Twin-Timescale Artificial Intelligence Aided Mobility-Aware Edge Caching and Computing in Vehicular Networks

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

***Abstract*—In this paper, we propose a joint communication, caching and computing strategy for achieving cost efficiency in vehicular networks. In particular, the resource allocation policy is specifically designed by considering the vehicle's mobility and the hard service deadline constraint. An artificial intelligence-based multi-timescale framework is proposed for tackling these challenges. To mitigate the complexity associated with this large action and search space in the sophisticated multi-timescale framework considered, we propose to maximize a carefully constructed mobility-aware reward function using the classic particle swarm optimization scheme at the associated large timescale level, while we employ deep reinforcement learning at the small timescale level of our sophisticated twin-timescale solution. Numerical results are presented to illustrate the theoretical findings and to quantify the performance gains attained.**

***Index Terms*—Vehicular networks; vehicular mobility; edge caching and computing; artificial intelligence; deep reinforcement learning; particle swarm optimization**

I. Introduction

The emerging broadband wireless applications have led to an unprecedented tele-traffic escalation in vehicular networks, which were designed for improving safety and fuel-economy, for reducing accidents and traffic congestion in the transportation systems. To tackle these challenges, edge caching and computing capable of offloading tasks to the road side units (RSUs) have been proposed for improving the QoS, despite carrying out intensive computations to a hard deadline [1], [2]. To elaborate a little further, edge computing may be viewed as an alternative to cloud computing, since it moves the communication, control, computation and management functions from the centralized cloud to the edge of a network [3].

Wireless cooperative caching constitutes one of the most widely studied paradigms, where the relay nodes can cooperatively store the multimedia contents [4]–[6]. Hence, a user can download the requested content directly from the nearby relays instead of acquiring it from the 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. Moreover, exploiting the node-mobility for creating efficient caching strategies has recently received substantial research attention [6]–[8]. In [7], Wang *et al.* modeled the node-mobility pattern in terms of the inter-contact time between different users and proposed a mobility-aware caching placement policy for maximizing the data offloading ratio. Poularakis and Tassiulas [8] aimed for minimizing the workload at the macro BS by caching the contents at the small-cell base stations (SBS), when the user mobility is considered.

In this paper, we make a further bold step in designing, analyzing and optimizing the cooperative coded caching placement and computing allocation by considering the constraints of limited dynamic storage capacities and computational resources at the RSUs as well as giving cognizance to the constraints of the vehicle's mobility and hard deadline delay. Specifically, the contributions of this paper can be summarized as follows.

1) We model heterogeneous networks (HetNets) relying on a mobility-aware coded probabilistic caching and computation offloading scheme. The content transmission is deemed successful only if *1) the requested content is received from the nearby RSUs during the vehicular movement; and 2) its corresponding tasks are offloaded to the nearby mobile edge computing (MEC) servers under the delay constraint to be observed*. Otherwise, the requesting vehicle receives the content requested from the BS and offloads the computational tasks to the BS.
2) We formulate the joint optimal caching and computing allocation problem to minimize the system cost under the constraints of dynamically fluctuating limited storage capacities and computational resources at the RSUs as well as under the constraints of vehicular mobility and hard end-to-end deadline delay.
3) To reduce the complexity imposed by the large action space, we develop an algorithm based on the multi-timescale framework of [9] for beneficially configuring the parameters of caching placement and computing resource allocation as well as for determining the sets of potentially connecting RSUs. In particular, we develop efficient mobility-aware algorithms using both particle swarm optimization (PSO) [10] and deep Q-learning [11] for the large timescale and small timescale models, respectively.

Manuscript received June 01, 2018; revised November 23, 2018; accepted January 11, 2019.

L. T. Tan and R. Q. Hu are with the Department of Electrical and Computer Engineering, Utah State University, Logan, Utah 84322-4120, USA. Emails: {tan.le, rose.hu}@usu.edu.

L. Hanzo is with the School of Electronics and Computer Science, University of Southampton, UK. Email: lh@ecs.soton.ac.uk.

4) We present numerical results for illustrating the performance of the proposed algorithms by using the optimal parameter configuration found for caching, for computing and for vehicular mobility. The impact of the user mobility, data size, RSUs' caching storage, backhaul capacities and cloud resources on the system performance is also studied.

