtual network for offloading their computation task. Besides, the fee for vehicle u_s to compute the task at BS k is defined as ϕ_{u_s} per bps. The backhaul cost, paid by MVNO is defined as κ_{u_s} per bps, can be saved when vehicles call for the contents that have already been cached at cache c.

The system reward is the MVNO's revenue, and it is formulated as the function of received SNR of the access wireless link, the computation capability and the cache state. The system action determines whether the reward will be gained. Therefore, we define the reward for a specific vehicle u_s as:

$$\begin{aligned}
R_{u_s}(t) &= \sum_{k=1}^{K} R_{u_s,k}^{\text{comm}}(t) + \sum_{m=1}^{M} R_{u_s,m}^{\text{comp}}(t) + \sum_{c=1}^{C} R_{u_s,c}^{\text{cache}}(t) \\
&= \sum_{k=1}^{K} a_{u_s,k}^{\text{comm}}(t)(\tau_{u_s} r_{u_s,k}^{\text{comm}}(1 - w^{\text{comp}} a_{u_s,m}^{\text{comp}}(t)) - \delta_k b_{u_s,k}) \\
&\quad + \sum_{m=1}^{M} a_{u_s,m}^{\text{comp}}(t)(\phi_{u_s} r_{u_s,m}^{\text{comp}} - \eta_m q_{u_s} e_m) \\
&\quad + \sum_{c=1}^{C} a_{u_s,c}^{\text{cache}}(t)(\kappa_{u_s} r_{u_s,k}^{\text{cache}} - \zeta_c o_{u_s}) \\
&= \sum_{k=1}^{K} a_{u_s,k}^{\text{comm}}(t)(\tau_{u_s} b_{u_s,k} v_{u_s,k}(1 - w^{\text{comp}} a_{u_s,m}^{\text{comp}}(t)) - \delta_k b_{u_s,k}) \\
&\quad + \sum_{m=1}^{M} a_{u_s,m}^{\text{comp}}(t)(\phi_{u_s} \frac{F_{u_s}^{m}(t) o_{u_s}}{q_{u_s}} - \eta_m q_{u_s} e_m) \\
&\quad + \sum_{c=1}^{C} a_{u_s,c}^{\text{cache}}(t)(\kappa_{u_s} b_{u_s,k} v_{u_s,k} \Xi_{u_s}^{c}(t) - \zeta_c o_{u_s}).
\end{aligned} \tag{18}$$

The immediate system reward is the MVNO's revenue from a certain user u_s at time instant t, i.e., $R_{u_s}(t)$ in equation (25). The agent gets $R_{u_s}(t)$ in state $\chi_{u_s}(t)$ when action $a_{u_s}(t)$ is performed in time slot t. The goal of using deep Q network is to find an optimal policy to maximize the long-term revenue of MVNO, and the cumulative revenue can be written as

$$R_{u_s}^{\text{long}} = \max E\left[\sum_{t=0}^{t=T-1} \epsilon^t R_{u_s}(t)\right], \tag{19}$$

where ϵ^t approaches to zero when t is large enough. In practice, a threshold for terminating the process can be set.

VI. Simulation Results and Discussions

In this section, we use computer simulations to show the performance of the proposed deep reinforcement learning approach to vehicular networks with integrared networking, caching and computing. We use TensorFlow [41] in our simulations to implement deep reinforcement learning. In this section, we first present simulation settings. Then, simulation results are discussed.

A. Simulation Settings

In our simulations, we used a GPU-based server, which has 4 Nvidia GPUs with version GTX TITAN. The CPU is Intel Xeno E5-2683 v3 with 128G memory. Software environment we utilized is TensorFlow 0.12.1 with Python 2.7 on Ubuntu 14.04 LTS.

For performance comparison, 4 schemes are compared with the proposed scheme:

1) Existing static scheme with (w.) communication, computing and caching: The system state is assumed to be static, which does not change dynamically. Communication,

computing and caching are jointly optimized under this static assumption.

