Mobility-Aware Edge Caching and Computing in Vehicle Networks: A Deep Reinforcement Learning

Le Thanh Tan, *Member, IEEE*, and Rose Qingyang Hu, *Senior Member, IEEE*

***Abstract*—This paper studies the joint communication, caching and computing design problem for achieving the operational excellence and the cost efficiency of the vehicular networks. Moreover, the resource allocation policy is designed by considering the vehicle's mobility and the hard service deadline constraint. These critical challenges have often been either neglected or addressed inadequately in the existing work on the vehicular networks because of their high complexity. We develop a deep reinforcement learning with the multi-timescale framework to tackle these grand challenges in this paper. Furthermore, we propose the mobility-aware reward estimation for the large timescale model to mitigate the complexity due to the large action space. Numerical results are presented to illustrate the theoretical findings developed in the paper and to quantify the performance gains attained.**

***Index Terms*—Vehicular networks, mobility, edge caching, edge computing, deep reinforcement learning.**

I. Introduction

THE flawless multimedia applications supported by the next-generation networks demand an unprecedented capacity escalation and stringent quality of service (QoS) in the connected vehicular networks. The vehicular networks must employ advanced communications technologies and data collecting techniques to improve safety, enhance efficiency, reduce accidents and decrease traffic congestion in the transportation systems. To tackle these grant challenges, advanced technologies are required for enhancing the capacity and QoS [1]–[3]. Furthermore, recent studies reveal that different contents require different levels of priority [4], [5]. Explicitly, the most popular contents are requested by the majority of users, whilst the remaining large portion of the contents impose rather infrequent access demands [4]. Recently, edge caching placements and computing offloading at the vehicles and the road side units (RSUs) have been proposed to efficiently improve the QoS for applications with intensive computations and hard deadlines [6], [7]. In particular, edge computing is a term for an alternative to cloud computing where it moves the storage, communication, control, configuration, computation, measurement and management from the centralized cloud to the edge of the network [8]–[10]. Edge computing can alleviate the challenges and overcome the disadvantages of cloud computing with many constraints. We would like to design an edge computing architecture by considering supporting technologies, namely caching placement and D2D communications under the scenario of the user's mobility.

D2D communication in heterogeneous networks (HetNets) is gaining increasing attentions. It can cache popular contents at various nearby devices [11]. In general, the integration of D2D communications into cellular networks helps improve cell coverage, spectral efficiency as well as traffic congestion [12]–[15]. In the recent information-centric networking paradigm, cooperative caching constitutes one of the most widely studied wireless caching paradigms, where local user terminals and relay nodes can cooperatively store the multimedia contents [16]–[20], [23]–[26]. Hence, a user can download the requested contents directly from the caching-enabled nearby users or from the nearby relays instead of acquiring it from the far away base station (BS). By doing so, the performance quantified in terms of access delay, throughput and scalability of the wireless network would be significantly improved. Recently, exploiting user's mobility for caching placement strategies has also received much attention from researchers [27]–[29]. In the dense network, a user can receive a portion of the requested content during a short time duration. The authors in [27], [29] developed a framework of mobility-aware coded caching. The authors in [28] worked on minimizing the workload at macro BSs by making the strategy of content caching at the small-cell base stations when the user's mobility is considered.

Machine learning is an emerging tool to tackle problems encountered in caching, computing and communications in 5G wireless communications [21], [23], [26], [30]–[40]. Bharath *et al.* [30] considered the scenario of unknown time-varying popularity profile of cached contents, which were then estimated by using their proposed transfer learning-based approach. Although, this paper is one of the first few works introducing an analytical treatment of training time, their proposed scheme does not theoretically guarantee the convergence when there is an overlapping coverage between different BSs. Wang *et al.* [31] proposed a joint BS caching and D2D offloading using Q-learning. In fact, they applied Q-learning for the distributed cache replacement strategy according to the content popularity. Tanzil *et al.* [32] introduced a novel scheme for estimating the unknown popularity of caching contents, namely an extreme-learning machine framework. As a result, the proposed algorithms help improve the user's QoS as well as reduce the network traffic.

Manuscript received April 1, 2018; revised July 19, 2018 and July 31, 2018; accepted July 31, 2018. Date of publication August 27, 2018; date of current version November 12, 2018. This work was supported by the National Science Foundation under Grants NeTS-1423348 and EARS-1547312. The review of this paper was coordinated by Dr. Editors of CVS TVT. *(Corresponding author: Rose Qingyang Hu.)*

The authors are with the Department of Electrical and Computer Engineering, Utah State University, Logan, UT 84322-4120 USA (e-mail: tan.le@usu.edu; rose.hu@usu.edu).

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.2018.2867191

0018-9545 © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: China University of Petroleum. Downloaded on December 20,2020 at 19:16:57 UTC from IEEE Xplore. Restrictions apply.

In this paper, we make a further bold step in designing, analyzing and optimizing the cooperative coded caching placement and computing allocation at both the vehicle level and the RSU level by considering the constraints of vehicle's mobility and hard deadline delay. Specifically, the contributions of this paper can be summarized as follows.

1) We model the HetNets with mobility-aware coded probabilistic caching and computational offloading scheme, where both vehicles and RSUs have caching storage and computing capabilities. A cache hit occurs if the requested content is received from the nearby vehicles via D2D communications or from the nearby RSUs during the vehicular movement. In the case of a cache miss, the vehicle receives the requested content from the BS. Also the vehicle offloads the tasks to the RSUs and the nearby vehicles under the delay constraint. In the case of lacking computing resources from the vehicle's neighbors, the requesting vehicle must offload the tasks to the BS.
2) We formulate the joint optimal caching and computing allocation problem to minimize the system cost under the constraints of limited and dynamic storage capacities and computation resources at the vehicles and the RSUs as well as the constraints of the vehicle's mobility and the hard deadline delay.
3) We develop the algorithm based on deep Q-learning with multi-timescale framework [33]–[36] to configure the parameters of caching placement and computing resource allocation as well as to determine the sets of possible connecting RSUs and neighboring vehicles. To reduce the complexity due to the large action space, we propose the mobility-aware reward estimation for the large timescale model.
4) We present numerical results to illustrate the performance of the proposed algorithms by using the optimal parameter configuration for caching, computing and vehicular mobility. The impact of the vehicular mobility, data size, backhaul capacities and cloud resources on the system performance is studied.

The outline of this paper is as follows. Section II describes our system model. Section III briefly presents the reinforcement learning and deep Q-learning. In Section IV, we use deep reinforcement learning to formulate the resource allocation optimization problem. Then we describe the large timescale deep Q-learning in Section V and the large timescale deep Q-learning in Section VI. Section VII presents our performance results followed by the concluding remarks in Section VIII.

II. System Models

A. Network Architecture

We consider a vehicular network that includes one BS, K relays/RSUs integrating mobile edge computing (MEC) servers and U vehicles. Note that MEC servers are installed at the RSUs for computing, which help reduce the workload at the BS. Furthermore, we investigate the case that vehicles have both computing and caching capabilities. Let $\mathcal{K} = \{1, \ldots, K\}$ and $\mathcal{U} = \{1, \ldots, U\}$ be the sets of the RSUs (or the MEC servers) and the vehicles, respectively. We assume that the requesting vehicle can concurrently download the requested content and upload its tasks to the contact RSUs/vehicles/BS by employing the full-duplex radio [40], [41]. Let us look at an example in the mobility-aware caching and computational model in Fig. 1. The red vehicle first moves from RSU 1 to RSU 2. So the data offloading and the computation offloading also change from RSU 1 to RSU 2 during its movement. The vehicle further moves away from RSU 2 and enters yellow vehicle's communication range. It then performs the data offloading and the computation offloading to the yellow vehicle. Finally, it moves out of the communication ranges of two RSUs and other vehicles. Therefore, it connects directly to the BS for the data offloading and the computation offloading.

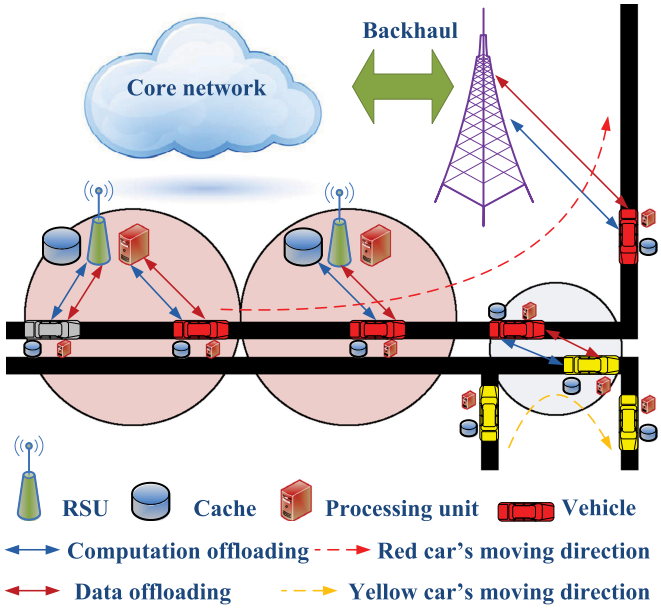


Fig. 1. Mobility-aware caching and computational model.

B. Communication Model

We assume that the channel parameters of vehicle-to-vehicle links and vehicle-to-RSU links are time-varying and are modeled as the finite-state Markov chains (FSMC) [42]. Let γ_i^k (and γ_i^u) denote the received SNR of the link between vehicle i and RSU k (and the link between vehicle i and vehicle u). The link between vehicle i and RSU k is modeled as an FSMC and the link from vehicle i to vehicle u is modeled in the same manner. We partition and quantize γ_i^k into L discrete levels, each of which represents a state of the Markov chain. Let T_d be the duration of communication for the content c, which is called an epoch. Each epoch can be divided into T_d time slots. So the realization of γ_i^k at instant t is $\Gamma_i^k(t)$. The channel state transition probability matrix for the link of vehicle i-RSU k is written as

$$\Psi_i^k(t) = [\psi_{g_s h_s}(t)]_{L \times L}, \tag{1}$$

where $\psi_{g_s h_s}(t) = \Pr\left(\Gamma_i^k(t+1) = h_s \left| \Gamma_i^k(t) = g_s \right.\right)$. Here, h_s and g_s represent two states.

We assign the OFMD based orthogonal bandwidth of $b_{i,k}$ (and $b_{i,u}$) to the link between vehicle i and RSU k (and the link between vehicle i and vehicle u). Hence, there is no interference from one link to the other. Let $\nu_{i,k}(t)$ (and $\nu_{i,u}(t)$) denote the spectral efficiency of the link between vehicle i and RSU k (and

Authorized licensed use limited to: China University of Petroleum. Downloaded on December 20,2020 at 19:16:57 UTC from IEEE Xplore. Restrictions apply.

the link between vehicle i to vehicle u). The communication rate of vehicle i can be expressed as

$$R_{i,k}(t) = b_{i,k}(t)v_{i,k}(t), \forall i \in \mathcal{U},$$
$$R_{i,u}(t) = b_{i,u}(t)v_{i,u}(t), \forall i \in \mathcal{U}. \quad (2)$$

C. Computing Model

Each task W_i^c consists of two components, i.e. $\{l_w^c, D_w^c\}$. Here, l_w^c is the size of the computation task for the content c, which includes the program code and the input parameters. Moreover, D_w^c is the number of CPU cycles required to accomplish the task. After accomplishing the computing, RSU k or vehicle u having computing capability sends the computing results (control signals) back to vehicle i. Note that we ignore the transmission time for the control signals due to the small amount of data.