The outline of this paper is as follows. In Section II-A, we discuss the related literature, background and potential applications. Section III describes our system model. Section IV briefly presents the proposed framework of artificial intelligent-based mobility-aware edge caching and computing for the resource allocation. Then we describe the large timescale model relying on PSO-based reward maximization in Section V and the small timescale deep Q-learning in Section VI. Section VII presents our performance results followed by our concluding remarks in Section VIII.

II. Related Works, Background and Applications

A. Related Works

Various research problems and solutions have been considered in the edge caching and computing literature. The strategy of caching popular contents at various local devices aims for placing contents closer to mobile users with the assistance of device-to-device (D2D) communications in HetNets, as studied in [12], [13]. In [12], relay nodes with caching capability are introduced to deliver the stored messages cooperatively with the BS, yielding a low delay. In [13], Ji *et al.* considered the combined effect of using both coding in the delivery phase, achieving "coded multicast gain" and spatial reuse as a benefit of local short-range D2D communication. Wireless cooperative caching constitutes one of the most widely studied wireless caching network paradigms, where both the local user terminals and the relay nodes can cooperatively store the multimedia contents [12]–[16].

Small-cell caching, referred to as "Femtocaching" in [14], [15], utilized SBSs in HetNets as distributed caching devices. In [16], Li *et al.* analyzed a deterministic content placement, where the placement strategy was optimized by exploiting both the knowledge of the node locations and of the instantaneous wireless channels. Moreover, Gregori *et al.* [17] investigated the case that both the relays and the user devices can perform caching to improve performances of throughput and delay. Poularakis *et al.* [18] studied the joint design of routing and caching for maximizing the data offloading ratio. Recently, exploiting user-mobility for enhancing the caching placement strategies has received much attention [6], [19]. They then extended their discussions to caching at mobile devices for supporting D2D communication networks [7].

B. Background and Applications

In this section, we first present the state of the art of caching, computing, HetNet and D2D communications in 5G standards and industrial applications [20]. We further introduce the applications for vehicular networks, which have not been adequately addressed in the research community.

1) Background of Fog Computing Network: Compared with 4G/LTE network, 5G network is expected to serve the rapidly growing demands for the thousand-fold growth of mobile data traffic [21]. Mobile network operators and their suppliers are actively developing strategies such as getting more spectrum, seeking advanced technologies, using diverse infrastructures, and offloading traffic to alternative access networks. These technologies aim to increase coverage, boost network capacity and cost-effectively bring contents closer to users. We first introduce the recent literature about the network architecture in the following.

HetNet is the key architecture for the future wireless networks. It can bring the cell sites closer to end users and shorten the radio transmission distance. HetNet is comprised of a variety of radio access technologies with different formats and aspects. Many applications prefer to deploy distributed antennas and small cellular access points (such as femtocells, picocells, metro cells and microcells) in residential homes, subways, enterprises and hot-spot areas. In particular, cell sizes have been progressively shrinking from the order of hundreds of square kilometers to the order of tens of square meters or even less. The small cell can enhance spatial reuse, increase system capacity, extend coverage and offload traffic efficiently. There exist rich research literature in HetNets (e.g., see [22] and references therein), where various aspects including interference management, cell association, stochastic network modeling and energy-efficiency have been investigated. Novel cell association mechanisms and architectures shall be developed, where recent advanced techniques, namely, flexible uplink/downlink communications, massive multiple-input multiple-output (MIMO), D2D communications, full-duplexing and etc., can be utilized [23]. Network modeling approaches based on stochastic geometric tools have recently gained enormous attention in both academia and industry [24].

We now investigate the network protocols of fog computing networks, which are employed in the fog/edge units to provide the communication, computation and storage resources and services to the users. One of the most important research topics in content delivery networks is caching at the edge of the network, which is one of the most promising solutions in 5G wireless networks [6], [12]–[16]. It helps lower down the content-access latency and backhaul traffic loading, improve user Quality of Experience and reduce network cost. The concept of caching originally came from algorithm designs in operating systems, which aims to improve the scalability of world wide web and to offload the network by caching contents in the proxy servers and/or intermediate nodes of the network [25]. Recently, information-centric networking has been proposed as a promising solution for Internet of Things (IoTs) due to its focus on uniquely naming contents and smartly distributing these across the network, rather than on endpoints [26]. Note that normally a few popular contents are most frequently requested by users in spite of the substantial amount of data traffic in the network [27]. It implies that a small portion of popular contents will actually contribute to the majority of data traffic over a period of time. The wireless cooperative caching is received substantial attention, where both local user terminals and relay nodes can cooperatively store the multimedia contents [28]. So a user can download the requested contents directly from the nearby storage unit instead of from the BS. As a result, caching-aided networks have been shown to present a great potential to lower the backhaul load, in turn reducing the end-to-end access delay and increasing the throughput.