2) Proposed scheme without (w.o.) virtualization: Virtulization is not considered in this scheme, and vehicles can only connect to one SP. MEC offloading and caching are considered in this scheme.
3) Proposed scheme without (w.o.) MEC offloading: MEC offloading is not considered in this scheme, and vehicles can only perform the computation tasks locally. Virtualization and caching are considered in this scheme.
4) Proposed scheme without (w.o.) edge caching: Edge caching is not considered in this scheme, and vehicles can only get the video contents from the remote server. Virtualization and MEC offloading are considered in this scheme.

All BSs and vehicles are randomly distributed within the covered area of the MBS. In the simulations, we assume that there are five SPs, five BSs, five MEC server, one MVNO, and the bandwidth of each BS is normalized. The wireless channels between the vehicles and BSs follow the Markov model. We assume that the state of the channel can be bad (the spectrum efficiency $v_{u_s,k} = 1$) or good (the spectrum efficiency $v_{u_s,k} = 3$). We set the transition probability of staying in the same state as 0.7, and set the transition probability from one state to another as 0.3. There is one video content, and the cache state follows the Markov model, We set the cache state transition probability of staying in the same state as 0.6, and set the transition probability from one state to another as 0.4. The computation states of MEC servers follow the Markov model. We assume that the computation state of the MEC server can be very low, low, medium, high, and very high. We set the transition probability matrix as follows.

$$\Theta = \begin{bmatrix} 0.5 & 0.25 & 0.125 & 0.0625 & 0.0625 \\ 0.0625 & 0.5 & 0.25 & 0.125 & 0.0625 \\ 0.0625 & 0.0625 & 0.5 & 0.25 & 0.125 \\ 0.125 & 0.0625 & 0.0625 & 0.5 & 0.25 \\ 0.25 & 0.125 & 0.0625 & 0.0625 & 0.5 \end{bmatrix}, \quad (20)$$

The values of the rest of parameters are summarized in Table I.

B. Simulation Results

Fig. 3 shows the convergence performance of different scenarios in the proposed scheme using the deep reinforcement learning algorithm. From Fig. 3, we can observe that the total utility of different scenarios in the proposed scheme is very low at the beginning of the learning process. With the increase of the number of the episodes, the total utility increases until it reaches a relatively stable value, which is around 6600 in the proposed scheme with integrated networking, caching and computing in Fig. 3. This shows the convergence performance of the proposed scheme. We can also observe that the total utility of other scenarios is less when networking, caching and computing are not jointly considered. Particularly, the total utility is the lowest when virtualization is not considered compared to other scenarios.

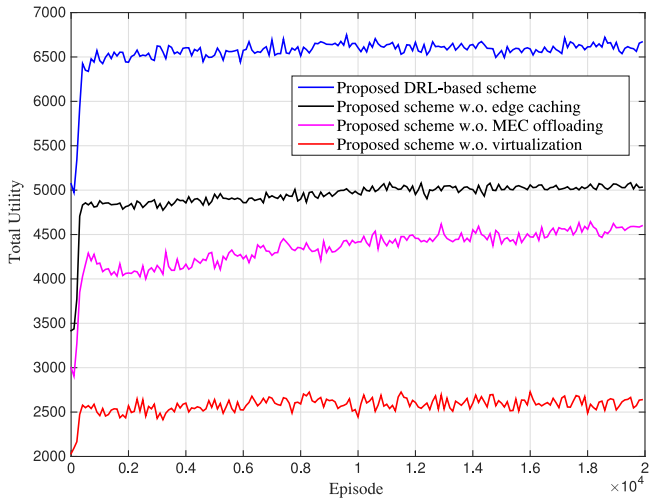


Fig. 3. Convergence performance of different schemes.