Let $f_{i,k}$ (and/or $f_{i,u}$) (CPU cycles per second) be the computation capability of RSU k (and/or the computation capability of vehicle u) allocated to vehicle i. Multiple vehicles may access the same RSU and share the same MEC server at a given time. Hence, we do not know exactly the computation capability for vehicle i at the next time instant. Thus, the computation capability $f_{i,k}$ can be modeled as a random variable. We divide $f_{i,k}$ into N levels, where N corresponds to the number of available computation capability states. So the realization of $f_{i,k}$ is $F_{i,k}(t)$ at time slot t and the transition probability matrix of RSU k reserved for vehicle i is

$$\Theta_{k,i}(t) = [\theta_{x_s y_s}(t)]_{N\times N}, \quad (3)$$

where $\theta_{x_s y_s}(t) = \Pr(F_{i,k}(t+1) = y_s \,|F_{i,k}(t) = x_s)$. Here, x_s and y_s are the states. The same analysis on $f_{i,k}$ can be applied to $f_{i,u}$. So the computing rate (bits computed per second) is expressed as

$$R_{i,k}^{w,c}(t) = \frac{f_{i,k}(t)l_w^c}{D_w^c},$$
$$R_{i,u}^{w,c}(t) = \frac{f_{i,u}(t)l_w^c}{D_w^c}. \quad (4)$$

D. Caching Model

We consider the scenario, in which C contents are requested by vehicles. The average request rate for content c ($c \in [1, 2, \ldots, C]$) at time t can be denoted as

$$\lambda_c(t) = \frac{\beta}{\rho c^\alpha}. \quad (5)$$

Note that the set $\{1, 2, \ldots, C\}$ is given in a descending order, i.e. the index c stands for the c-th most popular content. We assume that the requests follow a Poisson process with parameter β. The requesting probability can be calculated by a Zipf function, i.e. $1/\rho c^\alpha$, where $\rho = \sum_{c=1}^{C} 1/c^\alpha$ and α $(0 < \alpha < 1)$ is the popularity skew.

Here, the content is periodically cached at the caching storage. Let ς_c denote the availability of content c. Furthermore, $\varsigma_c = 1$ if content c is available and $\varsigma_c = 0$ otherwise. So the transition probability matrix of content c is expressed as

$$\Gamma = [\delta_{v_s \omega_s}(t)]_{2\times 2}, \quad (6)$$

where $\delta_{v_s \omega_s}(t) = \Pr(\varsigma_i(t+1) = \omega_s \,|\varsigma_i(t) = v_s)$. Moreover, v_s and ω_s are the states. In the finite cache capacity, we employ the least recently used (LRU) cache replacement policy [43] to model ς_c. Note that the lifetime of content c follows an exponential distribution with mean $1/\mu_c$. So the transition probability matrix can be written as [43]

$$\Lambda_c = \begin{bmatrix} -\lambda_c & 0 & \ldots & 0 & 0 & \lambda_c \\ \kappa_c + \mu_c & -\beta - \mu_c & \ldots & 0 & 0 & \lambda_c \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mu_c & 0 & \vdots & \kappa_c & -\beta - \mu_c & \lambda_c \\ \mu_c & 0 & \vdots & 0 & \kappa_c & -\kappa_c - \mu_c \end{bmatrix}, \quad (7)$$

where $\kappa_c = \beta - \lambda_c$.

E. Mobility Model and Coded Caching Scheme

In this paper, we adopt the contact time (or the sojourn time) T_{con} and the connect frequency λ to model the mobility pattern of vehicles. Although vehicle positions can change any time, the moving range of a vehicle is relatively small during a short time. We assume that within contact time T_{con}^U, a vehicle keeps connected with the same vehicle and within contact time T_{con}^R, a vehicle keeps connected with the same RSU. In particular, we assume $T_{con}^U < T_{con}^R$, since the two vehicles that are connected via D2D can move at the same time while RSUs normally remain static. If vehicles move at a high speed, the contact time can be short, compared with a rather longer contact time if vehicles move at a low speed.

Vehicle's mobility is modeled by discrete random jumps with the corresponding intensity characterized by the average sojourn time between jumps. The numbers of contacts between vehicle i and vehicle u as well as between vehicle i and RSU k both follow the Poisson distribution with parameters of $\chi_{i,u}$ and $\lambda_{i,k}$, respectively. Here $\lambda_{i,k}$ and $\chi_{i,u}$ are the connect frequencies, which account for mobility intensities.

In a coded caching scheme, each content c is encoded into multiple segments with length l_c by the rateless Fountain code. Therefore, a requested content can be recovered by collecting any s_c encoded segments, which are a subset of the complete segments. These segments can be cached in the local vehicle storage or in the RSU storage. If a vehicle can not collect enough encoded segments from the local cache via D2D communications or RSU transmission within the tolerate time T_d, the cellular network sends the missing data to the vehicle. Hence, coded caching scheme particularly helps the vehicles in motion.

III. Deep Reinforcement Learning

This section briefly presents the reinforcement learning and deep Q-learning. More details can be found in [33]–[36].

A. Overview of Reinforcement Learning

Reinforcement learning aims to find a balance between exploration (of uncharted territory) and exploitation [33]. In reinforcement learning, the environment is typically formulated as a

Authorized licensed use limited to: China University of Petroleum. Downloaded on December 20,2020 at 19:16:57 UTC from IEEE Xplore. Restrictions apply.

Markov decision process (MDP). However, the state space, the explicit transition probability, and the reward function are not necessarily required [33], [34]. The agent interacts with an unknown environment through the repeated observations, actions and rewards to construct the optimal strategy. It is a promising approach to deal with tasks with high complexity in the real world [34]. There are two kinds of reinforcement learning, namely model-free and model-based reinforcement learning, depending on whether the transition probability is given or not. Moreover, a variety of model-free and model-based algorithms [35] can be used to approximate the reward functions. We now briefly describe the two mechanisms for reinforcement learning.

B. Model-Based Reinforcement Learning

Let $X = \{x_1, x_2, \ldots, x_n\}$ denote the state space and let $A = \{a_1, a_2, \ldots, a_m\}$ denote the action set. The agent takes an action $a(t)$ ($a(t) \in A$) based on the current state $x(t)$ ($x(t) \in X$). Then the system transfers to a new state $x(t+1)$ ($x(t+1) \in X$) with the transition probability $P_{x(t)x(t+1)}(a)$ and obtains the immediate reward $r(x(t), a(t))$.

For the long-term consideration, the target is the future reward that is characterized by a discount factor ϵ ($0 < \epsilon < 1$). The reinforcement learning agent aims to determine an optimal policy a^* ($a^* = \pi^*(x) \in A$) for each state x, which maximizes the expected time-average reward. This quantity is expressed as

$$V^{\pi}(x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \epsilon^t r(x(t), a(t)) \,|x(0) = x\right], \tag{8}$$

where $\mathbb{E}$ denotes the expectation.

Recall that environment is formulated as an MDP. Hence, the value function can be rewritten as

$$V^{\pi}(x) = R(x, \pi(x)) + \epsilon \sum_{x' \in X} P_{xx'}(\pi(x)) V^{\pi}(x'), \tag{9}$$

where $R(x, \pi(x))$ is the mean value of the immediate reward $r(x, \pi(x))$ and $P_{xx'}(\pi(x))$ is the transition probability from x to x', when the action $\pi(x)$ is executed. So the optimal policy π^* follows Bellman's criterion as

$$V^{\pi^*}(x) = \max_{a \in A}\left[R(x, a) + \epsilon \sum_{x' \in X} P_{xx'}(a) V^{\pi^*}(x')\right]. \tag{10}$$

In the model-based learning mechanism, the reward R and the transition probability P are given. Thus, the optimal policy can be derived. The model-free learning algorithm is based on unknown R and P. This mechanism will be presented in the following.

C. Model-Free Reinforcement Learning

We now consider the model-free reinforcement learning, where both R and P are unknown. Note that Q-learning is one of the strategies to determine the best policy π^*. A state-action function, namely Q-function, is defined as

$$Q^{\pi}(x, a) = R(x, a) + \epsilon \sum_{x' \in X} P_{xx'}(a) V^{\pi}(x'), \tag{11}$$

which represents the discounted cumulative reward, when action a is performed at state x and continues optimal policy from that point on. The maximum Q-function is expressed as

$$Q^{\pi^*}(x, a) = R(x, a) + \epsilon \sum_{x' \in X} P_{xx'}(a) V^{\pi^*}(x'). \tag{12}$$

The discounted cumulative state function can be written as

$$V^{\pi^*}(x) = \max_{a \in A} Q^{\pi^*}(x, a). \tag{13}$$

We now aim to estimate the best Q-function instead of finding the best policy. The Q-function can be obtained by using the recursive method [35].

$$Q_{t+1}(x, a) = Q_t(x, a) + \alpha\left(r + \epsilon \max_{a'} Q_t(x', a') - Q_t(x, a)\right), \tag{14}$$

where α is the learning rate. Furthermore, $Q_t(x, a)$ definitely converges to $Q^*(x, a)$, when a proper α is selected [36]. One way to estimate the Q-function is to use a function approximation such as the neural network $Q(x, a; \theta) \approx Q^*(x, a)$, where the parameter θ is the weight of the neural network [34]. In particular, θ is adjusted at each iteration during the Q network training in order to reduce the mean square error. Note that the neural network acts as a function approximation. Hence, we can represent any number of possible states as a vector. Then the system learns to map these states to the Q-values. Although the combination of the neural networks and the Q-learning makes the reinforcement learning algorithms more flexible and efficient, it still exhibits some instabilities. Therefore, deep Q-learning is introduced to deal with this issue as well as to enhance the reinforcement learning algorithms. This novel mechanism will be presented the next section.

D. Deep Q-Learning

Deep Q-learning is developed to make the reinforcement learning applicable to the real applications. It requires two improvements to transform the regular Q-learning to deep Q-learning. The first one is an experience replay. At each time instant t, an agent stores its interaction experience tuple, $e(t) = (x(t), a(t), r(t), x(t+1))$, into a replay memory, $D(t) = \{e(1), \ldots, e(t)\}$. Different from the traditional Q-learning, where the arrived samples are used to train neural network's parameters, deep Q-learning randomly selects the samples experience pool to train the deep neural network's parameters. The second modification is that deep Q-learning updates the weight every N time steps, instead of updating it every time step. By doing so, the learning process becomes more stable.

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

$$Loss(\theta) = \mathbb{E}\left[\left(y - Q(x, a, \theta)^2\right)\right], \tag{15}$$

where the target value y is expressed as $y = r + \max_{a'} Q(x', a', \theta_i^-)$. In the Q-learning, the weights $\theta_i^- = \theta_{i-1}$, whereas in deep Q-learning $\theta_i^- = \theta_{i-N}$.

Authorized licensed use limited to: China University of Petroleum. Downloaded on December 20,2020 at 19:16:57 UTC from IEEE Xplore. Restrictions apply.

IV. Mobility-Aware Edge Caching and Computing Framework using Deep Q-learning in Vehicle Networks

We now use deep reinforcement learning to formulate the resource allocation optimization problem, where the parameters of caching, computing and communication are optimized jointly. Recall that the network has K RSUs/MEC servers, U vehicles and C contents. The downlink channel conditions, the computation capabilities, the caching states and vehicular mobility intensity all change dynamically. We aim to determine the subset of the RSUs and the nearby vehicles as well as their resources to serve the requesting vehicle based on the current states. Due to the dynamic variants as well as the large space of system states and actions, it is very hard to solve this optimization problem by employing the traditional methods. To deal with this issue, we exploit the recent advanced deep Q-learning, which can effectively and efficiently determine the optimal action according to the large amount of input data.

The operation for control system is as follows. At first, the control system collects the status from each RSU/MEC server, each vehicle and each vehicle's mobility. Then it constructs the system states, which are the vehicle's mobility and communication channel information as well as the caching contents and the computing resources at the RSU/MEC servers and the vehicles. The deep Q-network (or agent) receives the system states and determines the optimal action a^*. This action includes the sets of the RSUs and the vehicles as well as their caching and computing resources for the requesting vehicle. Finally, the control system receives this action and forwards it to the vehicle.

In the following, we give the brief description on the deep Q-network. Interested readers can find detailed information in [34]. At the deep Q-network, the replay memory stores the agent's experience at each time slot. The Q-network parameter, θ, is updated at every time instant with samples from the replay memory. The target Q-network parameter $\bar{\theta}$ is copied from the Q-network every N time instants. The ϵ-greedy policy is utilized to balance the exploration and exploitation. It implies that this policy balances the reward maximization based on the knowledge already acquired with attempting new actions to further increase knowledge [34]. The training algorithm of the deep Q-network is described in Algorithm 1.

Algorithm 1: Mobility-Aware Edge Caching and Computing via Deep Reinforcement Learning.

