Deep Reinforcement Learning for Cooperative Content Caching in Vehicular Edge Computing and Networks

Guanhua Qiao, Supeng Leng, Sabita Maharjan, *Member, IEEE*, Yan Zhang, *Senior Member, IEEE*, and Nirwan Ansari, *Fellow, IEEE*

***Abstract*—In this article, we propose a cooperative edge caching scheme, a new paradigm to jointly optimize the content placement and content delivery in the vehicular edge computing and networks, with the aid of the flexible trilateral cooperations among a macro-cell station, roadside units, and smart vehicles. We formulate the joint optimization problem as a double time-scale Markov decision process (DTS-MDP), based on the fact that the time-scale of content timeliness changes less frequently as compared to the vehicle mobility and network states during the content delivery process. At the beginning of the large time-scale, the content placement/updating decision can be obtained according to the content popularity, vehicle driving paths, and resource availability. On the small time-scale, the joint vehicle scheduling and bandwidth allocation scheme is designed to minimize the content access cost while satisfying the constraint on content delivery latency. To solve the long-term mixed integer linear programming (LT-MILP) problem, we propose a nature-inspired method based on the deep deterministic policy gradient (DDPG) framework to obtain a suboptimal solution with a low computation complexity. The simulation results demonstrate that the proposed cooperative caching system can reduce the system cost, as well as the content delivery latency, and improve content hit ratio, as compared to the noncooperative and random edge caching schemes.**

***Index Terms*—Content delivery, content placement, cooperative edge caching, deep deterministic policy gradient (DDPG), double time-scale Markov decision process (DTS-MDP), vehicular edge computing and networks.**

I. Introduction

SMART vehicles accessing popular content and sharing transportation information are evolving as an emerging paradigm to support the advanced driver assistance systems and self-driving. Advances in sensing and artificial intelligence technologies have enabled innovative vehicular applications for enhancing driving safety and travel comfort [1], [2]. Meanwhile, these applications are resulting in a growing demand for communication, computation and storage resources. The low content access latency and diverse application requirements may not be satisfied if the contents are fetched from remote data centers. Fortunately, the vehicular edge computing and networks can serve as an effective framework for alleviating strain in the backhaul links by migrating the cloud caching servers to edge devices [3].

The existing mobile edge caching strategies in heterogeneous networks are mainly divided into four categories [4], i.e., caching in macro-cell base stations (MBS) [5], caching in small-cell base stations (SBSs) [6], caching in device-to-device (D2D) communication networks [7], and caching in mobile devices [8]. Nevertheless, these edge caching strategies cannot be directly applied in vehicular edge caching networks. Based on the assumption of low-speed moving scenarios, it is valid to access the whole content during the connection period within the coverage area of SBSs or in the range of the D2D communication networks. However, vehicles with high-speed mobility may pass several roadside units (RSUs) in the content delivery process, and the quality of vehicle-to-vehicle (V2V) communication links may not always remain stable. Thus, the whole content may not be obtained from a single edge caching node. On the other hand, most of the existing work regards content placement and content delivery as two separate subproblems. Wang *et al.* [9] analyzed a social-aware edge caching to improve the spectrum utilization in fog access networks. To improve the efficiency of content delivery in Internet of Vehicles (IoV), Yao *et al.* [10] proposed a caching scheme based on vehicular mobility pattern. The main idea of this strategy is to store, carry, and forward popular contents using vehicles as relay nodes. However, the above work does not consider the joint optimization of content placement and content delivery to enhance caching performance. These issues necessitate the design of a joint content placement and content delivery in vehicular edge caching network.

Manuscript received April 27, 2019; revised July 11, 2019 and August 25, 2019; accepted September 21, 2019. Date of publication October 22, 2019; date of current version January 10, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFC0807101, in part by the Science and Technology Program of Sichuan Province, China, under Grant 2019YFH0007, in part by the Fundamental Research Funds for the Central Universities, China, under Grant ZYGX2016Z011, and in part by the EU H2020 Project COSAFE under Grant MSCA-RISE-2018-824019. *(Corresponding author: Supeng Leng.)*

G. Qiao and S. Leng are with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: qghuestc@126.com; spleng@uestc.edu.cn).

S. Maharjan is with the Center for Resilient Networks and Applications, Simula Metropolitan Center for Digital Engineering, 0167 Oslo, Norway (e-mail: sabita@simula.no).

Y. Zhang is with the Department of Informatics, University of Oslo, 0316 Oslo, Norway (e-mail: yanzhang@ieee.org).

N. Ansari is with the Advanced Networking Laboratory, Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: nirwan.ansari@njit.edu).

Digital Object Identifier 10.1109/JIOT.2019.2945640

In this article, we investigate the cost-optimal cooperative caching in vehicular edge caching networks with limited storage capacity and bandwidth resource, taking account of time-varying content popularity, dynamic network topology, and vehicle driving paths. Based on the fact that the time-scale of content timeliness is much larger than that change of vehicle mobility and network states. The cooperative caching is modeled as a double time-scale edge caching process composed of content placement and content delivery. RSUs and smart vehicles collaboratively cache/update different types of contents aiming for minimal storage cost on the large time-scale. On the small time-scale, vehicle scheduling and bandwidth allocation can be jointly optimized to reduce the content access cost while satisfying the required delivery latency. However, the joint optimization problem in a realistic transportation environment is challenging. First, although the vehicle driving paths between origin and destination can be emulated based on real transport data, it is difficult to predict the exact location information of vehicle in different transport regions. Second, the joint content placement and content delivery problem is a long-term mixed integer linear programming (LT-MILP), which has been proven to be NP-hard [11]. Moreover, variable participation of vehicles and ephemeral interactions will increase the operation complexity of edge caching system. Thus, it is difficult to obtain the optimal solution in a tolerable time when the system state space becomes large. These inherent characteristics of vehicular networks require the adaptability of cooperative edge caching by utilizing some efficient solutions.

Recently, the nature-inspired approaches like evolutionary game, swarm intelligence, artificial neural networks, and reinforcement learning have been inspired by the biological system or a human brain [12]. Smart agents in these complex systems can imitate the best features or interact with the dynamic environment to learn a series of intelligence behaviors. With the aid of intelligence of natural-inspired approaches, we design a deep reinforcement learning (DRL)-based cooperative caching scheme to provide a low-complexity decision making and adaptive resource management. Particularly, we leverage a deep deterministic policy gradient (DDPG) learning algorithm to cope with continuous-valued control decision.

Below are the main contributions of this article.

1) We design a novel edge caching framework based on the cooperation among base station, RUSs, and vehicles. The cooperative caching problem is modeled as a double time-scale Markov decision process (DTS-MDP). The policy decisions of content placement/updating, vehicle scheduling, and bandwidth allocation occur on different time-scale, which have not been studied in the previous researches.
2) Based on the real data sets, the destination of vehicle can be predicted accurately and vehicle driving paths between origin and destination can be simulated as well. According to the obtained vehicle densities in different transport regions, the content placement can be decided in advance for the improvement of content hit ratio.
3) We propose a DDPG-based cooperative caching scheme by integrating deep neural networks and reinforcement learning approaches. The system agent can perform the joint caching decision from the historical experiences. Moreover, the proposed scheme applies deterministic policy and mini-batch gradient descent to accelerate the learning speed and improve caching performance.

The reminder of this article is organized as follows. Section II summarizes related work. In Section III, we present system model of cooperative edge caching. In Section IV, we formulate the joint optimization problem as a DT-MDP. We introduce a DDPG-based approach for cooperative caching in Section V. Section VI presents simulation results. Section VII concludes this article and future work.

II. Related Work

Many excellent works have investigated mobile edge computing and caching approach in wireless networks. Jiang *et al.* [13] proposed a cooperative content assignment and delivery framework to minimize average content downloading latency. In [14], a cooperative content caching approach among small cells was proposed, for which the tradeoff between content delivery latency and storage cost was investigated. In [15], mobile devices were clustered to provide unused storage resource in terms of minimizing the average content access latency. Yao and Ansari [16] jointly optimized the content placement and storage allocation for Internet of Things to minimize the network traffic cost.