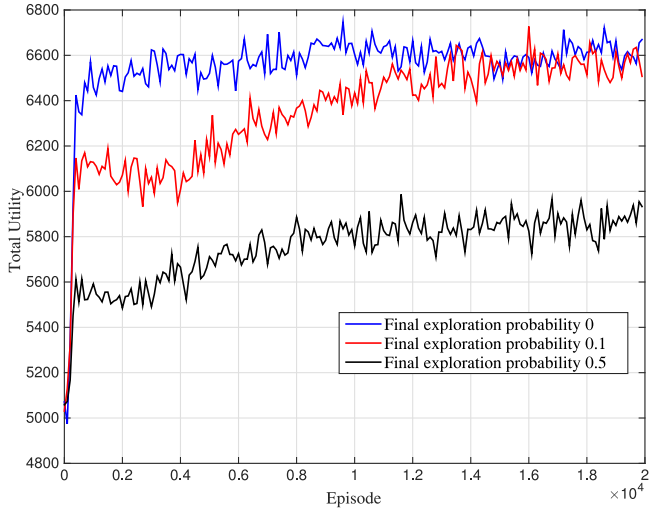


Fig. 4. Convergence performance with different final exploration probabilities.

Fig. 4 shows the convergence performance of the proposed scheme with different final exploration probabilities in the deep reinforcement learning algorithm. Exploration probability is the chance of performing a random action, otherwise it will perform the action produced by the deep Q-network. In order to balance the exploration and exploitation, the initial exploration probability is set large, and the final exploration probability is set much smaller, even zero. From Fig. 4, we can observe the that the final exploration probability has some effects on the total utility of the proposed scheme. Specifically, a higher final exploration probability will result in a lower total utility after the algorithm converges. Nevertheless, a lower final exploration probability has the risk of finding local optimum instead of global optimum. In the rest of the simulations, we choose the final exploration probability of 0.

Fig. 5 shows the effects of o_{u_s}, which is the content size. We can see from Fig. 5 that the total utility of the proposed scheme without MEC offloading deceases with the increase of content

TABLE I
PARAMETER VALUES

Parameter	Value	Description
$b_{u_s,k}$	5 MHz	The bandwidth of BS k allocated to vehicle u_s.
τ_{u_s}	10 units/Mbps	The unit charging-price for accessing to the virtualized networks.
δ_k	2 units/MHz	The unit paid-price for the usage of wireless spectrum.
ϕ_{u_s}	1 units/Mbps	The unit charging-price for executing MEC offloading.
q_{u_s}	100 Mcycles	The required number of CPU cycles to complete each task.
o_{u_s}	1 Mbits	The content size for one task.
$v_{u_s,k}$	[1, 3] bps/Hz	The spectrum efficiency.
κ_{u_s}	3 units/Mbps	The charging-price for connecting to the cache servers.
$F^k_{u_s}$	[4, 6, 8, 10,12] GHz	The realization of the computation capability $f^k_{u_s}$.
η_m	100 units/J	The paid-price for unit energy consumption of the MEC servers.
e_m	1W/GHz	The energy consumption for performing one CPU cycle.
w^{comp}	0.5	The effect factor of executing MEC offloading on the virtualization.
ζ_c	3 units/Mbits	The unit paid-price for connecting to the cache servers.

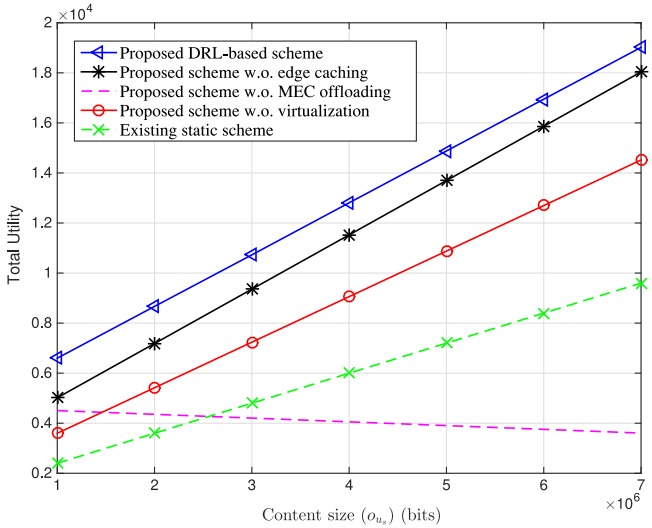


Fig. 5. Effects of o_{u_s}, which is the content size.