Initialize the experience replay buffer.
Initialize the main deep Q-network with weights θ.
Initialize the target deep Q-network with weights $\bar{\theta} = \theta$.
for each episode $j = 1, 2, \ldots, J$ **do**
  Receive the initial observation s_1, and pre-process s_1 to be the beginning state x_1.
  for $t = 1, 2, \ldots, T$ **do**
    Choose a random probability p.
    Choose a_t as,
    if $p \leq \epsilon$ **then**
      randomly select an action a_t,
    else
      $a_t = \arg\max_a Q(x, a; \theta)$.
    end if
    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 M samples (x_i, a_i, r_i, x_{i+1}) from the replay memory.
    Calculate the target Q-value $\bar{y}_t$ from the target deep Q-network, $\bar{y}_i = r_i + \epsilon Q(x_{i+1}, \arg\max_{a'} Q(x_{i+1}, a'; \theta); \bar{\theta})$.
    Update the main deep Q-network by minimizing the loss $Loss(\theta)$, $Loss(\theta) = 1/M \sum_i (\bar{y}_i - Q(x_i, a_i; \theta))^2$, and perform a gradient descent step on $Loss(\theta)$ with respect to θ.
    Update the target deep Q-network parameters with rate ψ, $\bar{\theta} = \psi\theta + (1 - \psi)\bar{\theta}$.
  end for
end for

Based on [46]–[48], we propose a multi-timescale system framework, where the large timescale Q-learning model is for every epoch (corresponding to T_d time slots) and the small timescale Q-learning model is for every time slot. Similar to [47], [48], the goal for the large timescale model is to receive the requested content and to compute its corresponding tasks within one epoch, while the optional goal is to perform the action based on the raw states. To do so, the system selects the appropriate sets of the possible RSUs/vehicles for caching as well as the sets of the MEC servers/vehicles for computing. Then the small timescale model selects the actions based on the states, where these selected action sets aim to maximize the immediate reward. However, the proposed algorithms in [47], [48] are only efficient for the small size networks.

To deal with the large networks with a massive number of nodes, we need to develop other algorithms, which allow for more robust and efficient learning. Therefore, the reward achieved can be estimated, when we take an action in the large timescale model, while in the small timescale model, we calculate the exact immediate reward achieved, when we take the action. This is because the resource allocation can be complicated in a mobility scenario. The action space is very large and causes the high complexity in the implementation. To realize a low-complex algorithm, we propose the mobility-aware reward estimation for the large timescale model. In order to obtain the optimal policy, we use Algorithm 1 for learning in both the large timescale and small timescale models. Moreover, we need to define the states, to perform the actions and to calculate the reward functions. All of these will be presented in the following sections.

V. The Large Timescale Deep Q-Learning Model

In this section, we describe the state, action, and reward for the large timescale deep Q-learning model. The best set of caching RSUs and/or vehicles are selected to serve the requesting vehicle and the best set of MEC severs and/or vehicles are selected to compute the offloading tasks from the requesting vehicle. Recall that the requested content needs to be received within T_d, i.e., the computation and content downloading must be done before the deadline of T_d. Furthermore, we use the mobility

Authorized licensed use limited to: China University of Petroleum. Downloaded on December 20,2020 at 19:16:57 UTC from IEEE Xplore. Restrictions apply.

model to estimate the reward for every epoch to reduce the high complexity due to the large space of actions.

A. System State

The states of the available RSU/MEC servers $k \in \{1, 2, \ldots, K\}$, the available vehicles $u \in \{1, 2, \ldots, U\}$ and the available caches $c \in \{1, 2, \ldots, C\}$ for vehicle i in time slot t (with duration T_d), are all determined by the realization of the states of the random variables γ_i^k (γ_i^u), the realization of the states of the random variables $f_{i,k}$ ($f_{i,u}$) and the realization of the states of the random variables ς_c. Furthermore, the input data also include the number of contacts per time slot (contact rate) and the contact times for every vehicle i-vehicle u and vehicle i-RSU k contact.

B. System Action

In the system, the agent has to decide which RSU and/or vehicle is assigned to the requesting vehicle, whether or not the requested content should be cached in the RSU and/or vehicle, how many coded packets should be cached and whether or not the computation task should be offloaded to the RSU/MEC server and/or the vehicle.

The current composite action, $a_i(t)$, is denoted by

$$a_i(t) = \left\{a_i^{\text{cp}}(t), a_i^{\text{ca}}(t), a_i^{\text{cd}}(t)\right\}, \tag{16}$$

where $a_i^{\text{cp}}(t)$, $a_i^{\text{ca}}(t)$ and $a_i^{\text{cd}}(t)$ are defined and interpreted as follows:

- For the caching control of RSUs, define $K \times C$ matrix $a_i^{\text{ca}}(t) = a_{K\times C}^{\text{ca}}$, where each element $a_{k,c}^{\text{ca}}(t)$ represents the cache control of the c-th content cached at RSU k for vehicle i and $a_{k,c}^{\text{ca}}(t) \in \{0, 1\}$. Here, $a_{k,c}^{\text{ca}}(t) = 0$ means that the c-th content is not cached at RSU k at t or RSU k is out of the communication range of vehicle i during time slot t, while $a_{k,c}^{\text{ca}}(t) = 1$ means that the c-th content is cached and RSU k is a contact candidate with vehicle i at t. It implies that at every epoch, $a_{K\times C}^{\text{ca}}$ determines the subset of RSUs that are the contact candidates with vehicle i as well as the subset of contents that are cached at these RSUs. Define matrix $a_i^{\text{cd}}(t) = a_{K\times C}^{\text{cd}}$, where each element $a_{k,c}^{\text{cd}}(t)$ represents the number of coded packets of the c-th content cached at RSU k and $a_{k,c}^{\text{cd}}(t) \in \{0, s_c\}$. Recall that s_c is the minimum number of coded packets, which is required to decode the content c successfully.
  Similarly, for the caching control of neighboring vehicles (if they have the caching storage), define $U \times C$ matrix $a_i^{\text{ca}}(t) = a_{U\times C}^{\text{ca}}$, where each element $a_{u,c}^{\text{ca}}(t)$ represents the cache control of the c-th content cache at vehicle u for vehicle i and $a_{u,c}^{\text{ca}}(t) \in \{0, 1\}$. Again, $a_{U\times C}^{\text{ca}}$ determines the subset of vehicles, which are the contact candidates with vehicle i as well as the subset of contents that are cached at these vehicles. Define matrix $a_i^{\text{cd}}(t) = a_{U\times C}^{\text{cd}}$, where each element $a_{u,c}^{\text{cd}}(t)$ represents the number of coded packets of the c-th content cached at vehicle u and $a_{u,c}^{\text{cd}}(t) \in [0, s_c]$.
- For the computing control of RSUs (MEC servers), define $K \times C$ matrix $a_i^{\text{cp}}(t) = a_{K\times C}^{\text{cp}}$, where each element $a_{k,c}^{\text{cp}}(t)$ represents the offloading control of the k-th MEC server for vehicle i and $a_{k,c}^{\text{cp}}(t) \in \{0, 1\}$. Here, $a_{k,c}^{\text{cp}}(t) = 0$ means that the task is not offloaded to MEC server k at time t or RSU k is out of the communication range of vehicle i during time slot t, while $a_{i,k}^{\text{cp}}(t) = 1$ means that it is offloaded and RSU k is a contact candidate with vehicle i at time t. Similarly, for the computing control of neighboring vehicles (if they have computing capability), define $U \times C$ matrix $a_i^{\text{cp}}(t) = a_{U\times C}^{\text{cp}}$, where each element $a_{u,c}^{\text{cp}}(t)$ represents the computing control of the tasks of c-th content at vehicle u for vehicle i and $a_{u,c}^{\text{ca}}(t) \in \{0, 1\}$.

C. Reward Function

To maximize the reward, we aim to minimize the cost of communication, storage and computation. Firstly, δ^R is defined as the cost of communication for data transmission between vehicles and RSUs (or between vehicles and vehicles), while δ^{BS} is the cost of communications for data transmission between vehicles and the BS. Then ξ_c is the cost of caching storage. Finally, η^R and η^{BS} are the cost of computation of MEC servers and BS, respectively. We define the reward for a specific vehicle i as:

$$R_i(t) = R_i^{\text{cp}}(t) + R_i^{\text{ca}}(t). \tag{17}$$

$R_i^{\text{cp}}(t)$ and $R_i^{\text{ca}}(t)$ can be calculated in two scenarios. In scenario 1, only RSUs have the caching storage and computing capability while users do not have content caching storage or computing capability. In scenario 2, both RSUs and vehicles have the caching storage and computing capability.

1) Calculation of $R_i^{ca}(t)$ in Scenario 1: Let matrix $\mathbb{X}_{K\times C}$ define the caching strategy of the encoded segments in the RSUs, where $x_{k,c} \in \mathbb{X}$ represents the number of coded segments in RSU k, i.e., $x_{k,c} = a_{i,c}^{\text{ca}} \times a_{i,c}^{\text{cd}}$. Let $U_i^c(\mathbb{X})$ denote the total amount of coded segments that vehicle i can obtain from the RSUs within T_d.

Next, we give the solution to $\Pr(U_i^c(\mathbb{X}) = a)$, where a is the number of coded segments that vehicle i obtains from the RSUs. Within time T_d, let $M_{i,k}$ denote the number of contacts between vehicle i and RSU k, where $M_{i,k}$ is a random variable following Poisson distribution with $\lambda_{i,k}$. Then, the total size of the file that vehicle i can obtain from RSU k within T_d can be calculated as:

$$V_{i,k}^c(\mathbb{X}) = \sum_{n=1}^{M_{i,k}} \mathcal{A}_{i,k}^n. \tag{18}$$

Here, $\mathcal{A}_{i,k}^n$ is the maximum amount of content received at vehicle i from RSU k, which follows an exponential distribution with parameter $A_{i,k}$. Note that $A_{i,k}$ can be calculated by $A_{i,k} = T_{\text{con},k}^R b_{i,k}^R \nu_{i,k}^R$, where $T_{\text{con},k}^R$, $b_{i,k}^R$ and $\nu_{i,k}^R$ are the average duration of every contact, the spectrum bandwidth, and the achievable spectrum efficiency of the link between vehicle i and RSU k, respectively.

Since the number of contacts follows the Poisson distribution, the average number of contacts for vehicle i with RSU k can be denoted as $\lambda_{i,k} T_d$. Through such a simplification, we can obtain

$$V_{i,k}^c(\mathbb{X}) = \sum_{n=1}^{\lambda_{i,k} T_d} \mathcal{A}_{i,k}^n. \tag{19}$$

Since $\mathcal{A}_{i,k}^n$ ($n = 1, 2, \ldots, \lambda_{i,k} T_d$) is a collection of independent and identically distributed random variables, according to [44],

Authorized licensed use limited to: China University of Petroleum. Downloaded on December 20,2020 at 19:16:57 UTC from IEEE Xplore. Restrictions apply.

we obtain $V_{i,k}^c(\mathbb{X}) \propto \Gamma(\lambda_{i,k}T_d, A_{i,k})$. Thus, the size of the file that vehicle i can obtain is

$$V_i^c(\mathbb{X}) = \sum_{k\in\mathcal{K}} V_{i,k}^c(\mathbb{X}). \tag{20}$$

Now, we determine the probability distribution function (PDF) of $f_{V_i^c(\mathbb{X})}(v)$. Let $f_{V_{i,k}^c(\mathbb{X})}(v)$ denote the PDF of variable $V_{i,k}^c(\mathbb{X})$. According to the above analysis, the PDF of $f_{V_i^c(\mathbb{X})}(v)$ is the discrete convolution of $f_{V_{i,k}^c(\mathbb{X})}(v)$ $(k = 1, 2, \ldots, K)$. Then,

$$\begin{aligned} f_{V_i^c(\mathbb{X})}(v) = &\frac{v^{\lambda_{i,1}T_d-1}e^{-vA_{i,1}}}{(A_{i,1})^{-\lambda_{i,1}T_d}\Gamma(\lambda_{i,1}T_d)} \\ &\otimes \ldots \otimes \frac{v^{\lambda_{i,K}T_d-1}e^{-vA_{i,K}}}{(A_{i,K})^{-\lambda_{i,K}T_d}\Gamma(\lambda_{i,K}T_d)}, \end{aligned} \tag{21}$$

where $\otimes$ is the operation of the discrete convolution. Due to the intrinsic difficulty feature to solve the convolution, Welch-Satterthwaite estimation [45] is used to obtain

$$f_{V_i^c(\mathbb{X})}(v) \approx \frac{v^{\gamma_1-1}e^{-v\sigma_1}}{\sigma_1^{-\gamma_1}\Gamma(\gamma_1)}, \tag{22}$$

where

$$\gamma_1 = \frac{(\sum_{k\in\mathcal{K}}\lambda_{i,k}T_dA_{i,k})^2}{\sum_{k\in\mathcal{K}}\lambda_{i,k}T_d(A_{i,k})^2}, \tag{23}$$

$$\sigma_1 = \frac{\sum_{k\in\mathcal{K}}\lambda_{i,k}T_d(A_{i,k})^2}{\sum_{k\in\mathcal{K}}\lambda_{i,k}T_dA_{i,k}}. \tag{24}$$