Vehicular content network has emerged as a promising paradigm to support diverse vehicular applications. In [17], an efficient caching scheme was proposed based on the similarity and population of user community in a V2V communication scenario to improve content hit rate. Abdelhamid *et al.* [18] proposed a caching-assisted data delivery scheme to leverage massive caching servers deployed on the roadside for collaboratively collecting content and supporting vehicular applications. To tackle the data loss caused by vehicular mobility, Hu *et al.* [19] proposed a vehicle-assisted content caching scheme to assign content at every exit and entrance of the road segments. The cached information can be shared between leaving and incoming vehicles.

Due to the stochastic characteristics of content popularity and vehicle mobility, the decision-making in caching system is a stochastic optimization. In addition, the complexity of the caching system is very high in joint content placement and content delivery. The nature-inspired approaches are a set of metaheuristic algorithms to effectively overcome these problems and can find near-optimal solutions. These approaches have been successfully applied in IoV due to their intelligence and adaptability. Thereinto, DRL has been utilized as an efficient technique for content caching in vehicular networks. Zhang *et al.* [20] jointly optimized content caching and rate adaption to improve quality of experience (QoE). In order to solve the problem of continuous action variables in MDP model, Li *et al.* [21] used deterministic policy gradient learning algorithm to provide the optimal resource pricing strategy. Sadeghi *et al.* [22] introduced the concepts of global and local content popularity, and proposed a novel RL-based content caching scheme to execute the optimal caching policy.

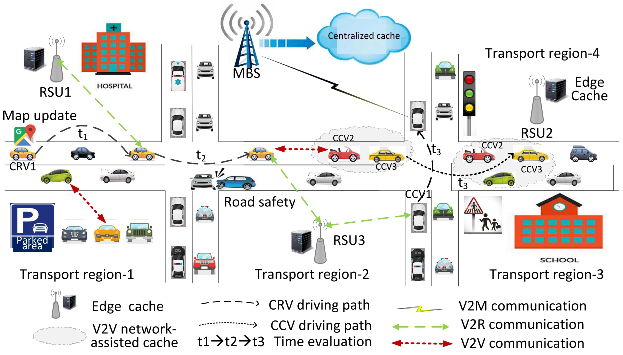

Fig. 1. Cooperative caching framework based on vehicle driving path.

The above studies are based on the assumption that the whole content can be fetched by vehicles, which are associated to corresponding RSUs or connected with other vehicles. This is not always feasible in practice due to dynamic network topology. In addition, the cooperation between edge caching and V2V-assisted caching in content placement and delivery processes is an unexplored topic. To fill this research gap, we focus on the cooperative edge and vehicle-assisted caching, and present a natural-inspired learning approach to solve the complex joint optimization problem.

TABLE I
SUMMARY OF KEY NOTATIONS

Notation	Description
$\mathcal{N}$	Index set of all edge caching nodes (MBS, RSUs and CCVs)
$\mathcal{K}$	Index set of edge caching nodes (RSUs)
$\mathcal{V}$	Index set of contents caching vehicles (CCVs)
$\mathcal{M}$	Index set of contents requesting vehicles (CRVs)
$\mathcal{F}$	Set of all available contents
$\mathcal{X}$	Index Set of content placement/updating period
$\mathcal{Y}$	Index set of content delivery time slot
$\mathcal{F}_k^{rsu}$	Index set of available contents cached in RSU $k \in \mathcal{K}$
$\mathcal{F}_v^{ccv}$	Index set of available contents cached in CCV $v \in \mathcal{V}$
$\mathcal{L}$	Index set of transport regions on which CRVs and CCVs driving or parked
t_x, t_x^y	Content caching period and content fetching step.

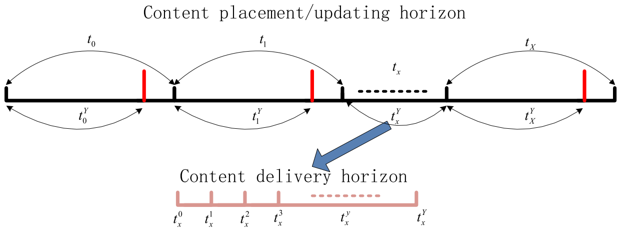

Fig. 2. Double time-scale content caching model with finite number of content fetching time slots t_x^y at the content placement/updating period t_x.

III. SYSTEM MODEL

In this section, we propose the cooperative caching framework including network model, double time-scale caching model, vehicular mobility pattern, and communication model.

A. Network Model

We consider the vehicular edge computing and network with different types of edge caching nodes, including an MBS, several RSUs, and content caching vehicles (CCVs) with the unutilized storage resource. Let $\mathcal{N} = \{0, 1, \ldots, K, K+1, \ldots, N\}$ represent the index set of edge caching nodes, 0 is the index of MBS, $\mathcal{K} = \{1, \ldots, K\}$ is the index set of RSUs, and $\mathcal{V} = \{K+1, \ldots, N\}$ is the index set of CCVs to provide V2V-assisted content caching, respectively. Let $\mathcal{M} = \{1, 2, \ldots, M\}$ denote the index set of content requesting vehicles (CRVs) to seek their desired contents. Let $\mathcal{F} = \{1, \ldots, F\}$ denote the index set of available contents. Considering the abundant storage capacity, we assume MBS is the centralized content provider to cache all available contents. In addition, each edge caching node RSU $k \in \mathcal{K}$ and CCV $v \in \mathcal{V}$ are equipped with a content caching unit of limited storage resource, which can cache $\mathcal{F}_k^{rsu} \subset \mathcal{F}$ and $\mathcal{F}_v^{ccv} \subset \mathcal{F}$, respectively. MBS is deployed at the center of the road network to provide service continuity and RSUs are randomly located at the center of different transport regions. Table I lists the key notions and descriptions of our proposed caching framework.

Fig. 1 shows a graphic illustration of a cooperative edge caching framework. During the content delivery process, CRV1 requires a desired content (e.g., navigation map update, software download, etc.) and wants to pay the minimum resource usage cost while also satisfying its requirement on content delivery latency. Based on its driving paths, CRV1 can make different choices to fetch a segment of the whole content. Specifically, if the desired content is cached in itself, CRV1 can obtain the whole content without access cost and delivery latency. Otherwise, it will seek a segment of content from an RSU in its region or within the communication range of CCVs. For example, CRV1 can associate with RSU3 (V2R) or connect with CCV2 (V2V) in transport region-2. Based on the wireless link quality and content access cost, CRV1 will select the optimal cache node to fetch the segment of whole content. If there is no desired content in the attached RSU or CCVs, CRV1 determines whether to fetch the segment of the content through MBS (V2M) or just keep idle at the current time slot. Intuitively, the cooperative edge caching can not only provide users with a continuous service experience but also reduce system cost by implementing the joint optimization of content updating and content delivery.

B. Double Time-Scale Content Caching Model

Based on the fact that the time-varying scale of content timeliness and popularity is much larger than the content delivery process caused by the vehicle mobility and channel change, we modeled the cooperative edge caching as a double time-scale caching model. As illustrated in Fig. 2, the cooperative content caching process takes place in the different time granularities: 1) content placement/updating horizon and 2) content delivery horizon composed of several content fetching time slots. In the large time-scale, let $\mathcal{X} = \{0, 1, \ldots, X\}$ denote the index set of content placement/updating decision made at the

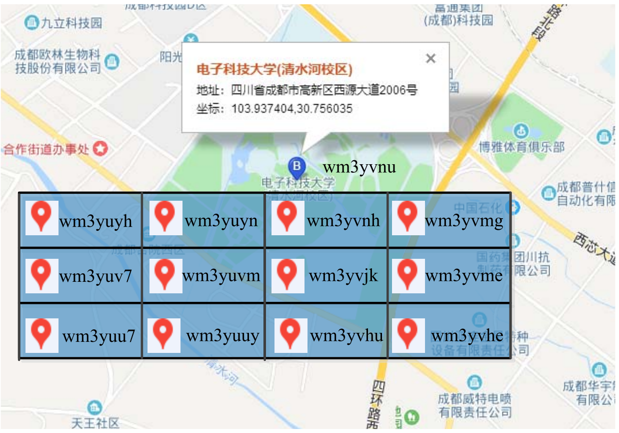

Fig. 3. Division of transport regions based on Geohash algorithm.

beginning of caching period t_x, $\{t_x | x \in \mathcal{X}\}$. Furthermore, each content caching period can be slotted as a set of small time slots, which can be indexed by the set $\mathcal{Y} = \{0, 1, \ldots, Y\}$. The joint decision of vehicle scheduling and bandwidth allocation is implemented at the beginning of content delivery time slot t_x^y, $\{t_x^y | x \in \mathcal{X}, y \in \mathcal{Y}\}$, where t_x^Y is the maximal content delivery latency to all CRVs.