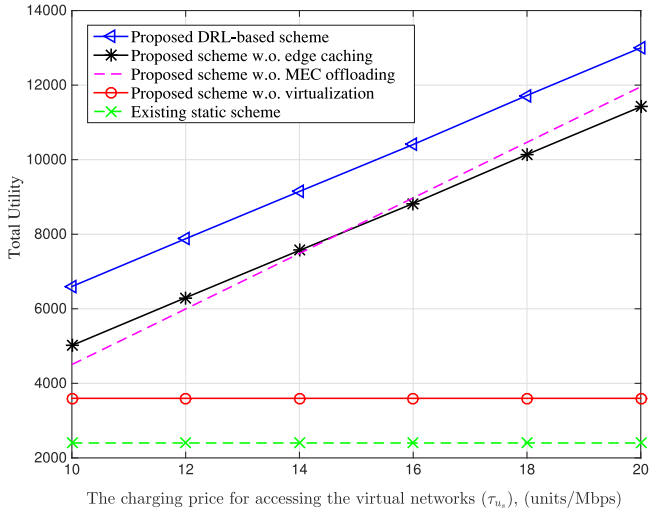


Fig. 6. Effects of τ_{u_s}, which is the unit charging price for accessing to the virtualized networks.

size. This is because a larger content size will increase the fee for caching the content, which induces a lower gain of caching utility, and thus the total utility decreases. By contrast, the total utility of the schemes with MEC offloading increases with the increase of content size due to the fact that a larger content size will increase the charging fee for computation at the MEC server, which induces a higher gain of computation utility, and thus the total utility increases.

Fig. 6 shows the effects of τ_{u_s}, which is the unit charging price for accessing to the virtualized networks. From the figure, we can see that the gap between the proposed scheme with and without MEC offloading narrows with the increase of τ_{u_s}. This is due to the fact that MEC offloading will reduce the charging fees from accessing to the virtualized networks. With the increase of τ_{u_s}, the virtualization occupies increasingly higher proportion in the overall incomings, thus MEC offloading will be less activated and phases out in the proposed scheme. By contrast, the scheme without MEC offloading will increase much faster than the scheme with MEC offloading, until they equal.

Fig. 7 shows the effects of ϕ_{u_s}, which is the unit charging price for executing MEC offloading. From the figure, it can be observed that the difference between the proposed scheme with and without virtualization increases with ϕ_{u_s}. This is because that in the proposed scheme with the rise of ϕ_{u_s}, the incomings of executing MEC offloading would increase, thus MEC offloading is of highly probability to be performed , which results in lower earnings from virtualization. This explains the increasing difference between the proposed scheme with and without virtualization.

Fig. 8 shows the effects of κ_{u_s}, which is the unit charging price for connecting to cache servers. From the figure, we can see that the total utility of the two schemes without virtualization and without MEC offloading grow parallelly with the proposed

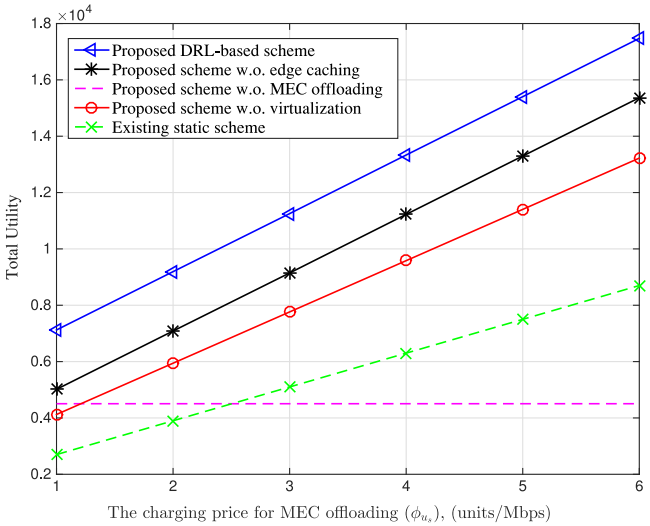


Fig. 7. Effects of ϕ_{u_s}, which is the unit charging price for MEC offloading.