Furthermore, random variable $V_i^c(\mathbb{X})$ can be estimated as a Gamma distribution, i.e., $V_i^c(\mathbb{X}) \propto \Gamma(\gamma_1, \sigma_1)$. Note that $U_i^c = \min(\lfloor V_i^c/l_c \rfloor, s_c)$, where l_c is the length of each coded segment. So the probability $\mathcal{P}_1(a) = \Pr(U_i^c(\mathbb{X}) = a)$ that vehicle i receives a coded segments is

$$\mathcal{P}_1(a) = \begin{cases} \int_{l_c a}^{l_c(a+1)} f_{V_i^c(\mathbb{X})}(v)dv & 0 \le a < \sum_{k\in\mathcal{K}} x_{k,c}, \\ \int_{l_c a}^{\infty} f_{V_i^c(\mathbb{X})}(v)dv & a = \sum_{k\in\mathcal{K}} x_{k,c}, \\ 0 & \text{otherwise.} \end{cases} \tag{25}$$

$R_i^{\text{ca}}(t)$ is expressed as

$$\begin{aligned} R_i^{\text{ca}}(t) = &-\sum_{c\in\mathcal{C}}\xi_c\sum_{k\in\mathcal{K}}x_{k,c} + \left(1 - \sum_{a=1}^{s_c}\mathcal{P}_1(a)\right)s_c\delta^R b^R\nu^R \\ &+ \sum_{a=1}^{s_c}\mathcal{P}_1(a)(a\delta^R b^R\nu^R - (s_c-a)^+\delta^{BS}b^{BS}\nu^{BS}), \end{aligned} \tag{26}$$

where $(a)^+ = \max(0, a)$. Note that b^R and ν^R are the minimum spectrum bandwidth and the minimum achievable spectrum efficiency of the link between vehicle i and RSU k, respectively. Similarly, b^{BS} and ν^{BS} are the minimum assigned spectrum bandwidth and the minimum achievable spectrum efficiency of the link between vehicle i and the BS. Finally, δ^R and δ^{BS} are the communication costs when vehicle i download coded packets from the RSU and the BS, respectively.

2) Calculation of $R_i^{ca}(t)$ in Scenario 2: Let matrix $\mathbb{Y}_{U\times C}$ define the caching strategy of the encoded segments in vehicles, where $y_{u,c} \in \mathbb{Y}$ represents the number of code segments in vehicle u . Let $U_i^c(\mathbb{X}, \mathbb{Y})$ denote the total amount of code segments that vehicle i can obtain from RSU k and from vehicle u within T_d.

Next, we calculate $\Pr(U_i^c(\mathbb{X}, \mathbb{Y}) = a)$, where a is the number of coded segments that vehicle i obtains. Within time T_d, let $N_{i,u}$ denote the number of contacts between vehicle i and vehicle u, where $N_{i,u}$ is a random variable following Poisson distribution with parameter $\chi_{i,u}$. Then, the total size of the file that vehicle i can obtain within T_d from vehicle u is calculated as:

$$V_{i,u}^c(\mathbb{Y}) = \sum_{n=1}^{N_{i,u}} \mathcal{B}_{i,u}^n, \tag{27}$$

where $\mathcal{B}_{i,u}^n$ is the maximum amount of content received at vehicle i from vehicle u, which follows an exponential distribution with parameter $B_{i,u}$. $B_{i,u}$ can be calculated by $B_{i,u} = T_{\text{con},u}^D b_{i,u}^D \nu_{i,u}^D$, where $T_{\text{con},u}^D$, $b_{i,u}^D$ and $\nu_{i,u}^D$ are the average duration of every contact, the spectrum bandwidth and the achievable spectrum efficiency of the link between vehicles i and u, respectively.

Since the number of contacts follows the Poisson distribution, the average number of contacts for vehicle i getting contact with vehicle u can be denoted as $\chi_{i,u}T_d$. Hence, we have

$$V_{i,u}^c(\mathbb{Y}) = \sum_{n=1}^{\chi_{i,u}T_d} \mathcal{B}_{i,u}^n. \tag{28}$$

Since $\mathcal{B}_{i,u}^n$ $(n = 1, 2, \ldots, \chi_{i,u}T_d)$ is a collection of independent and identically distributed random variables, according to [44], we obtain $V_{i,u}^c(\mathbb{Y}) \propto \Gamma(\chi_{i,u}T_d, B_{i,u})$. Thus, the size of the file that vehicle i can obtain from neighboring vehicles is expressed as

$$V_i^c(\mathbb{Y}) = \sum_{u\in\mathcal{U}} V_{i,u}^c(\mathbb{Y}). \tag{29}$$

So the total size of the file that vehicle i receives from other vehicles and from RSUs is expressed as

$$V_i^c(\mathbb{X}, \mathbb{Y}) = V_i^c(\mathbb{X}) + V_i^c(\mathbb{Y}), \tag{30}$$

where $V_i^c(\mathbb{X})$ is given in Eq. (20).

Now, we calculate the PDF of $f_{V_i^c(\mathbb{X},\mathbb{Y})}(v)$. Let $f_{V_{i,u}^c(\mathbb{Y})}(v)$ denote the PDF of variable $V_{i,u}^c(\mathbb{Y})$. Similar to Section V-C1, the PDF of $f_{V_i^c(\mathbb{X},\mathbb{Y})}(v)$ is the discrete convolution of $f_{V_{i,k}^c(\mathbb{X})}(v)$ $(k = 1, 2, \ldots, K)$ and $f_{V_{i,u}^c(\mathbb{Y})}(v)$ $(u = 1, 2, \ldots, i-1, i+1, \ldots, U)$. Then, we get

$$\begin{aligned} f_{V_i^c(\mathbb{X},\mathbb{Y})} = &f_{V_{i,1}^c(\mathbb{X})}(v) \otimes \ldots \otimes f_{V_{i,K}^c(\mathbb{X})}(v) \\ &\otimes f_{V_{i,1}^c(\mathbb{Y})}(v) \otimes \ldots \otimes f_{V_{i,i-1}^c(\mathbb{Y})}(v) \\ &\otimes f_{V_{i,i+1}^c(\mathbb{Y})}(v) \otimes \ldots \otimes f_{V_{i,U}^c(\mathbb{Y})}(v). \end{aligned} \tag{31}$$

Again, we can use Welch-Satterthwaite estimation [45] to obtain the PDF of $f_{V_i^c(\mathbb{X},\mathbb{Y})}(v)$. It implies that $V_i^c(\mathbb{X}, \mathbb{Y}) \propto \Gamma(\gamma_2, \sigma_2)$,

Authorized licensed use limited to: China University of Petroleum. Downloaded on December 20,2020 at 19:16:57 UTC from IEEE Xplore. Restrictions apply.

where

$$\gamma_2 = \frac{(\sum_{u\in\mathcal{U}} \chi_{i,u} T_d B_{i,u} + \sum_{k\in\mathcal{K}} \lambda_{i,k} T_d A_{i,k})^2}{\sum_{u\in\mathcal{U}} \chi_{i,u} T_d (B_{i,u})^2 + \sum_{k\in\mathcal{K}} \lambda_{i,k} T_d (A_{i,k})^2}, \tag{32}$$

$$\sigma_2 = \frac{\sum_{u\in\mathcal{U}} \chi_{i,u} T_d (B_{i,u})^2 + \sum_{k\in\mathcal{K}} \lambda_{i,k} T_d (A_{i,k})^2}{\sum_{u\in\mathcal{U}} \chi_{i,u} T_d B_{i,u} + \sum_{k\in\mathcal{K}} \lambda_{i,k} T_d A_{i,k}}, \tag{33}$$

$$f_{V_i^c(\mathbb{X},\mathbb{Y})}(v) \approx \frac{v^{\gamma_2-1} e^{-v\sigma_2}}{(\sigma_2)^{-\gamma_2}\Gamma(\gamma_2)}. \tag{34}$$

Note that $U_i^c(\mathbb{X},\mathbb{Y}) = \min\left(\lfloor V_i^c(\mathbb{X},\mathbb{Y})/l_c \rfloor, s_c\right)$. Hence, the probability $\mathcal{P}_2(a) = \Pr(U_i^c(\mathbb{X},\mathbb{Y}) = a)$ that vehicle i receives a coded packets can be written as

$$\mathcal{P}_2(a) = \begin{cases} \int_{l_c a}^{l_c(a+1)} f_{V_i^c(\mathbb{X},\mathbb{Y})}(v)dv & 0 \le a < \overline{a}, \\ \int_{l_c a}^{\infty} f_{V_i^{V,c}(\mathbb{X},\mathbb{Y})}(v)dv & a = \overline{a}, \\ 0 & \text{otherwise}, \end{cases} \tag{35}$$

where $\overline{a} = \sum_{k\in\mathcal{K}} x_{k,c} + \sum_{u\in\mathcal{U}} y_{u,c}$. $R_i^{\text{ca}}(t)$ is expressed as

$$\begin{aligned} R_i^{\text{ca}}(t) = &- \sum_{c\in\mathcal{C}} \xi_c \left(\sum_{k\in\mathcal{K}} x_{k,c} + \sum_{u\in\mathcal{U}} y_{u,c} \right) \\ &+ \left(1 - \sum_{a=0}^{s_c} \mathcal{P}_2(a)\right) s_c \delta^R b^R \nu^R \\ &+ \sum_{a=0}^{s_c} \mathcal{P}_2(a)(a\delta^R b^R \nu^R + (s_c - a)^+ \delta^{BS} b^{BS} \nu^{BS}). \end{aligned} \tag{36}$$

3) Calculation of $R_i^{cp}(t)$ in Scenario 1: The vehicle i requires that the computation for the content c must be done within the duration of T_d. Define s_w^c as the number of tasks for content the c. For every task w of the content c, l_w^c denotes the length of data that includes the program code and the input parameters and D_w^c represents the number of CPU cycles that are needed to accomplish the task w. The computing capacity of RSU k reserved to vehicle i is $f_{i,k}$ (cycles per second) and can be translated to the rate $R_{i,k}^{w,c} = f_{i,k} l_w^c / D_w^c$ (bits per second). To do so, we assume that the length/size of task is proportional to the number of CPU cycles that are required to accomplish the task. At the beginning of an epoch, we reserve the subset of RSUs for computing for vehicle i by $a_i^{\text{cp}} = [a_{k,c}^{\text{cp}}]$. Then, the computing capacity for K RSUs is $(Z)_{K\times C} = [R_{i,k}^{w,c} \times a_{k,c}^{\text{cp}}]$. It is readily observed that after making the translation of the computing capacity, the same scheme in Section V-C1 can be applied for the following derivations.