Each content is identified with three features, which is defined as $[\rho_f, \nu_f, d_f], f \in \mathcal{F}$, ρ_f and ν_f are the popularity of content f and size of content f, respectively, and d_f is the maximum allowed access latency to obtain content f. Let $\boldsymbol{\beta}(t_x^0) \in \{0, 1\}^{N \times F}$ denote the $N \times F$ binary matrix of content updating at time slot t_x. That is, $[\beta(t_x^0)]_{k \times f} = 1$ and $[\beta(t_x^0)]_{v \times f} = 1$ indicate that the content f is cached in RSU k and CCV v, respectively. Otherwise, $[\beta(t_x^0)]_{k \times f} = 0$ and $[\beta(t_x^0)]_{v \times f} = 0$. In addition, we have $[\beta(t_x^0)]_{0 \times f} = 1$ due to all available contents can be cached in the MBS.

Most of existing work assumes that content popularity follows the Zipf distribution in mobile social networks [23]. However, the assumption may not always be true in vehicular networks as smart vehicles pass several RSUs during the content fetching process. The statistical information of local content popularity cannot properly reflect the actual content access requirements. Meanwhile, the modeling of content popularity still requires further investigation in vehicular networks. In order to provide a reasonable model of content popularity, we define a global content popularity derived from content requesting rate. Specifically, at the beginning of the content caching period, content requesting information of the CRVs will be sent to the MBS, which is responsible for calculating the content popularity. Based on the received content access requests from CRVs, MBSs compute and update the global content popularity at the caching period t_x

$$\rho_f(t_x) := \frac{\varrho^f(t_x)}{M}, \quad f \in \mathcal{F} \tag{1}$$

where $\varrho^f(t_x)$ is the total number of requests for content f.

C. Vehicular Mobility Pattern

To facilitate modeling of the vehicular mobility pattern, we use discrete geographical regions named transport regions instead of continuous locations. As shown in Fig. 3, a typical road networks can be divided into L discrete transport regions. Let $\mathcal{L} = \{1, \ldots, L\}$ represent the index set of transport regions. In this article, the Goehash algorithm [24] is used to convert the latitude and longitude coordination on the global position system (GPS) map into a string representing different transport regions. Specifically, each string represents a rectangular transport region and all the vehicles driving through the same region can be labeled the same string. Considering the actual coverage range of RSUs and V2V communication, the seven-bit string can well match the transport region with an accuracy of hundreds of meters.

In each content caching period, it is difficult to accurately predict the vehicle location when vehicle drives through the transport regions. Nevertheless, combined with coverage of RSUs and V2V communication range, the prediction of vehicle destination can be obtained and the driving paths between origin and destination can also be simulated through the advanced data mining based on the real transport data. In the adjacent content fetching time intervals, CRVs and CCVs can only drive to the neighboring transport regions along with the origin and destination, and the transition probability follows standard distribution, such as uniform or normal distribution. Based on the aforementioned scenario, we define the following vehicular mobility pattern as:

$$q\left(l_j' | l_j\right) = \begin{cases} \mu, & l_j' = l_j, j \in \mathcal{M} \text{ or } j \in \mathcal{V} \\ \eta_e, & \text{otherwise} \end{cases} \tag{2}$$

where

$$\mu + \sum \eta_e = 1, \quad e \in N, S, E, W \tag{3}$$

where e represents one of four possible driving directions [i.e., north (N), south (S), east (E), and west (W)]. The new location l_j' of vehicle j depends only on the past location l_j, $l_j \in \mathcal{L}$. Vehicle j may stay in a transport region in the next time slot with probability μ. Accordingly, vehicle j can randomly drive into an adjacent transport region with probability η_e.

D. Communication Model

In this section, we formulate the communication topology between CRVs and different types of cache nodes. Without loss of generality, we consider that network topology remains constant during one content fetching time slot.

Let $\alpha_{i,j,f}(t_x^y)$ be the association indicator between CRV j and edge cache node i, $i \in \mathcal{N}$ at time slot t_x^y, where $\alpha_{i,j,f}(t_x^y) = 1$ means that CRV j fetches content f from cache node i, otherwise $\alpha_{i,j,f}(t_x^y) = 0$. Moreover, CRV can only choose one type of communication mode at one time slot.

In the cellular and V2V communication modes, we consider that the MBS allocates the orthogonal spectrum resources to CRVs such that there is no interference between cellular and V2V communication. The signal-to-noise ratio (SNR) at time slot t_x^y can be formulated as

$$\gamma_{i,j}\big(t_x^y\big) = \frac{p_{i,j}\big(t_x^y\big) g_{i,j}\big(t_x^y\big)}{\xi_{i,j}\big(t_x^y\big) d_{i,j}\big(t_x^y\big)^{\kappa} \sigma_{i,j}\big(t_x^y\big)^2}, \forall i \in \mathcal{N} \tag{4}$$

where $p_{i,j}(t_x^y)$ is the transmission power from cache node i to CRV j, $d_{i,j}(t_x^y)$ is the distance between CRV j and cache node i, $g_{i,j}(t_x^y)$ is the antenna gain at cache node i, $\xi_{i,j}(t_x^y)$ and κ are path loss at a reference unit distance and path loss exponent, respectively, and $\sigma_{i,j}(t_x^y)^2$ is the power of additive white Gaussian noise. For the WLAN communication mode, we mainly consider a contention-free channel access [25]–[27]. In this case, the SNR at CRV k is similar to (4).

The available spectrum resource is denoted as W_0^{mbs} Hz for MBS, W_k^{rsu} Hz for RSU k and W_v^{ccv} for CRV v, $w_{i,j}(t_x^y)$ can be allocated as the continuous bandwidth resource by edge node $i, i \in \mathcal{N}$. Thus, the data rate to fetch a segment of content f between CRV j and cache node i is given by

$$\begin{aligned} r_{i,j,f}\big(t_x^y\big) &= \alpha_{i,j,f}\big(t_x^y\big) w_{i,j}\big(t_x^y\big) \\ &\quad \times \ \log_2\big(1 + \gamma_{i,j}\big(t_x^y\big)\big), i \in \mathcal{N}, j \in \mathcal{M}, f \in \mathcal{F}. \end{aligned} \tag{5}$$

IV. Problem Formulation

In this section, we formulate the joint optimization of cooperative caching as a DTS-MDP and give the detailed definitions of each component of DTS-MDP.

A. Caching System State Space

At the beginning of each caching time period, the smart agent obtains vehicular content request information, which includes a list of transport regions, content features, and network states. Specifically, the system state space contains the following.

1) $f_j(t_x)$, $f_j(t_x) \in \mathcal{F}$*:* Requesting content type of CRV j at the caching period t_x.
2) $o_{j,f}(t_x)$, $o_{j,f}(t_x) \le d_f$*:* Content delivery deadline of CRV j for accessing content f at caching period t_x.
3) $b_{j,f}(t_x^y)$, $b_{j,f}(t_x^y) \in [0, \nu_f]$*:* Remaining size of requesting content f at time slot t_x^y, $b_{j,f}(t_x^y) \in [0, \nu_f]$.
4) $l_j(t_x^y)$, $l_v(t_x^y)$*:* Index of transport region in which CRV j or CCV v is driving at time slot t_x^y, respectively.
5) $\rho_f(t_x)$*:* Content popularity of f in caching period t_x.
6) $\beta_{i,f}(t_x)$*:* Caching indicator on whether cache node i stores content f at caching time period t_x, $i \in \mathcal{K} \cap \mathcal{V}$.