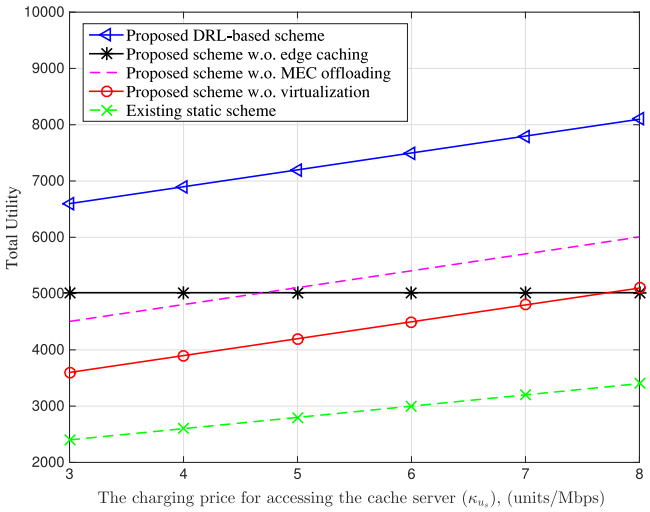


Fig. 8. Effects of κ_{u_s}, which is the unit charging price for connecting to the cache servers.

scheme, which indicates that the increase of κ_{u_s} has not much effect on these two schemes.

VII. Conclusions and Future Work

In this paper, we jointly studied networking, caching and computing to enhance the performance of vehicular networks. Based on the programmable control principle originated from SDN and caching originated from ICN, we proposed an integrated framework that can enable dynamic orchestration of networking, caching and computing resources to improve the performance of next generation vehicular networks. In this framework, we formulated the resource allocation strategy as a joint optimization problem, where the gains of not only networking but also caching and computing are taken into consideration in the proposed framework. Then, we proposed a novel deep reinforcement learning approach in this paper. We presented the convergence performance of different scenarios in the proposed scheme using the deep reinforcement learning algorithm. Simulation results with different system parameters were presented to show the effectiveness of the proposed scheme. Future work is in progress to consider energy efficiency issues in the proposed framework.

References

[1] C. Liang and F. R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Commun. Surveys Tut.*, vol. 17, no. 1, pp. 358–380, Jan.–Mar. 2015.

[2] L. Cui, F. R. Yu, and Q. Yan, "When big data meets software-defined networking (SDN): SDN for big data and big data for SDN," *IEEE Netw.*, vol. 30, no. 1, pp. 58–65, Jan./Feb. 2016.

[3] Y. Cai, F. R. Yu, C. Liang, B. Sun, and Q. Yan, "Software defined device-to-device (D2D) communications in virtual wireless networks with imperfect network state information (NSI)," *IEEE Trans. Veh. Technol.*, vol. 65, no. 9, pp. 7349–7360, Sep. 2016.

[4] K. Liu, J. K. Y. Ng, V. C. S. Lee, S. H. Son, and I. Stojmenovic, "Cooperative data scheduling in hybrid vehicular ad hoc networks: VANET as a software defined network," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1759–1773, Jun. 2016.

[5] Q. Zheng, K. Zheng, H. Zhang, and V. C. M. Leung, "Delay-optimal virtualized radio resource scheduling in software-defined vehicular networks via stochastic learning," *IEEE Trans. Veh. Technol.*, vol. 65, no. 4, pp. 7857–7867, Oct. 2016.

[6] R. dos Reis Fontes, C. Campolo, C. E. Rothenberg, and A. Molinaro, "From theory to experimental evaluation: Resource management in software-defined vehicular networks," *IEEE Access*, vol. 5, pp. 3069–3076, Feb. 2017.

[7] C. Fang, F. R. Yu, T. Huang, J. Liu, and Y. Liu, "A survey of green information-centric networking: Research issues and challenges," *IEEE Commun. Surveys Tut.*, vol. 17, no. 3, pp. 1455–1472, Jul.–Sep. 2015.