Let $U_i^c(\mathbb{Z})$ denote the total amount of tasks that can be computed by the RSUs (the MEC servers) within T_d. Next, we determine the solution of $\Pr(U_i^c(\mathbb{Z}) = a)$, where a is the number of tasks that are computed by the RSUs. Let $M_{i,k}$ denote the number of contacts between vehicle i and RSU k within T_d, where $M_{i,k}$ is a random variable following Poisson distribution with the parameter of $\lambda_{i,k}$. Then, the total size, which vehicle i can be offloaded to RSU k within T_d, can be calculated as:

$$V_{i,k}^c(\mathbb{Z}) = \sum_{n=1}^{M_{i,k}} \mathcal{H}_{i,k}^n. \tag{37}$$

Here $\mathcal{H}_{i,k}^n$ is the maximum amount of content offloaded to RSU k and it follows an exponential distribution with the parameter $H_{i,k}$. $H_{i,k}$ can be calculated as follows. For one contact with the duration $T_{\text{con},k}^R$, vehicle i must transmit the amount of $H_{i,k}$ bits to RSU k and RSU k must accomplish the computing for this amount of data. So we have

$$T_{\text{con},k}^R = \frac{H_{i,k}}{R_{i,k}} + \frac{H_{i,k}}{R_{i,k}^{w,c}}, \tag{38}$$

where $R_{i,k} = b_{i,k}^R \nu_{i,k}^R$ is the communication rate, $R_{i,k}^{w,c}$ is the computing rate. Then, we obtain

$$H_{i,k} = \frac{T_{\text{con},k}^R}{\frac{1}{R_{i,k}^{w,c}} + \frac{1}{R_{i,k}}}. \tag{39}$$

Since the number of contacts follows the Poisson distribution, the average number of contacts for vehicle i getting contact with RSU k can be denoted as $\lambda_{i,k} T_d$. Through such a simplification, we can obtain

$$V_{i,k}^c(\mathbb{Z}) = \sum_{n=1}^{\lambda_{i,k} T_d} \mathcal{H}_{i,k}^n. \tag{40}$$

Since $\mathcal{H}_{i,k}^n$ $(n = 1, 2, \ldots, \lambda_{i,k} T_d)$ is a collection of independent and identically distributed random variables, according to [44], we obtain $V_{i,k}^c(\mathbb{Z}) \propto \Gamma(\lambda_{i,k} T_d, H_{i,k})$. Thus, vehicle i can offload the size of the file c as follows:

$$V_i^c(\mathbb{Z}) = \sum_{k\in\mathcal{K}} V_{i,k}^c(\mathbb{Z}). \tag{41}$$

Now, we find the PDF of $f_{V_i^c(\mathbb{Z})}(v)$. Let $f_{V_{i,k}^c(\mathbb{Z})}(v)$ be the PDF of variable $V_{i,k}^c(\mathbb{Z})$. According to the above analysis, the PDF of $f_{V_i^c(\mathbb{Z})}(v)$ is the discrete convolution of $f_{V_{i,k}^c(\mathbb{Z})}(v)$ $(k = 1, 2, \ldots, K)$. Then,

$$f_{V_i^c(\mathbb{Z})}(v) = f_{V_{i,1}^c(\mathbb{Z})}(v) \otimes \ldots \otimes f_{V_{i,K}^c(\mathbb{Z})}(v). \tag{42}$$

So random variable $V_i^c(\mathbb{Z})$ follows the Gamma distribution, i.e. $V_i^c(\mathbb{Z}) \propto \Gamma(\gamma_3, \sigma_3)$. Then, we have

$$f_{V_i^c(\mathbb{Z})}(v) \approx \frac{v^{\gamma_3-1} e^{-v\sigma_3}}{\sigma_3^{-\gamma_3}\Gamma(\gamma_3)}, \tag{43}$$

$$\gamma_3 = \frac{(\sum_{k\in\mathcal{K}} \lambda_{i,k} T_d H_{i,k})^2}{\sum_{k\in\mathcal{K}} \lambda_{i,k} T_d (H_{i,k})^2}, \tag{44}$$

$$\sigma_3 = \frac{\sum_{k\in\mathcal{K}} \lambda_{i,k} T_d (H_{i,k})^2}{\sum_{k\in\mathcal{K}} \lambda_{i,k} T_d H_{i,k}}. \tag{45}$$

Note that $U_i^c = \min(\lfloor V_i^c / l_w^c \rfloor, s_w^c)$, where l_w^c is the length of each task. So the probability $\mathcal{P}_3(a) = \Pr(U_i^c(\mathbb{Z}) = a)$ that

Authorized licensed use limited to: China University of Petroleum. Downloaded on December 20,2020 at 19:16:57 UTC from IEEE Xplore. Restrictions apply.

vehicle i offloads a tasks is

$$\mathcal{P}_3(a)=\begin{cases}\int_{l_w^c a}^{l_w^c (a+1)} f_{V_i^c(\mathbb{Z})}(v)dv & 0\le a<\frac{T_d}{K}\sum_{k\in\mathcal{K}} z_{k,c},\\ \int_{l_w^c a}^{\infty} f_{V_i^c(\mathbb{X})}(v)dv & a=\frac{T_d}{K}\sum_{k\in\mathcal{K}} z_{k,c},\\ 0 & \text{otherwise.}\end{cases} \tag{46}$$

$R_i^{\text{cp}}(t)$ is expressed as

$$R_i^{\text{cp}}(t)=-\sum_{c\in\mathcal{C}}\sum_{a=1}^{s_w^c}\mathcal{P}_3(a)(a\mathcal{C}^R-(s_w^c-a)^+\mathcal{C}^{BS}) +\left(1-\sum_{a=1}^{s_w^c}\mathcal{P}_3(a)\right)s_w^c\mathcal{C}^R, \tag{47}$$

where $(a)^+=\max(0,a)$. Note that $\mathcal{C}^R=\delta^R b^R\nu^R+\eta^R D_w^c e^R$ is the total offloading cost to RSUs for one task, which include the communication cost, $\delta^R b^R\nu^R$ and the computing cost, $\eta^R D_w^c e^R$. Similarly, $\mathcal{C}^{BS}=\delta^{BS}b^{BS}\nu^{BS}+\eta^{BS}D_w^c e^{BS}$ is the total offloading cost to BSs for one task, which includes the communication cost, $\delta^{BS}b^{BS}\nu^{BS}$ and the computing cost, $\eta^{BS}D_w^c e^{BS}$.

4) Calculation of $R_i^{cp}(t)$ in Scenario 2: In this section, we consider the case that the vehicles have the computing capability. Similar to Section V-C3, the computing capacity $f_{i,u}$ (cycles per second) of vehicle u reserved to vehicle i can be converted to the rate $R_{i,u}^{w,c}=f_{i,u}l_w^c/D_w^c$ (bits per second). Furthermore, at the beginning of each epoch, we reserve the subset of the possible vehicles for computing for vehicle i by $a_i^{\text{cp}}=[a_{u,c}^{\text{cp}}]$. So the computing capacity for U vehicles is $\mathbb{W}_{U\times C}=[R_{i,u}^{w,c}\times a_{u,c}^{\text{cp}}]$.

Let $U_i^c(\mathbb{Z},\mathbb{W})$ denote the total amount of data that vehicle i offloads to RSU k and vehicle u within T_d. Next, we will derive $\Pr(U_i^c(\mathbb{Z},\mathbb{W})=a)$, where a is the number of offloaded tasks. The total size of the file c, which vehicle i can offload to vehicle u within T_d, can be calculated as:

$$V_{i,u}^c(\mathbb{W})=\sum_{n=1}^{\chi_{i,u}T_d}\mathcal{L}_{i,u}^n, \tag{48}$$

where $\mathcal{L}_{i,u}^n$ is the maximum amount of content received at vehicle i from vehicle u, which follows an exponential distribution with parameter $L_{i,u}$. Similar to Section V-C3, $L_{i,u}$ can be calculated by $L_{i,u}=\frac{T_{\text{con},k}^U}{\frac{1}{R_{i,u}^{w,c}}+\frac{1}{R_{i,u}}}$, where $R_{i,u}=b_{i,u}^U\nu_{i,u}^U$ is the communication rate. Moreover, $N_{i,u}=\chi_{i,u}T_d$ denotes the number of contacts between vehicle i and vehicle u, where $N_{i,u}$ is a random variable following Poisson distribution with parameter $\chi_{i,u}$. Here, we observe that $V_{i,u}^c(\mathbb{W})\propto\Gamma(\chi_{i,u}T_d,L_{i,u})$.

So vehicle i can offload the task size of the file c to nearby vehicles as follows:

$$V_i^c(\mathbb{W})=\sum_{u\in\mathcal{U}}V_{i,u}^c(\mathbb{W}). \tag{49}$$

So the task size of the file c that vehicle i offloads to nearby vehicles and RSUs is expressed as

$$V_i^c(\mathbb{Z},\mathbb{W})=V_i^c(\mathbb{Z})+V_i^c(\mathbb{W}), \tag{50}$$

where $V_i^c(\mathbb{Z})$ is given in Eq. (41).

Now, we calculate the PDF of $f_{V_i^c(\mathbb{Z},\mathbb{W})}(v)$. Let $f_{V_{i,u}^c(\mathbb{W})}(v)$ denote the PDF of variable $V_{i,u}^c(\mathbb{W})$. Similar to Section V-C3, the PDF of $f_{V_i^c(\mathbb{Z},\mathbb{W})}(v)$ is the discrete convolution of $f_{V_{i,k}^c(\mathbb{Z})}(v)$ $(k=1,2,\ldots,K)$ and $f_{V_{i,u}^c(\mathbb{W})}(v)$ $(u=1,2,\ldots,i-1,i+1,\ldots,U)$. Then, we can use Welch-Satterthwaite estimation [45] to obtain the PDF of $f_{V_i^c(\mathbb{Z},\mathbb{W})}(v)$, where $V_i^c(\mathbb{Z},\mathbb{W})\propto\Gamma(\gamma_4,\sigma_4)$ and

$$\gamma_4=\frac{(\sum_{u\in\mathcal{U}}\chi_{i,u}T_dL_{i,u}+\sum_{k\in\mathcal{K}}\lambda_{i,k}T_dH_{i,k})^2}{\sum_{u\in\mathcal{U}}\chi_{i,u}T_d(L_{i,u})^2+\sum_{k\in\mathcal{K}}\lambda_{i,k}T_d(H_{i,k})^2}, \tag{51}$$

$$\sigma_4=\frac{\sum_{u\in\mathcal{U}}\chi_{i,u}T_d(L_{i,u})^2+\sum_{k\in\mathcal{K}}\lambda_{i,k}T_d(H_{i,k})^2}{\sum_{u\in\mathcal{U}}\chi_{i,u}T_dL_{i,u}+\sum_{k\in\mathcal{K}}\lambda_{i,k}T_dH_{i,k}}, \tag{52}$$

$$f_{V_i^c(\mathbb{Z},\mathbb{W})}(v)\approx\frac{v^{\gamma_4-1}e^{-v\sigma_4}}{(\sigma_4)^{-\gamma_4}\Gamma(\gamma_4)}. \tag{53}$$

Note that $U_i^c(\mathbb{Z},\mathbb{W})=\min\left(\lfloor V_i^c(\mathbb{Z},\mathbb{W})/l_w^c\rfloor,s_w^c\right)$. Hence, the probability $\mathcal{P}_4(a)=\Pr(U_i^c(\mathbb{Z},\mathbb{W})=a)$ that vehicle i offloads a tasks can be written as

$$\mathcal{P}_4(a)=\begin{cases}\int_{l_w^c a}^{l_w^c (a+1)} f_{V_i^c(\mathbb{Z},\mathbb{W})}(v)dv & 0\le a<\overline{a},\\ \int_{l_w^c a}^{\infty} f_{V_i^{V,c}(\mathbb{Z},\mathbb{W})}(v)dv & a=\overline{a},\\ 0 & \text{otherwise,}\end{cases} \tag{54}$$

where $\overline{a}=\frac{T_d}{K}\sum_{k\in\mathcal{K}}z_{k,c}+\frac{T_d}{U}\sum_{k\in\mathcal{U}}w_{u,c}$. So $R_i^{\text{cp}}(t)$ is expressed as

$$R_i^{\text{cp}}(t)=-\sum_{c\in\mathcal{C}}\sum_{a=1}^{s_w^c}\mathcal{P}_4(a)(a\mathcal{C}^R-(s_w^c-a)^+\mathcal{C}^{BS}) +\left(1-\sum_{a=1}^{s_w^c}\mathcal{P}_4(a)\right)s_w^c\mathcal{C}^R, \tag{55}$$

where $(a)^+=\max(0,a)$. Note that $\mathcal{C}^R=\delta^R b^R\nu^R+\eta^R D_w^c e^R$ is the cost of offloading to RSUs for one task including the communication cost, $\delta^R b^R\nu^R$ and the computing cost, $\eta^R D_w^c e^R$. Here, we assume that the cost for the nearby vehicles is equal to that for the RSUs for simplicity. Similarly, $\mathcal{C}^{BS}=\delta^{BS}b^{BS}\nu^{BS}+\eta^{BS}D_w^c e^{BS}$ is the cost of offloading to the BS for one task including the communication cost, $\delta^{BS}b^{BS}\nu^{BS}$ and the computing cost, $\eta^{BS}D_w^c e^{BS}$.