Let $\mathbf{s}(t_x^y) \in S$ denote the joint system state in our cooperative caching system, i.e.,

$$\mathbf{s}\big(t_x^y\big) = \big\{[\mathbf{f}(t_x)], [\mathbf{o}(t_x)], \big[\mathbf{b}\big(t_x^y\big)\big], \big[\boldsymbol{\rho}(t_x)\big], \big[\mathbf{l}\big(t_x^y\big)\big], [\boldsymbol{\beta}(t_x)]\big\}. \tag{6}$$

B. Caching System Action Space

After receiving the content requests, the MBS calculates the content popularity and then decides which contents should be stored to which cache nodes. At each content fetching time slot, a joint vehicle scheduling and bandwidth allocation is used to reduce the content access cost subject to the content delivery deadline. Thus, the action space contains the following.

1) $\beta_{i,f}(t_x)$*:* Whether the content f is saved in the caching period t_x or not, i.e., $\beta_{i,f}(t_x)$ will be one if the content is cached, $i \in \mathcal{K} \cap \mathcal{V}$.
2) $\alpha_{i,j,f}(t_x^y)$*:* Association indicator on whether CRV j fetches a segment of content f from edge cache node i, $i \in \mathcal{N}$.
3) $w_{i,j}(t_x^y)$*:* Allocated bandwidth when CRV j associates with cache node i at time slot t_x^y, $i \in \mathcal{N}$.

Let $\mathbf{a}(t_x^y) \in A$ denote the joint action space in the cooperative content caching system, i.e.,

$$\mathbf{a}\big(t_x^y\big) = \big\{[\boldsymbol{\beta}(t_x)], \big[\boldsymbol{\alpha}\big(t_x^y\big)\big], \big[\mathbf{w}\big(t_x^y\big)\big]\big\}. \tag{7}$$

C. State Transition Probabilities

The probabilities associated with different system states changes are called state transition probabilities. We denote transition probability as

$$\Pr\Big(\mathbf{s}\Big(t_x^{y+1}\Big)|\mathbf{s}\big(t_x^y\big), \mathbf{a}\big(t_x^y\big)\Big). \tag{8}$$

Equation (8) is the probability of state $\mathbf{s}(t_x^{y+1})$ if action $\mathbf{a}(t_x^y)$ is chosen at system state $\mathbf{s}(t_x^y)$.

D. Cost Function

The objective of the service provider is to minimize the storage cost. Each cache node updates its stored contents and broadcasts this information to all the CRVs. CRVs aim to obtain the desired contents at low content access cost within the fetching deadline. To realize these objectives, we design the following cost functions.

1) Content Storage Cost: The storage cost $c_{1,i}$ corresponds to the cost of updating the contents. That is, for cache node i, the storage cost can be regarded as the operations required for caching the number of contents at caching period t_x but not cache at the (t_{x-1})th period. Thus, $c_{1,i}$ is a function of action $\boldsymbol{\beta}(t_{x-1})$ and upcoming action $\boldsymbol{\beta}(t_x)$.

$$c_{1,i}\big(\boldsymbol{\beta}_i(t_x), \boldsymbol{\beta}_i(t_{x-1})\big) = \lambda_1\big(F_i - f_{cou}\big\{\boldsymbol{\beta}_i(t_x) \cap \big(\boldsymbol{\beta}_i(t_{x-1})\big)\big\}\big) \tag{9}$$

where the function $f_{cou}\{(a)\}$ represents the statistics of the number of identical contents in the cache node between the caching period t_{x-1} and t_x.

We introduce the content access cost to pay for cache node i if the CRV j fetches a segment of desired content f.

2) Content Access Cost: We assume that the service provider adopts resource-usage-based pricing, which is widely used in China and USA [28]. The price per unit bandwidth for cache node i is p_i^b. Thus, the content access cost of CRV j paying for cache node i is given by

$$c_{2,i,j,f}\big(s\big(t_x^y\big), a\big(t_x^y\big)\big) = \lambda_2 p_i^b \min\big(b_{i,j,f}\big(t_x^y\big), \phi_{i,j,f}\big(t_x^y\big)\big) \tag{10}$$

where

$$\phi_{i,j,f}\big(t_x^y\big) = r_{i,j,f}\big(t_x^y\big)\Delta t, \ i \in \mathcal{N}, j \in \mathcal{M}, f \in \mathcal{F} \tag{11}$$

$r_{i,j,f}(t_x^y)$ is the data rate defined in (5) and Δt is the time required to fetch content f. In addition, (11) is the size of the obtained content at current time slot.

3) Penalty Cost: If the whole content is not obtained before the fetching deadline, the penalty cost for system will be

$$c_{3,j,f}\big(s\big(o_{j,f}(t_x)\big)\big) = \lambda_3 \rho_{i,f}\big(o_{j,f}(t_x)\big) \tag{12}$$

where $\rho_{j,f}(o_{j,f}(t_x))$ is the popularity of content f and $o_{j,f}(t_x)$ represents the fetching deadline of CRV j to acquire content f. λ_1–λ_3 are the weight factor to represent the impact on different cost functions, respectively. With the system state $\boldsymbol{s}(t_x^y)$ and action $\boldsymbol{a}(t_x^y)$ taken at time slot t_x^y, the aggregate system cost can be expressed as

$$\begin{aligned} c\big(\boldsymbol{s}\big(t_x^y\big), \boldsymbol{a}\big(t_x^y\big)\big) = &\sum_{i\in\mathcal{N}} c_{1,i}(\boldsymbol{s}(t_x), \boldsymbol{a}(t_x)) \\ &+ \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}}\sum_{f\in\mathcal{F}} c_{2,i,j,f}\big(\boldsymbol{s}\big(t_x^y\big), \boldsymbol{a}\big(t_x^y\big)\big) \\ &+ \sum_{j\in\mathcal{M}}\sum_{f\in\mathcal{F}} c_{3,j,f}\big(\boldsymbol{s}\big(t_x^y\big), \boldsymbol{a}\big(t_x^y\big)\big). \end{aligned} \tag{13}$$

E. Value Function

The objective of cooperative caching is to achieve an optimal tradeoff between the total caching cost and the average content delivery latency. We define $\pi : S \to A$ as a stationary policy, which realizes the mapping from current system state to a series of actions, e.g., $\boldsymbol{a} = \pi(\boldsymbol{s})$. In addition, we denote Π as a set of all stationary policies. For any initial system state $\boldsymbol{s}$ and corresponding policy $\pi \in \Pi$, the cumulative system cost from 0 to X is given by

$$\min_{\pi\in\Pi} \sum_{x=0}^{X}\sum_{y=0}^{Y} \gamma^{t_x^y} c\big(\boldsymbol{s}^{\pi}\big(t_x^y\big), \boldsymbol{a}\big(t_x^y\big)\big)\big) \tag{14}$$

where X indicates that the caching system runs X episodes to represent the number of content caching periods, and each caching period has Y time slots, $\gamma \in [0, 1]$ is the discounted factor. Here, π^* is the optimal policies at time slot t_x^y. The value function $V(\cdot)$ can be defined as

$$V^*(\boldsymbol{s}) = \min_{\boldsymbol{a}\in A}\left\{ c(\boldsymbol{s}, \boldsymbol{a}) + \gamma^{t_x^y} \sum_{\boldsymbol{s}'\in S} \Pr\big(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}\big) V^*\big(\boldsymbol{s}'\big) \right\}. \tag{15}$$

For simplicity, $\boldsymbol{s}$ and $\boldsymbol{a}$ are the current system state and caching action at the time slot t_x, respectively, and $\boldsymbol{s}'$ is the next system state at the time slot t_x^{y+1}. Equation (15) is the Bellman equation. In general, the Bellman equation can be solved by traditional value or policy iteration approach. However, it is challenging to solve (15) due to the following reasons.