[8] K. Liu, J. K. Y. Ng, J. Wang, V. C. S. Lee, W. Wu, and S. H. Son, "Network-coding-assisted data dissemination via cooperative vehicle-to-vehicle/-infrastructure communications," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 6, pp. 1509–1520, Jun. 2016.

[9] C. Liang, F. R. Yu, and X. Zhang, "Information-centric network function virtualization over 5G mobile wireless networks," *IEEE Netw.*, vol. 29, no. 3, pp. 68–74, May/Jun. 2015.

[10] M. Amadeo, C. Campolo, and A. Molinaro, "Information-centric networking for connected vehicles: A survey and future perspectives," *IEEE Commun. Mag.*, vol. 54, no. 2, pp. 98–104, Feb. 2016.

[11] M. Armbrust, A. Fox, R. Griffith, and A. D. Joseph, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[12] H. Zhang, Q. Zhang, and X. Du, "Toward vehicle-assisted cloud computing for smartphones," *IEEE Trans. Veh. Technol.*, vol. 64, no. 12, pp. 5610–5618, Dec. 2015.

[13] ETSI, "Mobile-edge computing—Introductory technical white paper," *ETSI White Paper*, Sep. 2014.

[14] F. R. Yu, C. Liang, Y. He, and N. Zhao, "Energy-efficient resource allocation in software-defined mobile networks with mobile edge computing and caching," in *Proc. IEEE Infocom Workshops*, May 2017.

[15] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[16] N. Kumar, S. Zeadally, and J. J. P. C. Rodrigues, "Vehicular delay-tolerant networks for smart grid data management using mobile edge computing," *IEEE Commun. Mag.*, vol. 54, no. 10, pp. 60–66, Oct. 2016.

[17] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

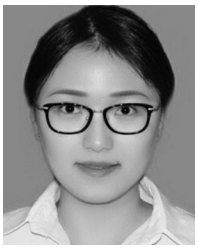
[18] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Mag.*, vol. 55, Dec. 2017.

[19] Y. He *et al.*, "Deep reinforcement learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Trans. Veh. Technol.*, to be published, doi: 10.1109/TVT.2017.2751641.

[20] C. Campoloa, A. Molinaroa, and R. Scopigno, "From today's vanets to tomorrow's planning and the bets for the day after," *Elsevier Veh. Commun.*, vol. 2, no. 3, pp. 158–171, 2015.

[21] K. Abboud, H. A. Omar, and W. Zhuang, "Interworking of DSRC and cellular network technologies for V2X communications: A survey," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 9457–9470, Dec. 2016.

[22] L. Ma, F. Yu, V. C. M. Leung, and T. Randhawa, "A new method to support UMTS/WLAN vertical handover using SCTP," *IEEE Wireless Commun.*, vol. 11, no. 4, pp. 44–51, Aug. 2004.
[23] F. Yu and V. Krishnamurthy, "Optimal joint session admission control in integrated WLAN and CDMA cellular networks with vertical handoff," *IEEE Trans. Mobile Comput.*, vol. 6, no. 1, pp. 126–139, Jan. 2007.
[24] L. Zhang *et al.* "Named data networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, Jul. 2014.
[25] G. Xylomenos *et al.*, "A survey of information-centric networking research," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 1024–1049, Apr.–Jun. 2014.
[26] Y. He, F. R. Yu, N. Zhao, H. Yin, H. Yao, and R. C. Qiu, "Big data analytics in mobile cellular networks," *IEEE Access*, vol. 4, pp. 1985–1996, Mar. 2016.
[27] Y. Wei, F. R. Yu, and M. Song, "Distributed optimal relay selection in wireless cooperative networks with finite-state Markov channels," *IEEE Trans. Veh. Technol.*, vol. 59, no. 5, pp. 2149 –2158, Jun. 2010.
[28] H. Wang, F. R. Yu, L. Zhu, T. Tang, and B. Ning, "Finite-state Markov modeling for wireless channels in tunnel communication-based train control (CBTC) systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 3, pp. 1083–1090, Jun. 2014.
[29] H. Gomaa, G. G. Messier, C. Williamson, and R. Davies, "Estimating instantaneous cache hit ratio using Markov chain analysis," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1472–1483, Oct. 2013.
[30] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. IEEE Annu. Joint Conf. IEEE Comput. Commun. Soc.*, pp. 126–134, vol. 1, 1999.
[31] R. S. Sutton, "Introduction: The challenge of reinforcement learning," in *Reinforcement Learning*. New York, NY, USA: Springer, 1992, pp. 1–3.
[32] H. Y. Ong, K. Chavez, and A. Hong, "Distributed deep Q-learning," arXiv:1508.04186, 2015.
[33] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," arXiv:1511.06581, 2015.
[34] M. Hausknecht and P. Stone, "Deep reinforcement learning in parameterized action space," arXiv:1511.04143, 2015.
[35] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," arXiv:1509.02971, 2015.
[36] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep Q-learning with model-based acceleration," in *Proc. 33th Int. Conf. Mach. Learn.*, vol. 48, pp. 2829–2838, 2016.
[37] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 39, pp. 1–40, 2016.
[38] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 465–472.
[39] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," arXiv:1312.5602, 2013.
[40] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. 30th AAAI Conf. Artificial Intell.*, 2016, pp. 2094–2100.
[41] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous systems," arXiv:1603.04467, Nov. 2016.