VI. The Small Timescale Deep Q-Learning Model

Given the action a_i^{ca}, a_i^{cp}, and a_i^{cd} at the large timescale deep Q-learning model, we perform the actions $\tilde{a}_i^{\text{cp}}(t)$ and $\tilde{a}_i^{\text{ca}}(t)$ according to the states in every time slot. The vehicle's mobility has an impact on the actions of $\tilde{a}_i^{\text{cp}}(t)$ and $\tilde{a}_i^{\text{ca}}(t)$ as well as on the immediate reward.

A. System States

The state of the available RSU/MEC servers $k\in\{1,2,\ldots,K\}$, the available vehicles $u\in\{1,2,\ldots,U\}$ and the available caches $c\in\{1,2,\ldots,C\}$ for vehicle i in time slot t (with the duration of one small time slot) is determined by the realization of the states of the random variables γ_i^k (γ_i^u), the

Authorized licensed use limited to: China University of Petroleum. Downloaded on December 20,2020 at 19:16:57 UTC from IEEE Xplore. Restrictions apply.

realization of the states of the random variables $f_{i,k}$ ($f_{i,u}$) and the realization of the states of the random variables ς_c.

B. System Actions

Within the time slot, the agent has to decide which RSU and/or vehicle is assigned to the requesting vehicle and whether or not the computation task should be offloaded to the RSU/MEC server and/or the vehicle. Note that the caching placement is performed in the large timescale model and the amount of each content dynamically changes as in Section II-D.

The current composite action $\tilde{a}_i(t)$ is denoted by

$$\tilde{a}_i(t) = \left\{\tilde{a}_i^{\text{cp}}(t), \tilde{a}_i^{\text{ca}}(t)\right\}, \tag{56}$$

where $\tilde{a}_i^{\text{cp}}(t)$ and $\tilde{a}_i^{\text{ca}}(t)$ are defined and interpreted as follows:

- For the communication control, define $(K+U)\times C$ matrix $\tilde{a}_i^{\text{ca}}(t) = \tilde{a}_{(K+U)\times C}^{\text{ca}}$, where each element $\tilde{a}_{k,c}^{\text{ca}}(t)$ (or $\tilde{a}_{u,c}^{\text{ca}}(t)$) represents the connection control of RSU k (or vehicle u) and vehicle i for the c-th content cache and $\tilde{a}_{k,c}^{\text{ca}}(t), \tilde{a}_{u,c}^{\text{ca}}(t) \in \{0,1\}$. Here, $\tilde{a}_{k/u,c}^{\text{ca}}(t) = 0$ means that the content c is not cached at RSU k (or vehicle u) at time slot t or RSU k (or vehicle u) is out of the communication range of vehicle i during time slot t, while $\tilde{a}_{k/u,c}^{\text{ca}}(t) = 1$ means that the content c is cached at RSU k (or vehicle u) and RSU k (or vehicle u) is the best contact for vehicle i during time slot t. Note that if none of vehicles has caching storage capability, we completely remove the vehicle component from all the above parameters.
- For the computing control, define $(K+U)\times C$ matrix $\tilde{a}_i^{\text{cp}}(t) = \tilde{a}_{(K+U)\times C}^{\text{cp}}$, where each element $\tilde{a}_{k/u,c}^{\text{cp}}(t)$ represents the connection between the MEC server k (or vehicle u) and vehicle i for the data offloading and $\tilde{a}_{k/u,c}^{\text{cp}}(t) \in \{0,1\}$. Here, $\tilde{a}_{k/u,c}^{\text{cp}}(t) = 0$ means that the task is not offloaded to the MEC server k (or vehicle u) at time slot t, or RSU k (or vehicle u) is out of the communication range of vehicle i during time slot t, while $\tilde{a}_{i,k}^{\text{cp}}(t) = 1$ means that the task is offloaded and RSU k (or vehicle u) is the best contact for vehicle i during time slot t. Again, if none of the vehicles has computing resources, we completely remove the vehicle component from all the above parameters.

C. Reward Function

We will calculate the exact immediate reward for the actions. This quantity is written as follows:

$$\tilde{R}_i(t) = \tilde{R}_i^{\text{cp}}(t) + \tilde{R}_i^{\text{ca}}(t). \tag{57}$$

$\tilde{R}_i^{\text{cp}}(t)$ and $\tilde{R}_i^{\text{ca}}(t)$ can be calculated as

$$\begin{aligned}\tilde{R}_i^{\text{ca}}(t) = &-\sum_{c\in\mathcal{C}}\xi_c\left(\sum_{k\in\mathcal{K}}x_{k,c}+\sum_{u\in\mathcal{U}}y_{u,c}\right)\\ &+\sum_{k\in\mathcal{K}}\tilde{a}_{k,c}^{\text{ca}}(t)\delta^R b_{i,k}^R\nu_{i,k}^R+\sum_{u\in\mathcal{U}}\tilde{a}_{u,c}^{\text{ca}}(t)\delta^R b_{i,u}^U\nu_{i,u}^U\\ &+\left(1-\sum_{u\in\mathcal{U}}\tilde{a}_{u,c}^{\text{ca}}(t)-\sum_{k\in\mathcal{K}}\tilde{a}_{k,c}^{\text{ca}}(t)\right)\delta^{BS}b^{BS}\nu^{BS},\end{aligned} \tag{58}$$

$$\begin{aligned}\tilde{R}_i^{\text{cp}}(t) = &-\sum_{c\in\mathcal{C}}\sum_{k\in\mathcal{K}}\tilde{a}_{k,c}^{\text{cp}}(t)\tilde{\mathcal{C}}^R+\sum_{u\in\mathcal{U}}\tilde{a}_{u,c}^{\text{cp}}(t)\tilde{\mathcal{C}}^U\\ &+\left(1-\sum_{k\in\mathcal{K}}\tilde{a}_{k,c}^{\text{cp}}(t)-\sum_{u\in\mathcal{U}}\tilde{a}_{u,c}^{\text{cp}}(t)\right)\mathcal{C}^{BS}),\end{aligned} \tag{59}$$

where $\tilde{\mathcal{C}}^R = \delta^R b_{i,k}^R\nu_{i,k}^R + \eta^R R_{i,k}^{w,c}e^R$ is the cost of offloading to RSUs including the communication cost and the computing cost. Similarly, $\tilde{\mathcal{C}}^U = \delta^U b_{i,u}^U\nu_{i,u}^U + \eta^U R_{i,u}^{w,c}e^U$ is the cost of offloading to vehicles and $\mathcal{C}^{BS} = \delta^{BS}b^{BS}\nu^{BS} + \eta^{BS}D_w^c e^{BS}$ is the cost of offloading to the BS. Recall that if the vehicles do not have the caching storage and the computing resources, we remove the vehicle part in all the above expressions.

Remark 1: For simplicity, we do not consider performance analysis of the other important metric such as latency. This enables us to gain insight into the investigated mobility-aware coded probabilistic caching and computational offloading problem, while keeping the problem tractable. In this work, we only use a reward (or a cost) of communication, computation and data offloading as the main optimization objective. However, the formulated optimization problem also considers the hard deadline T_d of the requested content. It means that we aim to maximize the contributed reward of communication, computation and data offloading, while guaranteeing the hard deadline T_d. These two performance metrics are very important for a VANET in the sense that the services of vehicles are always satisfied with low cost. Note that analysis performed in the above content can be extended to consider the latency performance. However, the extension of the analysis model to capture this important metric will be considered in our future works, because we should focus on the primary purpose as well as satisfy the space limit. Relevant results reported in some recent works such as those in [16], [18], [19] would be useful for these further studies.

VII. Numerical Results

This section presents the numerical results to illustrate the cost performance of the proposed resource allocation of communication, computing and caching. The key parameters are chosen as follows, unless stated otherwise. $K = 10$; $U = 50$; $W_u = 10$ MHz; $4b_{i,k}^{BS} = b_{i,k}^R = b_{u,k}^U = 1$ MHz; $\xi_c = 2$ units/MB; $\delta^{BS} = 10\delta^R = 10\delta^U = 20$ units/MHz; $\eta^{BS} = 10\eta^R = 10\eta^U = 100$ units/J; $e^{BS} = e^R = e^U = 1$ W/GHz; $D_w^c = 100$ Mcycles; $l_c/l_w^c = 20$; $f_{i,k}$ in the range of $[10, 20]$ GHz; $f_{i,u}$ in the range of $[1, 10]$ GHz. Note that the cost when using the services of the BS (for downloading cached contents and computing their corresponding tasks) is ten times of the cost when using the services of the RSUs or the vehicles. Detailed explanation of our model is illustrated in Fig. 1. In particular, we provide an example, which illustrates the movement of the tagged requesting user (the red vehicle) and its data offloading and computational offloading during the vehicular mobility.

The wireless channels, i.e., vehicle-to-vehicle and vehicle-to-RSU links, all follow the Markov model. During the contact time of vehicle i to RSU k (or the nearby vehicle u), $\nu_{i,k}$ has two states, i.e., $\nu_{i,k} = \{1, 4\}$ (or $\nu_{i,u} = \{1, 4\}$), with 1 corresponding to the bad channel and 4 for the good channel. The probabilities of remaining in the same state and the transiting from one state to the other are respectively set to 0.7 and 0.3.

Authorized licensed use limited to: China University of Petroleum. Downloaded on December 20,2020 at 19:16:57 UTC from IEEE Xplore. Restrictions apply.

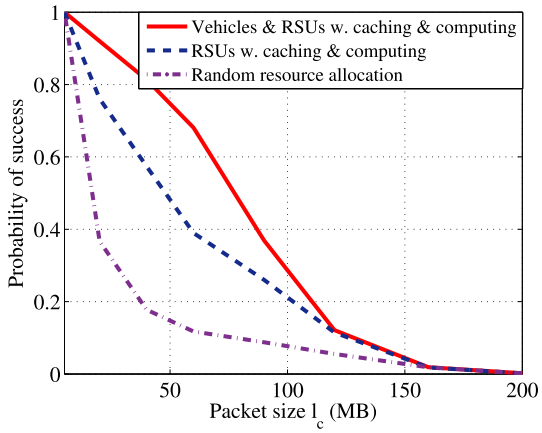


Fig. 2. Probability of success vs coded packet size l_c for $\lambda T_d = 0.1$ and $\chi T_d = 8$.

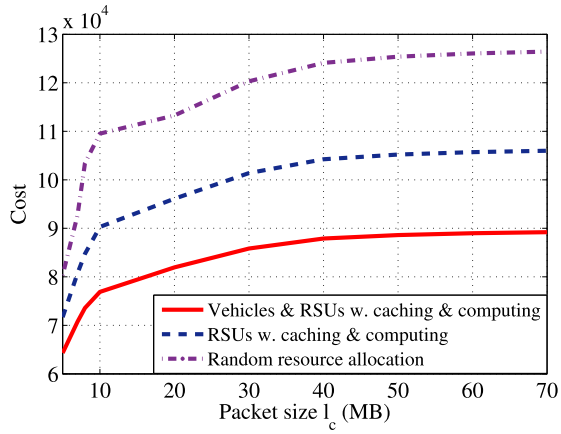


Fig. 3. Cost vs coded packet size l_c for $\lambda T_d = 0.05$, and $\chi T_d = 4$.

Similarly, the probabilities for caching are set to 0.7 and 0.3. The computation states of the MEC servers (or the vehicles) follow the Markov model, where the transition probabilities $\Theta_{i,j}$ are in the range of $[0, 1]$ and $\sum_{i,j} \Theta_{i,j} = 1$. The following simulation results validate the theoretical findings. More results are omitted here owing to the space limit, but motivated readers can find the details in [50].