1) Although the state transition probability is defined in (8), the whole mapping relation cannot be used directly to solve the Bellman equation since it is difficult to precisely estimate the state transition in the real transportation and networks environment.
2) As a complicated DTS-MDP problem and high-dimensional system state space, the extremely high computation complexity using traditional approaches will be a huge drawback limiting its applicability in a practical caching system. Thus, we need to find an effective approach to address these issues.

V. Deep Reinforcement Learning for Cooperative Content Caching

Reinforcement learning is an important branch of machine learning that can make a smart agent learn from a set of optimal policies in a dynamic environment by minimizing/maximizing the expected cumulative cost/reward. There are two major categories of RL algorithms: 1) the model-based approach and 2) the model-free approach. The former is mainly used in the automatic control field. Generally, the Gaussian process or the Bayesian network is used to model this type of problems. The model-free framework can be regraded as a data-driven approach to obtain an optimal policy by estimating value function or policy function. In this article, we focus on the model-free learning approach in order to provide a training guidelines based on a large number of historical experiences. The model-free approach can be divided into three types: 1) critic-model (value-based approach); 2) actor-model (policy-based approach); and 3) actor-critic learning approach. Critic-model, such as Q-learning, is a classical RL learning algorithm that obtains optimal policies based on the estimated action-value function $Q(s, a)$. Actor-model can learn a stochastic policy function $\pi_{\varpi}(s|a)$ with network parameter ϖ. Konda and Tsitsiklis [29] combined the actor-model and the critic-model in order to address the individual disadvantages of policy gradient and value predication scheme. The actor-critic approach exhibits good convergence properties with continuous action spaces. However, the good performance is based on a large amount of training samples. In order to solve the challenge, Silver *et al.* [30] developed a deterministic policy gradient approach that can directly learn deterministic policy $\mu(s)$ instead of the stochastic stationary policy distribution $\pi(a|s)$.

Based on the deterministic actor-critic model, we leverage deep neural networks to provide accurate estimation of deterministic policy function $\mu(s)$ and value function $Q(s, a)$. The combination can be utilized to implement our proposed joint optimization of content caching and resource allocation problem. The schematic of the DDPG is shown in Fig. 4 [31]. There are three major modules of the RL agent: 1) primary networks; 2) target networks; and 3) replay memory. Thereinto, deep actor-network and deep critic-network are key models in the primary networks and target networks, respectively. In the following sections, we will discuss the detailed principles of each module and present our proposed cooperative caching algorithm.

A. Deep Critic-Networks Training and Update

The critic-network of DRL is responsible for evaluating policies based on the action-value function $Q(s, a)$, which should satisfy the following condition:

$$Q(s, a) = c(s, a) + \gamma \sum_{s'\in S} \Pr\big(s'\big|s, a\big) V^*\big(s'\big) \tag{16}$$

where $c(s, a)$ is the instantaneous cost function in (13) and $V^*(s')$ is the optimal value function in (15).

The Q-learning algorithm is prone to the curse of dimensionality with the increasing number of system states and a large number of vehicles. In addition, $Q(s, a)$ cannot be

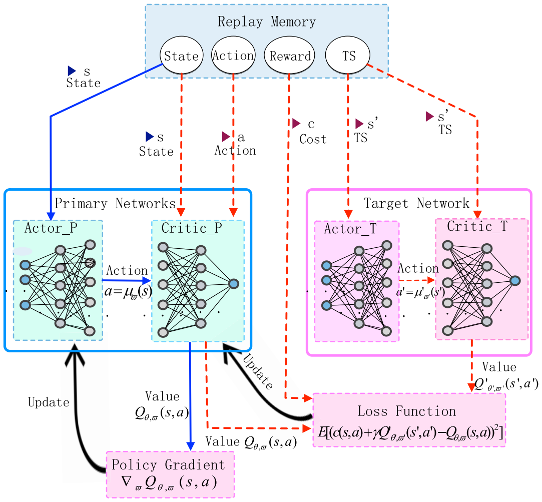

Fig. 4. Schematic of the DDPG learning algorithm.

directly calculated due to continuous action space. Therefore, we introduce an approximator based on a deep neural network to estimate the action-value function and update parameters using the history experience stored in replay memory buffer. The Q-function can be approximated by parameter matrix $\boldsymbol{\theta}$ and denoted as $Q_{\boldsymbol{\theta}}(s,a) \approx Q(s,a)$, the mapping function of any two adjacent layers of deep critic-network is give by

$$\boldsymbol{\theta}^T \cdot \Phi(s,a) = \sum_{h=1}^{H} \sum_{h'=1}^{H'} (\theta_{hh'}\phi_{mh}(s,a) + \theta_{0h'}). \tag{17}$$

For the mth training sample, $\Phi = (\phi_{m1}(s,a), \ldots, \phi_{mH}(s,a))$ is the activation function, $\boldsymbol{\theta}$ is the parameter matrix of deep neural network, $[\theta_{01}, \ldots, \theta_{0H'}]$ is the vector of bias value and the $[\theta_{h1}, \ldots, \theta_{hH'}]$ is the vector of weighted values, and H and H' are the number of hidden units of the adjacent layers of critic network, respectively.

In order to solve the exploration problem of deterministic policy, we leverage the off-policy learning method to differentiate the sampled policies and advanced policies. Specifically, we construct a replay memory to store a series of historical experiences $[s, a, c, s']$. This operation can avoid sample-correlation during the training process. To improve the efficiency of the learning algorithm, we utilize the stochastic method to updates parameters. More specifically, the network parameters can be updated by randomly choosing a mini-batch sample from replay memory. We define the size of replay memory buffer is B, and denote the set $\mathcal{D} = \{1, 2, \ldots, m, \ldots, D\}$ is the index set of training samples. The temporal difference error is given by

$$\delta_{TD} = c(s,a) + \gamma Q'_{\boldsymbol{\theta}'}\left(s', a'\right) - Q_{\boldsymbol{\theta}}(s,a) \tag{18}$$

where $Q'_{\boldsymbol{\theta}'}(s',a')$ is the target action-value function derived from the target critic-network and $\gamma = [0,1]$ is the learning rate. The training phase and parameter updating process of the target network will be discussed in the following section. Then, we update parameter $\boldsymbol{\theta}$ in the primary critic-network using mini-batch gradient descent, i.e.,

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \eta_{\boldsymbol{\theta}} \delta_{TD} \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}(s,a) \tag{19}$$

where $\eta_{\boldsymbol{\theta}}$ is a positive constant reflecting the learning rate for the value function evaluation. The small learning rate can avoid oscillation but it may result in a lager number of iterations for performance convergence.

B. Deep Actor-Networks Training and Update

Instead of learning stochastic policy distribution $\pi(a|s)$, the actor-network can learn a deterministic policy function $a = \mu_{\varpi}(s)$ with parameter vector $\boldsymbol{\varpi} = (\varpi_0, \varpi_1, \varpi_2, \ldots, \varpi_n)$. Likewise, we introduce a deep neural network as an approximator of deterministic policy function in which the mapping function of any two adjacent layers of deep actor-network can be expressed by

$$\boldsymbol{\varpi}^{\boldsymbol{T}} \cdot \boldsymbol{\Psi}(s,a) = \sum_{h=1}^{H} \sum_{h'=1}^{H'} (\varpi_{hh'}\psi_{mh}(s,a) + \varpi_{0h'}). \tag{20}$$

For the mth training sample, $\Psi = (\psi_{m1}(s), \ldots, \psi_{mH}(s))$ is the activation function, $\boldsymbol{\varpi}$ is the parameter matrix of deep actor neural network, $[\varpi_{01}, \ldots, \varpi_{0H'}]$ is the vector of bias value, and the $[\varpi_{h1}, \ldots, \varpi_{hH'}]$ is the vector of weighted values.