Ying He (S'16) received the B.S. degree from Dalian Ocean University, Dalian, China, and the M.S. degree from Dalian University of Technology, Dalian, China, in 2011 and 2015, respectively, both in communication and information systems. She is currently working toward the Ph.D. degree with both Dalian University of Technology and Carleton University, Ottawa, ON, Canada. Her current research interests include big data, wireless networks, and machine learning.

Nan Zhao (S'08–M'11–SM'16) received the B.S. degree in electronics and information engineering, the M.E. degree in signal and information processing, and the Ph.D. degree in information and communication engineering from the Harbin Institute of Technology, Harbin, China, in 2005, 2007, and 2011, respectively. He is currently an Associate Professor with the School of Information and Communication Engineering, Dalian University of Technology, Dalian, China. He has authored more than 100 papers in refereed journals and international conferences. His current research interests include interference alignment, cognitive radio, wireless power transfer, and physical layer security. He is a Senior Member of the Chinese Institute of Electronics. He is serving or has served on the editorial boards of several journals, including *Journal of Network and Computer Applications*, IEEE ACCESS, WIRELESS NETWORKS, PHYSICAL COMMUNICATION, *AEU-International Journal of Electronics and Communications, Ad Hoc & Sensor Wireless Networks*, and *KSII Transactions on Internet and Information Systems*. He received the Top Reviewer Award from the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY in 2016, and was nominated as an Exemplary Reviewer by the IEEE COMMUNICATIONS LETTERS in 2016. Additionally, he worked as a TPC member for many conferences, such as Globecom, VTC, and WCSP.

Hongxi Yin received the B.Sci. degree from Shandong University, Jinan, China, in 1982, the M.Eng. degree from Harbin Institute of Technology, Harbin, China, in 1988, and the Ph.D. degree from Zhongshan University, Guangzhou, China, in 1998. He finished his postdoctoral research work from Peking University, Beijing, China, in 2000. From 2005 to 2007, he was a Visiting Research Fellow with the Optoelectronics Research Centre, University of Southampton, Southampton, U.K. He was in the School of Electronics Engineering and Computer Science, Peking University as an Associate Professor from 2000 to 2008 and then joined the School of Information and Communication Engineering, Dalian University of Technology as a Professor. He is the author or coauthor of about 80 national and international journal papers and has obtained several patents. His recent interests include optical fiber communication technologies, all-optical code-division multiple access networks and photonic signal processing, optical cross connect and wavelength division multiplexing all-optical networking, optical packet switching, automatic switched optical network, as well as interference alignment in cognitive radio.

Dr. Yin is a Senior Member of the Chinese Institute of Electronics.