To implement deep reinforcement learning, we employ the powerful open source software library, i.e., Tensorflow 0.12.1 [49], which is developed by the Google Brain team. We briefly present it in the following, interested readers can find more detailed information in [49]. The machine learning system, namely TensorFlow, is effectively applied to the heterogeneous large-scale networks. In particular, the computation, states and actions can be represented by the dataflow graphs. It maps the nodes of a dataflow graph over many machines in a cluster as well as within a machine over multiple computational devices, which include multicore CPUs, general-purpose GPUs and custom-designed ASICs. Because of the open source, it allows developers to work on their novel optimizations and training algorithms based on the previous "parameter server" designs. TensorFlow has received a substantial attention from both academia and industry for a variety of applications, as it provides the training and inference of advanced deep neural networks.

TensorFlow is used to transform an ordinary Q-Network into a deep Q-Network (DQN) by making the following improvements. We first utilize the *tf.contrib.layers.convolution2d* function to create a constitutional layer, i.e. we transfer a single-layer network to a multi-layer convolutional network. Then, the experience replay is implemented to train our network by using the stored memories from its experience. In particular, the tuple of (state, action, reward, next-state), which is calculated from Sections V and VI, is used to store the experience. Finally, we separate the target network, which is used to generate the target Q-values. This is then used to compute the loss for every action during training. Detailed process is summarized in Algorithm 1.

In Fig. 2, we show the probability of success versus coded packet size l_c for $\lambda T_d = 0.1$ and $\chi T_d = 8$. Here, the event of success means that the tagged vehicle i completes downloading the requested content and offloading its corresponding tasks for computing within the hard deadline T_d, while avoiding the expensive assistance from the BS. We compare our proposed scheme with two cases: *1)* only RSUs have the caching and computing capabilities and the vehicles do not have these capabilities; and *2)* the random resource allocation. Similar to our proposed scheme, the random resource allocation also selects the sets of the possible RSUs and vehicles at every epoch in order to reduce the numbers of useless reserved RSUs or vehicles (i.e., reduce the cost of their caching storage). However, it pre-selects these sets as well as allocates caching contents and computing resources in a random manner. As expected, the probability of success decreases with the increase of the coded packet for all the schemes. When the size of coded segment is small, it is easy to transmit data within the contact duration. However, the success probability of random resource allocation decreases dramatically, when l_c reaches 40 MB, while our proposed scheme achieves a much higher success probability. This validates the beneficial contribution of the large timescale model design, where we carefully select the sets of necessary RSUs and vehicles for every epoch. A high number of supporting nodes causes inefficient learning for the small timescale model due to the large action space. On the other hand, if that number is too small, the requesting vehicle cannot receive enough required coded packets for the requested content and cannot accomplish its corresponding tasks within the hard deadline T_d. Moreover, our proposed scheme also outperforms the case that only RSUs have the caching and computing capabilities, as our proposed scheme fully exploits the computing resource diversity, content diversity and mobility diversity both amongst the RSUs and the vehicles.

Fig. 3 shows the cost versus coded packet size l_c for $\lambda T_d = 0.05$ and $\chi T_d = 4$. We can observe that the increase of the size of the coded segment increases the cost of the caching storage. Moreover, it is very hard for the tagged vehicle i to download the content and to offload its tasks under the constraints of vehicle's mobility and hard deadline, when the coded data size is large. Therefore, the tagged vehicle may miss the extensive required coded segments. These missed segments and their tasks can be downloaded and computed by the assistance of the BS with the expensive penalty. Again, our proposed scheme always provides the significant performance gain over the other cases, namely the case of only RSUs having the caching and computing capabilities and the random resource allocation case.

Authorized licensed use limited to: China University of Petroleum. Downloaded on December 20,2020 at 19:16:57 UTC from IEEE Xplore. Restrictions apply.

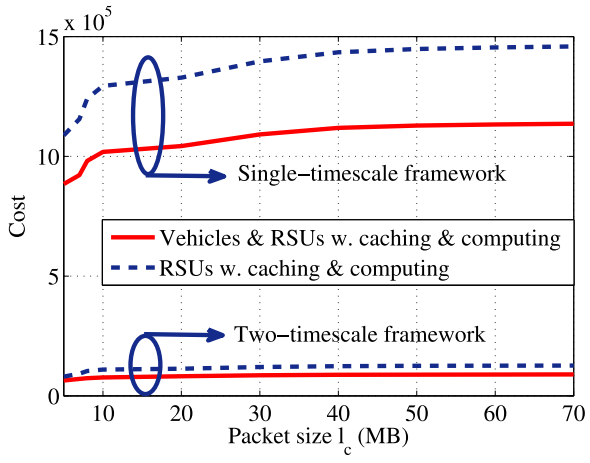


Fig. 4. Cost vs coded packet size l_c for $\lambda T_d = 0.05$, and $\chi T_d = 4$.

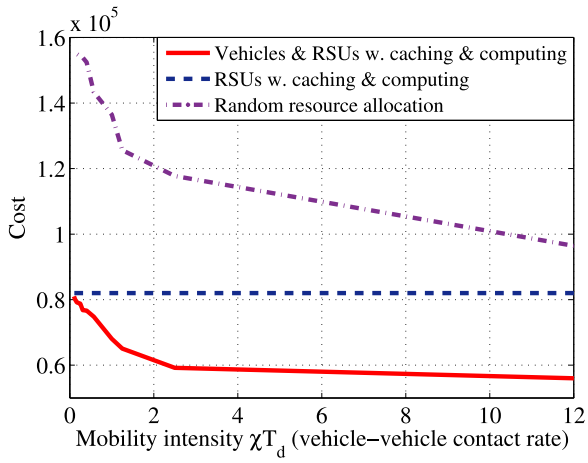


Fig. 5. Cost vs vehicle's mobility intensity, χT_d (vehicle-vehicle contact rate) for $l_c = 30$ MB and $\lambda T_d = 0.1$.

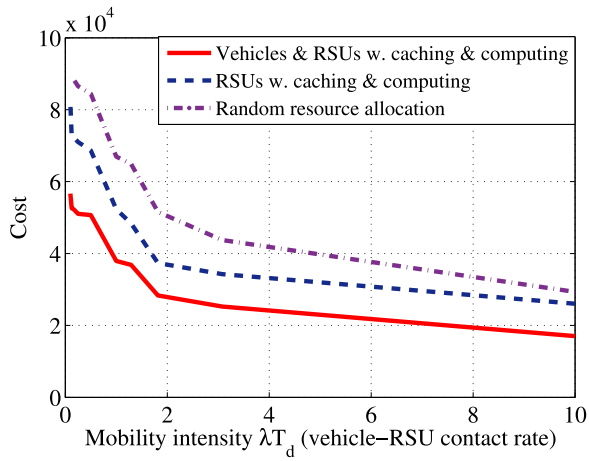


Fig. 6. Cost vs vehicle's mobility intensity, λT_d (vehicle-RSU contact rate) for $l_c = 30$ MB, $\chi T_d = 15$.

We now investigate the impact of multi-timescale framework on the system cost. Here, we make a comparison with the single-timescale scheme, where the pre-selection of the set of possible vehicles and RSUs is omitted. We set the parameters to $\lambda T_d = 0.05$ and $\chi T_d = 4$. For both single-timescale and multi-timescale frameworks, we consider two cases: *1)* only the RSUs have the caching and computing capabilities and the vehicles do not have these capabilities; *2)* both vehicles and RSUs have the caching and computing capabilities. We can observe in Fig. 4 that the system cost increases, when the coded data size is large. This is expected, as we can observe the results in Fig. 3. Moreover, our proposed multi-timescale framework offers a high cost gain, which can be explained as follows. In our proposed multi-timescale framework, we only determine caching for a pre-selected subset of RSUs and vehicles and hence the cost is small. In contrast, the single-timescale framework needs to allocate the coded segments to all RSUs and vehicles, hence the cost is significantly higher. Furthermore, there is a flexible caching cooperation in our proposed multi-timescale framework, where we can design different subsets of RSUs and vehicles as well as their caching contents for different requesting vehicles and for different requested contents. The action space becomes much smaller in the small timescale model after the pre-selection of RSUs and vehicles is performed in the large timescale model. However, the single-timescale framework only performs the homogeneous caching to reduce the action space. Thus, the implementation of the single-timescale framework receives a high complexity. So our proposed multi-timescale framework beneficially exploits the user mobility, distributed storage and caching cooperation amongst both the vehicles and the RSUs to achieve a high reward gain. From this point, we only consider the multi-timescale framework in the numerical results and ignore the case without pre-selecting the sets of possible RSUs and vehicles at every epoch (i.e., the single-timescale framework).

The impact of vehicle's mobility intensity, χT_d (vehicle-to-vehicle contact rate), on the cost performance is illustrated in Fig. 5. We consider the scenario with the following parameters, $l_c = 30$ MB and $\lambda T_d = 0.1$. It is readily seen that the curve for the case that only RSUs have caching and computing capabilities is constant with respect to vehicle's mobility intensity because vehicles that do not have caching storage and computation resources have no impact on cost performance. In contrast, our proposed scheme provides the substantial performance gain over other two schemes. With an increase of χT_d, the cost performance of both our proposed scheme and the random resource allocation scheme decreases. A high χT_d indicates a high mobility intensity and/or a large content deadline. If the content deadline is short, the vehicles can move at a high speed to keep the value of χT_d high. So we can draw a promising conclusion that our proposed scheme can exploit the vehicle's mobility to tackle the hard deadline constraints as well as to improve cost efficiency.

Similarly, the impact of mobility intensity on cost performance is demonstrated in Fig. 6. Here, we study the influence of the vehicle-to-RSUs contact rate λT_d with $l_c = 30$ MB and $\chi T_d = 15$. When the vehicle's mobility is low, the cost performance is high. This is because there are less opportunities for the vehicle to connect to the new RSUs. Hence, every vehicle and RSU need to cache more segments in order to meet the minimum amount of received coded segments for a successful decoding. However, our proposed scheme, which uses deep reinforcement learning and multi-timescale framework, can mitigate this

Authorized licensed use limited to: China University of Petroleum. Downloaded on December 20,2020 at 19:16:57 UTC from IEEE Xplore. Restrictions apply.

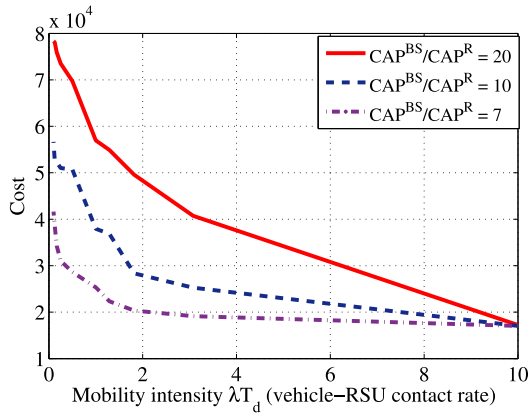


Fig. 7. Cost vs vehicle's mobility intensity, λT_d (vehicle-RSU contact rate), the backhaul capacities and cloud computing resources for $l_c = 30$ MB, $\chi T_d = 15$.

negative impact. In particular, it can coordinate the overlapping sets of contents for the vehicles and the RSUs.

Finally, we study the impact of backhaul capacities and cloud computing resources on the system cost. For the poor backhaul capacities and the low cloud computing resources, the cost of using the limited bandwidth to download cached content from the BS is high and using the cloud resources to compute its corresponding tasks are expensive. So the system imposes a penalty to the requesting vehicle, whenever it uses the limited backhaul capacities and cloud computing resources. We consider the scenario with $l_c = 30$ MB and $\chi T_d = 15$. For simplicity, we make two assumptions: *1)* the cost of using RSUs' resources equals that of using vehicles' resources (i.e., $\eta^R = \eta^U$ and $\delta^R = \delta^U$); *2)* the ratio of the communication costs equals the ratio of computing cost (i.e., $CAP^{BS}/CAP^R = \eta^{BS}/\eta^R = \delta^{BS}/\delta^R$). Here, CAP^{BS}/CAP^R is the ratio of the cost of using the BS's resources to the cost of using the RSUs' resources (or the vehicles' resources). In Fig. 7, CAP^{BS}/CAP^R is set to 7, 10 and 20, which correspond to low, medium and high backhaul and computing capabilities. With the increase of CAP^{BS}/CAP^R, the cost performance increases. This is in accordance with the fact that the poor capability on both the backhaul and BS computation degrades the system performance. It is also observed that when λT_d is large (i.e., the vehicle moves with a high speed), the performance curves follow the downward trend and the gap between the curves of low and high conditions becomes very small. In contrast, there is a big gap between the curves of low and high conditions at small λT_d. These observations indicate that our proposed scheme mitigates the limited backhaul capacity and low BS computation resource by effectively exploiting the distributed storage and computation as well as exploiting vehicle's mobility.