Our objective is to minimize the joint optimization problem defined in (14) by leveraging the stochastic gradient descent algorithm. Silver *et al.* [30] proved that the gradient of the objective function with respect to parameters $\boldsymbol{\varpi}$ is equal to solving the gradient of Q-function. Thus, the gradient of the objective function is given by

$$\nabla_{\boldsymbol{\varpi}} Q_{\boldsymbol{\theta},\boldsymbol{\varpi}}(s,a) = \frac{\partial \mu_{\boldsymbol{\varpi}}(s)}{\partial \boldsymbol{\varpi}} \frac{\partial Q_{\boldsymbol{\theta}}(s,a)}{\partial a}\bigg|_{a=\mu_{\boldsymbol{\varpi}}(s)}. \tag{21}$$

The global/local minimal Q-value can be found by the mini-batch gradient descent policy. The parameter update of policy gradient can be expressed as

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

where $\eta_{\boldsymbol{\varpi}} > 0$ is a positive constant representing learning rate.

C. Policy Exploration and Target Networks Updating

1) Policy Exploration: A major challenge of reinforcement learning is the credit assignment problem. Most existing work has demonstrated that it will yield poor performance if the RL agent strongly focuses on either exploitation or exploration. We introduce policy exploration to obtain good training performance. Instead of the traditional greed exploration and random sampling of Gaussian distribution, the Ornstein–Uhlenbeck noise mechanism [32] is used to implement policy exploration with a continuous action space.

2) Target Networks Updating: As shown in Fig. 4, the target networks can be regarded as "approximate copy" of primary networks. Actually, the deep target network architecture, such as the number of layers and hidden units, need to be consistent with the deep primary networks. Additionally, the learning performance can be stable and robust if the target policy function and value function are updated slowly compared to the primary networks. We use the exponentially weighted moving average (EWAM) scheme to update network parameters $\boldsymbol{\theta}'$ and $\boldsymbol{\varpi}'$ instead of copying the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\varpi}$

Algorithm 1 DDPG-Based Cooperative Caching Algorithm

```
**Parameters** $\gamma, \eta_{\boldsymbol{\theta}}, \eta_{\boldsymbol{\varpi}}, \tau_{\boldsymbol{\theta}'}, \tau_{\boldsymbol{\varpi}'}, X, Y, B, D$.
**Initialize** primary networks and target networks.
**Repeat** (each episode-content caching period):
  **Initialize** system sate $s(t_x^y)$.
  **Execute** (content fetching step of current episode):
    **Perform** policy exploration $\mu(s)$ using Ornstein-Uhlenbeck noise method.
    **Sample** transport regions and network environment.
    **Observe** feedback tuple $s(t_x^{y+1}), c(t_x^y)$.
    **Store** $s(t_x^y), a(t_x^y), c(t_x^y), s(t_x^{y+1})$.
    **Sample** $s(t_x^y), a(t_x^y), c(t_x^y), s(t_x^{y+1})$ from the $\mathcal{D}$.
    **Evaluate** temporal difference based on Eq. (18).
    **Update** parameters of deep primary networks based on Eq. (19) and Eq. (22).
    **Update** parameters of deep target networks based on Eq. (23) and Eq. (24).
  **Until** this episode is terminated.
```

directly [33]. The updated parameters θ' and ϖ' are given by

$$\boldsymbol{\theta}' := \tau_{\boldsymbol{\theta}'}\boldsymbol{\theta}' + \left(1 - \tau_{\boldsymbol{\theta}'}\right)\boldsymbol{\theta} \tag{23}$$

and

$$\boldsymbol{\varpi}' := \tau_{\boldsymbol{\varpi}'}\boldsymbol{\varpi}' + (1 - \tau_{\boldsymbol{\varpi}'})\boldsymbol{\varpi} \tag{24}$$

where $\tau_{\boldsymbol{\theta}}' \in [0, 1]$ and $\tau_{\boldsymbol{\varpi}'} \in [0, 1]$ are weights with respect to $\boldsymbol{\theta}'$ and $\boldsymbol{\varpi}'$. Furthermore, $\boldsymbol{\theta}'$ and $\boldsymbol{\varpi}'$ can be regarded as approximately average values over $1/(1 - \tau_{\boldsymbol{\theta}}')$ and $1/(1 - \tau_{\boldsymbol{\varpi}}')$ time slots, respectively.

D. DDPG-Based Cooperative Caching Scheme

By combining the double time-scale content caching model and the actor-critic learning framework, the whole DDPG-based cooperative caching scheme is presented in Algorithm 1. The algorithm parameters include the total amount of content caching episodes X, content delivery deadline t_x^Y at caching episode t_x, discount factor, learning rates, replay buffer size B, and mini-batch size D. After initializing the network parameters and hyper-parameters, the decision of content placement is executed at the beginning of each episode t_x. Furthermore, the content delivery process is implemented by content fetching time slot under each current episode t_x. Then, the parameters of primary networks are updated by leveraging the Ornstein–Uhlenbeck-based policy exploration and mini-batch gradient descent technique. After a given training period, the parameters of the target network are updated based on the EWMA scheme. Last, the training process is terminated after having completed the maximal content caching period X.

VI. Performance Evaluation

In this section, we use Python to build a simulation environment for the vehicular edge caching system based on the real transportation data sets [34]. For an area of road network, we choose 16 transport regions and each of them can be divided as a square area. Furthermore, we use TensorFlow platform to implement the DDPG-based cooperative caching scheme. Our implementation is based on the open-source package DDPG [35]. In addition, the training performance of DDPG is characterized by intense oscillation in a certain range as the number of episodes increases. In order to adapt to the dynamic environment of IoV, the Monte Carlo experiment method is used to calculate the cumulative average caching performance [36]. The main parameters employed in the simulations are summarized in Table II.

TABLE II
Simulation Parameters

System Parameter	Value/Description
Number of MBS	1
Number of RSUs,	5
Number of CCVs,	[10, 30]
Number of CRVs,	[30, 60]
Bandwidth of MBS, RSUs, CCVs	[5MHz, 20MHz]
MBS transmission power	35dBm
RSU transmission power	33dBm
CCV transmission power	30dBm
Number of contents (F)	10
Size of contents	[200, 300]MBytes
Required content fetching deadline	[0, 300]Seconds

For performance comparison, we present two benchmark schemes: 1) the random caching scheme and 2) the noncooperative caching scheme.

1) *Random Caching Scheme:* The decision of content placement/updating process is randomly executed based on the DDPG learning algorithm.
2) *Noncooperative Caching Scheme:* The joint content placement and delivery can only be implemented at the RSUs without vehicular caching cooperation.

A. Data Analysis of Destination Prediction

We conducted a heuristic analysis of a set of vehicle driving data sets, which are composed of eight features, including the index of sample, the index of vehicle, start time, end time, latitude of origin, longitude of origin, latitude of destination, and longitude of destination. By utilizing the machine learning model, the destination of the vehicle can be predicted given the index of vehicle, latitude, and longitude of origin. Compared with the method of using longer strings to represent the transport region of smart vehicles, using shorter strings can significantly improve the accuracy of vehicle destination prediction. This result also shows that the length of the Geohash string can be flexibly adjusted to meet the tradeoff between destination prediction and accurate location of smart vehicles.

B. Numerical Analysis of Caching Performance

Fig. 5 shows the comparison of the cumulative system cost per episode of different caching schemes based on the DDPG learning algorithm, for a caching system with $CRVs = 30$ and $CCVs = 20$. All the three content caching schemes can approach their stable cumulative average cost as the number of episodes increases. We can draw the following observations from Fig. 5. First, the noncooperative caching scheme has the highest average system cost including content storage cost and access cost. The reason is that the CRVs need to fetch more

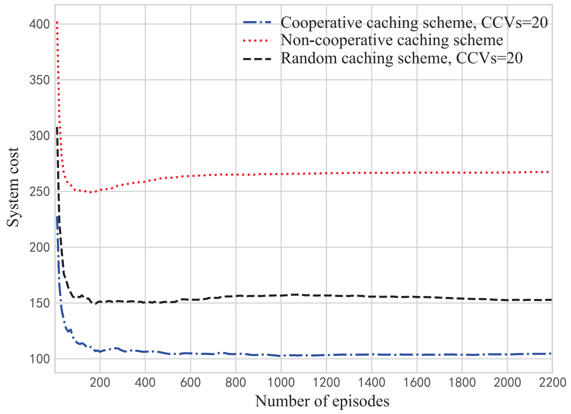

Fig. 5. Cumulative average system cost per episode achieved by different edge caching schemes.

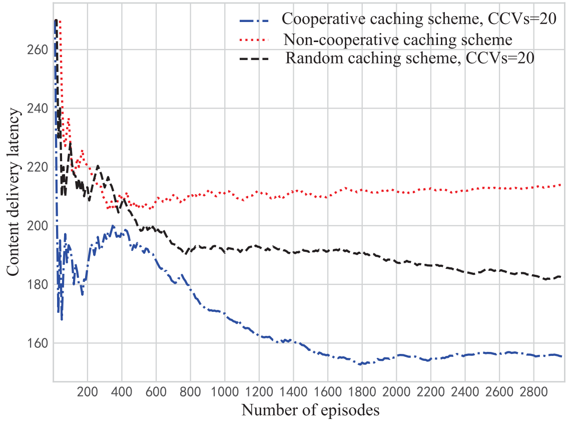

Fig. 6. Cumulative average delivery latency per episode achieved by different caching schemes.

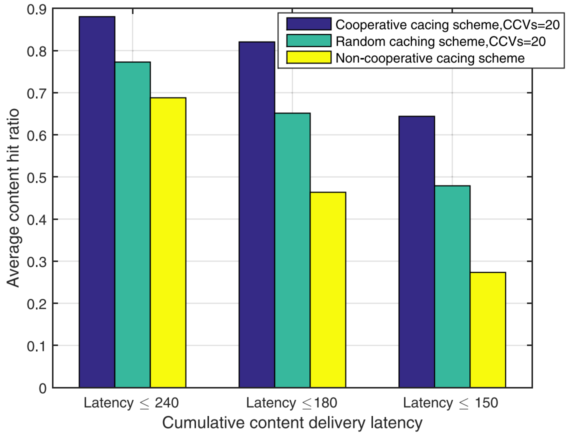

Fig. 7. Average content hit ratio per episode achieved by different content delivery latency.

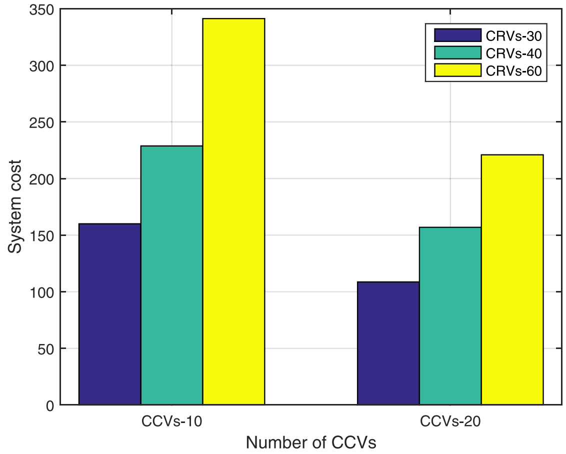

Fig. 8. Cumulative average system cost per episode achieved by different number of CRVs and CCVs.

segments of the whole content from centralized caching node within the individual content delivery deadline. Second, the V2V-assisted caching scheme can reduce system cost since CCVs can provide more mobile store capacity. In addition, the system cost can be further reduced when more CCVs participate in the edge caching scenarios. Third, our proposed cooperative caching scheme can yield the lowest system cost as compared to the other benchmark schemes. Thus, this result also demonstrates the efficiency of joint optimization of content placement and content delivery from the reduction of system cost.

On the other hand, the number of iterations of different caching schemes is basically same. It can be explained that the three caching schemes use the same deep learning framework which can largely determine the convergence performance of edge caching schemes.

Fig. 6 provides the comparison of cumulative average delivery latency of CRVs for the final content acquisition. In order to reduce the resource usage cost, CRVs should seek a cheap access mode to obtain segments of the whole content if there is enough time left to meet the content fetching deadline. For the noncooperative caching scheme, CRVs have to seek segments of the whole content from the centralized cache node within maximal fetching deadline. In contrast, the random caching scheme can reduce the content delivery latency since the CCVs can provide abundant of mobile bandwidth and storage resources. Furthermore, our proposed scheme results in the lowest content access latency due to the joint optimization of cooperative content placement/updating and opportunistic content delivery process.

In Fig. 7, we compare the cumulative average content hit ratio by different edge caching scheme. The cooperative caching scheme increases the content hit ratio by the (13.95%, 28.17%), (25.97%, 77.01%), and (34.44%, 135.62%), compared with random caching scheme and noncooperative scheme, respectively. This result shows that the V2V-assisted caching scheme can make full use the unutilized communication and storage resources of CCVs. Moreover, the result demonstrates that the proposed caching scheme can significantly improve the content hit ratio, especially in the low content delivery latency.

In Fig. 8, we compare the cumulative average system cost achieved by different numbers of CRVs and CCVs. First, when the number of CRVs is 30, the difference in system cost is not significant when 10 CCVs and 20 CCVs participate in the cooperative edge caching network. This explains that CRVs obtain the most segments of content through V2V-assisted caching while avoiding frequent content placement/updating operation. Second, more CCVs participating in the cooperative edge caching scenario can significantly reduce the system cost when massive CRVs need to access the contents. Moreover, the gap of the cumulative average system cost can be further reduced when the number of CCVs is 20 because more CCVs can provide a higher gain of content placement and opportunistic content fetching. Our proposed content caching scheme is beneficial for vehicular edge computing and networks especially when the vehicle density is high on the road.

VII. Conclusion

In this article, we have modeled the joint content placement and content delivery with DTS-MDP in vehicular edge computing and networks. In order to solve the LT-MILP problem, a DDPG-based caching scheme has been proposed to overcome with high-dimensional state space and continuous-valued action space. As compared with other benchmark schemes, numerical results have demonstrated the reduction of system cost and content delivery while enhancing the content hit ratio.

The proposed caching scheme is still under active improvement. To match the dynamic IoV environments, our proposed caching system should require the lifelong learning capability. We will continue to design a more effective and robust learning model. Moreover, the caching scheme can be further improved from the perspective of the cooperative optimization between road traffic control and resource management.

References

[1] M. Amadeo, C. Campolo, and A. Molinaro, “Priority-based content delivery in the Internet of Vehicles through named data networking,” *J. Sensor Actuator Netw.*, vol. 5, no. 4, pp. 1–17, 2016.

[2] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, and Y. Zhang, “Deep learning empowered task offloading for mobile edge computing in urban informatics,” *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7635–7647, Oct. 2019, doi: 10.1109/JIOT.2019.2903191.

[3] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, “Cooperative content caching in 5G networks with mobile edge computing,” *IEEE Wireless Commun. Mag.*, vol. 25, no. 3, pp. 80–87, Jun. 2018.

[4] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, “A survey on low latency towards 5G: RAN, core network and caching solutions,” *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3098–3130, 4th Quart., 2018.

[5] M. Chen, Y. Hao, M. Qiu, J. Song, D. Wu, and I. Humar, “Mobility-aware caching and computation offloading in 5G ultra-dense cellular networks,” *Sensors*, vol. 16, no. 7, p. 974, Jul. 2016.

[6] M. Gregori, J. Gómez-Vilardebó, J. Matamoros, and D. Gündüz, “Wireless content caching for small cell and D2D networks,” *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1222–1234, May 2016.

[7] Z. Chen and M. Kountouris, “D2D caching vs. small cell caching: Where to cache content in a wireless network?” in *Proc. IEEE 17th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, 2016, pp. 1–6.