VIII. Conclusion

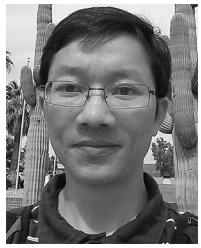
In this paper, we developed a framework of joint optimal resource allocation of communication, caching and computing for vehicular networks. In order to achieve the operational excellence and the cost efficiency, the vehicle's mobility was exploited to enhance caching and computing policies. We formulated the optimization problem for the resource allocation and proposed the deep reinforcement learning approach with the multi-timescale framework to solve this problem. Due to the complexity caused by the large action space, it is hard to implement the multi-timescale framework. Thus, we proposed the mobility-aware reward estimation for the large timescale model. Numerical results provided insights to the important theoretical findings and showed the significant performance gains achieved by the optimal parameter configurations for our proposed scheme.

References

[1] R. Q. Hu and Y. Qian, "An energy efficient and spectrum efficient wireless heterogeneous network framework for 5G systems," *IEEE Commun. Mag.*, vol. 52 no. 5, pp. 94–101, May 2014.
[2] L. Wei, R. Q. Hu, Y. Qian, and G. Wu, "Enable device-to-device communications underlaying cellular networks: Challenges and research aspects," *IEEE Commun. Mag.*, vol. 52, no. 6, pp. 90–96, Jun. 2014.
[3] C. Wang, Y. He, F. R. Yu, Q. Chen, and L. Tang, "Integration of networking, caching and computing in wireless systems: A survey, some research issues and challenges," *IEEE Commun. Surv. Tut.*, vol. 20, no. 1, pp. 7–38, First Quarter 2018.
[4] M. Cha, H. Kwak, P. Rodriguez, Y. Y. Ahn, and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1357–1370, Oct. 2009.
[5] P. Rodriguez, C. Spanner, and E. W. Biersack, "Analysis of web caching architectures: Hierarchical and distributed caching," in *IEEE/ACM Trans. Netw.*, vol. 9, no. 4, pp. 404–418, Aug. 2001.
[6] 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. Sys.*, vol. 17, no. 6, pp. 1509–1520, Jun. 2016.
[7] 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. Tech.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.
[8] R. Tandon and O. Simeone, "Harnessing cloud and edge synergies: Toward an information theory of fog radio access networks," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 44–50, Aug. 2016.
[9] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog-computing-based radio access networks: Issues and challenges," *IEEE Netw.*, vol. 30, no. 4, pp. 46–53, Jul. 2016.
[10] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 810–819, May 2017.
[11] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless D2D networks," *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 849–869, Feb. 2016.
[12] K. Ali, H. X. Nguyen, Q. T. Vien, P. Shah, and Z. Chu, "Disaster management using D2D communication with power transfer and clustering techniques," *IEEE Access*, vol. 6, pp. 14643–14654, Jan. 2018.
[13] A. Asadi and V. Mancuso, "Network-assisted outband D2D-clustering in 5G cellular networks: Theory and practice," *IEEE Trans. Mobile Comput.*, vol. 16, no. 8, pp. 2246–2259, Aug. 2017.
[14] R. Trestian, Q. T. Vien, H. X. Nguyen, and O. Gemikonakli, "ECO-M: Energy-efficient cluster-oriented multimedia delivery in a LTE D2D environment," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 55–61
[15] J. Liu, H. Nishiyama, N. Kato, and J. Guo, "On the outage probability of device-to-device-communication-enabled multichannel cellular networks: An RSS-threshold-based perspective," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 1, pp. 163–175, Jan. 2016.
[16] J. Li, H. Chen, Y. Chen, Z. Lin, B. Vucetic, and L. Hanzo, "Pricing and resource allocation via game theory for a small-cell video caching system," *IEEE J. Sel. Areas Commun*, vol. 34, no. 8, pp. 2115–2129, Aug. 2016.
[17] J. Hu, L. L. Yang, and L. Hanzo, "Energy-efficient cross-layer design of wireless mesh networks for content sharing in online social networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 9, pp. 8495–8509, Sep. 2017.
[18] J. Li, Y. Chen, Z. Lin, W. Chen, B. Vucetic, and L. Hanzo, "Distributed caching for data dissemination in the downlink of heterogeneous networks," *IEEE Trans. Commun.*, vol. 63, no. 10, pp. 3553–3568, Oct. 2015.

Authorized licensed use limited to: China University of Petroleum. Downloaded on December 20,2020 at 19:16:57 UTC from IEEE Xplore. Restrictions apply.

[19] Y. Chen, M. Ding, J. Li, Z. Lin, G. Mao, and L. Hanzo, "Probabilistic small-cell caching: Performance analysis and optimization," *IEEE Trans. Veh. Technol.*, vol. 66, no. 5, pp. 4341–4354, May 2017.
[20] Y. Zhou, F. R. Yu, J. Chen, and Y. Kuo, "Resource allocation for information-centric virtualized heterogeneous networks with in-network caching and mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 11339–11350, Dec. 2017.
[21] L. T. Tan and L. B. Le, "Compressed sensing based data processing and MAC protocol design for smartgrids," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2015, pp. 2138–2143.
[22] L. T. Tan and L. B. Le, "Joint data compression and MAC protocol design for smartgrids with renewable energy," *Wireless Commun. Mobile Comput.*, vol. 16, no. 16, pp. 2590–2604, Jul. 2016.
[23] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.
[24] Y. Guo, Q. Yang, F. R. Yu, and V. C. M. Leung, "Cache-enabled adaptive video streaming over vehicular networks: A dynamic approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5445–5459, Jun. 2018.
[25] Q. Chen, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "Joint resource allocation for software-defined networking, caching and computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 274–287, Feb. 2018.
[26] 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, no. 12, pp. 31–37, Dec. 2017.
[27] R. Wang, X. Peng, J. Zhang, and K. B. Letaief, "Mobility-aware caching for content-centric wireless networks: Modeling and methodology," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 77–83, Aug. 2016.
[28] K. Poularakis and L. Tassiulas, "Code, cache and deliver on the move: A novel caching paradigm in hyper-dense small-cell networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 3, pp. 675–687, Mar. 2017.
[29] R. Wang, J. Zhang, S. H. Song, and K. B. Letaief, "Mobility-aware caching in D2D networks," *IEEE Trans. Commun.*, vol. 16, no. 8, pp. 5001–5015, May 2017.
[30] B. N. Bharath, K. G. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," *IEEE Trans. Commun.*, vol. 64, no. 4, pp. 1674–1686, Apr. 2016.
[31] W. Wang, R. Lan, J. Gu, A. Huang, H. Shan, and Z. Zhang, "Edge caching at base stations with device-to-device offloading," *IEEE Access*, vol. 5, pp. 6399–6410, Mar. 2017.
[32] S. M. S. Tanzil, W. Hoiles, and V. Krishnamurthy, "Adaptive scheme for caching YouTube content in a cellular network: Machine learning approach," *IEEE Access*, vol. 5, pp. 5870–5881, Mar. 2017.
[33] H. Y. Ong, K. Chavez, and A. Hong, "Distributed deep Q-learning," 2015, arXiv:1508.04186.
[34] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[35] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep Q-learning with model-based acceleration," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2829–2838.
[36] J. Nie and S. Haykin, "A Q-learning-based dynamic channel assignment technique for mobile communication systems," *IEEE Trans. Veh. Technol.*, vol. 48, no. 5, pp. 1676–1687, Sep. 1999.
[37] L. T. Tan and H. Y. Kong, "A novel and efficient mixed-signal compressed sensing for wide-band cognitive radio," in *Proc. IEEE Int. Forum Strategic Technol.*, 2010, pp. 27–32.
[38] N. Kato *et al.*, "The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective," *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 146–153, Jun. 2017.
[39] Z. Fadlullah *et al.*, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Commun. Surv. Tuto.*, vol. 19, no. 4, pp. 2432–2455, Fourth Quarter 2017.
[40] L. T. Tan and L. B. Le, "Design and optimal configuration of full-duplex MAC protocol for cognitive radio networks considering self-interference," *IEEE Access*, vol. 3, pp. 2715–2729, 2015.
[41] L. T. Tan, L. Ying, and D. W. Bliss, "Power control and relay selection in full-duplex cognitive relay networks: Coherent versus non-coherent scenarios," in *Proc. IEEE 51st Annu. Conf. Inf. Sci. Syst.*, 2017, pp. 1–6.
[42] 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.
[43] 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.
[44] S. M. Ross, *Introduction to Probability Models*. San Francisco, CA, USA: Academic, 2014.
[45] Z. Lu, X. Sun, and T. L. Porta, "Cooperative data offload in opportunistic networks: From mobile devices to infrastructure," *IEEE Trans. Netw.*, vol. 25, no. 6, pp. 3382–3395, Dec. 2017.
[46] H. S. Chang, P. J. Fard, S. I. Marcus, and M. Shayman, "Multitime scale Markov decision processes," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 976–987, Jun. 2003.
[47] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial Intell.*, vol. 112, no. 1–2, pp. 181–211, 1999.
[48] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3675–3683.
[49] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Des. Implementation*, 2016, pp. 265–283.
[50] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing framework in vehicle networks: A deep reinforcement learning," Tech. Rep. [Online]. Available: https://www.dropbox.com/s/pqf6gndinqagp1a/TVTSSTechreport.pdf?dl=0



Le Thanh Tan (S'11–M'15) received the B.Eng. and M.Eng. degrees from the Ho Chi Minh University of Technology, Ho Chi Minh City, Vietnam, in 2002 and 2004, respectively, and the Ph.D. degree from Institut National de la Recherche Scientifique, Quebec City, QC, Canada, in 2015. From 2002 to 2010, he was a Lecturer with the Ho Chi Minh University of Technical Education. In 2015, he was a Postdoctoral Research Associate with Ecole Polytechnique de Montreal. From 2016 to 2017, he was a Postdoctoral Research Associate with Arizona State University. He is currently with Department of Electrical and Computer Engineering, Utah State University, Logan, UT, USA. His research interests include artificial intelligence, machine learning, Internet of Things, vehicular networks, 5G wireless communications, edge computing, fog computing and cloud computing, information centric networking, software defined networking, and network function virtualization. He has served on TPCs of different international conferences including IEEE CROWNCOM, VTC, PIMRC, etc.

Rose Qingyang Hu (S'95–M'98–SM'06) received the B.S. degree from the University of Science and Technology of China, Hefei, China, the M.S. degree from New York University, New York, NY, USA, and the Ph.D. degree from the University of Kansas, Lawrence, KS, USA. She is a full Professor of Electrical and Computer Engineering Department, Utah State University, Logan, UT, USA. Besides 10 years academia research experience, she has also more than 10 years R&D experience with Nortel, Blackberry, and Intel as a Technical Manager, a Senior Research Scientist, and a Senior Wireless System Architect, actively participating in industrial 3G/4G technology development, standardization, system level simulation and performance evaluation. She has published extensively and holds more than 30 patents in her research areas. Her current research interests include next-generation wireless communications, wireless network design and optimization, Internet of Things, cloud computing/Fog computing, multimedia QoS/QoE, wireless system modeling and performance analysis. She is the IEEE Communications Society Distinguished Lecturer 2015–2018 and the recipient of the Best Paper Awards from IEEE Globecom 2012, IEEE ICC 2015, IEEE VTC Spring 2016, and IEEE ICC 2016. She served as TPC Co-Chair for IEEE ICC 2018. She is currently serving on the editorial boards for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE COMMUNICATIONS MAGAZINE, the IEEE WIRELESS COMMUNICATIONS MAGAZINE, and the IEEE INTERNET OF THINGS JOURNAL. She is a member of Phi Kappa Phi Society.

Authorized licensed use limited to: China University of Petroleum. Downloaded on December 20,2020 at 19:16:57 UTC from IEEE Xplore. Restrictions apply.