[8] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, “Five disruptive technology directions for 5G,” *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 74–80, Feb. 2014.

[9] X. Wang, S. Leng, and K. Yang, “Social-aware edge caching in fog radio access networks,” *IEEE Access*, vol. 5, pp. 8492–8501, 2017.

[10] 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 Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5435–5444, Jun. 2018.

[11] G. Kleinberg and E. Tardos, *Algorithm Design*, Cornell Univ., Ithaca, NY, USA, 2004.

[12] N. Ansari and E. S. H. Hou, *Computational Intelligence for Optimization*. New York, NY, USA: Springer, 1997.

[13] W. Jiang, G. Feng, and S. Qin, “Optimal cooperative content caching and delivery policy for heterogeneous cellular networks,” *IEEE Trans. Mobile Comput.*, vol. 16, no. 5, pp. 1382–1393, May 2017.

[14] Y. Sun, Z. Chen, and H. Liu, “Delay analysis and optimization in cache-enabled multi-cell cooperative networks,” in *Proc. IEEE Glob. Commun. Conf.*, Dec. 2016, pp. 1–7.

[15] X. Zhang and Q. Zhu, “P2P caching schemes for jointly minimizing memory cost and transmission delay over information-centric networks,” in *Proc. IEEE Glob. Commun. Conf.*, Dec. 2016, pp. 1–6.

[16] J. Yao and N. Ansari, “Joint content placement and storage allocation in C-RANs for IoT sensing service,” *IEEE Internet Things J.*, vol. 6, no. 1, pp. 1060–1067, Feb. 2019.

[17] W. Zhao, Y. Qin, D. Gao, C. H. Foh, and H.-C. Chao, “An efficient cache strategy in information centric networking vehicle-to-vehicle scenario,” *IEEE Access*, vol. 5, pp. 12657–12667, 2017.

[18] S. Abdelhamid, H. S. Hassanein, and G. Takahara, “On-road caching assistance for ubiquitous vehicle-based information service,” *IEEE Trans. Veh. Technol.*, vol. 64, no. 12, pp. 5477–5492, Dec. 2015.

[19] B. Hu, L. Fang, X. Cheng, and L. Yang, “Vehicle-to-vehicle distributed storage in vehicular networks,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jul. 2018, pp. 1–6.

[20] Z. Zhang *et al.*, “Cache-enabled dynamic rate allocation via deep self-transfer reinforcement learning,” *arXiv: 1893.11334v1*, Mar. 2018.

[21] Z. Li, T. Chu, I. V. Kolmanovsky, X. Yin, and X. Yin, “Cloud resource allocation for cloud-based automotive applications,” *Mechatronics*, vol. 50, pp. 356–365, Apr. 2018.

[22] A. Sadeghi, F. Sheikholeslam, and G. B. Giannakis, “Optimal and scalable caching for 5G using reinforcement learning of space-time popularities,” *IEEE J. Sel. Areas Commun.*, vol. 12, no. 1, pp. 180–190, Feb. 2018.

[23] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, “Web caching and Zipf-like distribution: Evidence and implications,” in *Proc. IEEE Comput. Commun. Soc. (INFOCOM)*, New York, NY, USA, Mar. 2014, pp. 126–134.

[24] (2018). *Geohash Algorithm*. [Online]. Available: http://geohash.org

[25] A. T. Gamage, H. Liang, and X. Shen, “Two time-scale cross-layer scheduling for cellular/WLAN Interworking,” *IEEE Trans. Commun.*, vol. 62, no. 8, pp. 2773–2789, Aug. 2014.

[26] Q. Wang, S. Leng, H. Fu, and Y. Zhang, “IEEE 802.11p-based multichannel MAC scheme with channel coordination for vehicular ad hoc networks,” *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 449–458, Jun. 2012.

[27] C. Shao, S. Leng, Y. Zhang, A. Vinel, and M. Jonsson, “Performance analysis of connectivity probability and connectivity-aware MAC protocol design for platoon-based VANETs,” *IEEE Trans. Veh. Technol.*, vol. 64, no. 2, pp. 5596–5609, Dec. 2015.

[28] C. Zhang, Z. Liu, B. Gu, K. Yamori, and Y. Tanaka, “A deep reinforcement learning based approach for cost- and energy-aware multi-flow mobile data offloading,” *IEICE Trans. Commun.*, vol. E101-B, no. 7, pp. 1625–1634, Jul. 2018.

[29] V. R. Konda and J. N. Tsitsiklis, “Actor–critic algorithm,” in *Proc. NIPS*, vol. 13, 1999, pp. 1008–1014.

[30] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstr, and M. Riedmiller, “Determinstic policy gradient algorithms,” in *Proc. 31st Int. Conf. Mach. Learn. (ICML)*, 2014, pp. 387–395.

[31] X. Huang, T. Yuan, G. Qiao, and Y. Ren, “Deep reinforcement learning for multimedia traffic control in software defined networking,” *IEEE Netw. Mag.*, vol. 32, no. 6, pp. 35–41, Nov./Dec. 2018.

[32] S. Csaba, “Algorithms for reinforcement learning,” in *Synthesis Lectures on Artificial Intelligence and Machine Learning*. San Rafael, CA, USA: Morgan & Claypool, 2010, pp. 1–103.

[33] T. Lillicrap *et al.*, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv: 1509.02971*, 2015.

[34] (2016). *DC Data Sets*. [Online]. Available: http://www.dcjingsai.com/common/cmptIndex.html

[35] (2016). *Reimplementation of DDPG(Continuous Control With Deep Reinforcement Learning) Based on OpenAI Gym + Tensorflow*. [Online]. Available: https://github.com/songrotek/DDPG

[36] N. Mastronarde and M. V. D. Schaar, “Fast reinforcement learning for energy-efficient wireless communication,” *IEEE Trans. Signal Process.*, vol. 50, no. 12, pp. 6262–6266, Aug. 2011.

Guanhua Qiao is currently pursuing the Ph.D. degree with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China.

His current research interests include mobile edge computation and caching in wireless heterogeneous network, resource management, and network optimization.

Supeng Leng received the Ph.D. degree from Nanyang Technological University, Singapore.

He is a Full Professor and a Vice Dean with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China. His current research interests include Internet of Things, vehicular networks, and next generation mobile networks.

Prof. Leng serves as the organizing committee chair and a TPC member for several international conferences, as well as a reviewer for over ten international research journals.

Sabita Maharjan (M'09) received the Ph.D. degree in networks and distributed systems from the Simula Research Laboratory, University of Oslo, Oslo, Norway, in 2013.

She is currently a Senior Research Scientist with the Simula Metropolitan Center for Digital Engineering, Oslo, and an Associate Professor with the University of Oslo. Her current research interests include wireless networks, network security and resilience, smart grid communications, Internet of Things, machine-to-machine communications, and Internet of Vehicles.

Yan Zhang (M'05–SM'10) received the Ph.D. degree from Nanyang Technological University, Singapore.

He is a Full Professor with the University of Oslo, Oslo, Norway. His current research interests include next generation wireless networks and cyber physical systems.

Prof. Zhang was a recipient of the Highly Cited Researcher Award (Web of Science top 1% most cited) by Clarivate Analytics. He is an Editor of several IEEE publications, including the *IEEE Communications Magazine*, IEEE NETWORK, and the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING. He is an IEEE VTS Distinguished Lecturer.

Nirwan Ansari (S'78–M'83–SM'94–F'09) received the Ph.D. degree from Purdue University, West Lafayette, IN, USA.

He is a Distinguished Professor of Electrical and Computer Engineering with the New Jersey Institute of Technology, Newark, NJ, USA. He has (co)authored three books, and over 600 technical publications and over 280 published in widely cited journals/magazines. His current research interests include green communications and networking, cloud computing, drone-assisted networking, and various aspects of broadband networks.

Prof. Ansari has guest edited a number of special issues covering various emerging topics in communications and networking. He has served on the editorial/advisory board of over ten journals including as an Associate Editor-in-Chief of the *IEEE Wireless Communications Magazine*.