2) Potential Applications: The industrial IoT system has recently attracted extensive research attention, because it can connect machines and devices in different vertical segments such as oil and gas, transportation, power generation and health-care [6], [28]–[31]. In particular, a substantial number of devices having Internet connections can collect, analyze and share data. A huge amount of data can be collected or analyzed in cloud servers, which extract meaningful information and provide insights to network operations. Hence, the advanced tools of artificial intelligence and signal processing [6], [29]–[33] would be employed as the excellent solutions to tackle the big data.

Let us consider the scenario of smart city. People drive along the street of the city and are connected to a possibly far-away central cloud for music or YouTube videos. Alternatively, they can get these contents directly from the nearby small BSs and/or from the nearby devices, which cache the corresponding contents [6], [28], [29]. Moreover, the requesting users can also keep the received contents and then serve other nearby users via D2D communications [28]. Another application can be found in the smart monitoring network in the city. The monitoring devices collect road information such as weather, wind direction and intensity, road surface temperatures and conditions, air quality, etc. The measured data are transmitted to the cloud for processing and analyzing [29]. Then, the extracted insights are sent back to the small BSs to serve nearby devices, who request this information [28]–[31].

The smart city concept is gaining ever increasing attention by researchers in various areas. Recent proposals and technological initiatives seek to make cities smarter as well as more sustainable. To embrace the smart city paradigm shift, ones would improve the lives of the people living in the city as well as adapt to a fast-paced environment. Especially, four key elements such as climate management systems, building, transportation and electricity would be carefully addressed by the technological initiatives. The proposed work is ambitious to deal with one of the main elements, i.e. an intelligent transportation system. In particular, there is an observation on the issue of increasing frequency and severity of road accidents and rising traffic congestion due to the increasing number of vehicles, the poor infrastructure and the inefficient traffic controls. We put an effort on developing advanced communications technologies and intelligent data collecting techniques of the vehicular networks to improve safety, enhance efficiency, reduce accidents and decrease traffic congestion in the transportation systems. The proposed cost-effective models guarantee the real-time communication between the vehicles and the RSUs.

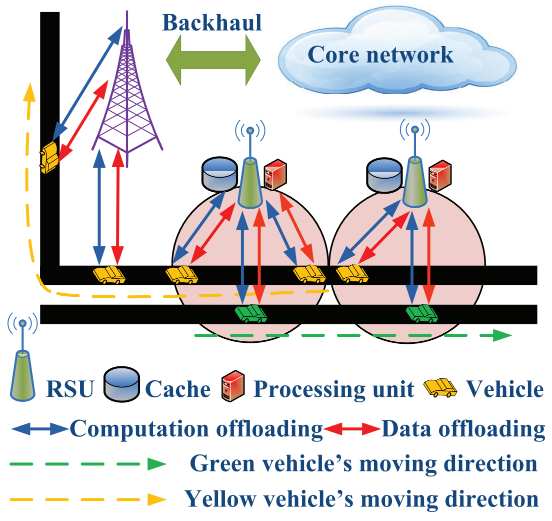

Fig. 1. Mobility-aware network model.

TABLE I
DEFINITION OF PARAMETERS

T_d	Duration of transporting the content c
$\nu_k^i(t)$	Spectral efficiency of link between vehicle i and RSU k
$b_k^i(t)$	Bandwidth assigned to link between vehicle i and RSU k
$r_k^i(t)$	Communication rate of link between vehicle i and RSU k
W_c^i	Task of content c for vehicle i
l_c^w	Size of computational task for content c
l_c	Size of Fountain-coded segment for content c
D_c^w	Number of CPU cycles required for accomplishing task
f_k^i	Computational capability of RSU k allocated to vehicle i
$r_{k,c}^{i,w}(t)$	Computing rate
T_{con}^R, λ	Contact duration and contact frequency
a	Number of segments obtained by vehicle i from RSUs within T_d
a^w	Number of tasks that are computed by the RSUs
s_c	Number of segments required to reconstruct content c
δ^R	Communication cost between vehicles and RSUs
δ^{BS}	Communication cost between vehicles and BS
ξ_c	Caching cost
η^R	Computational cost of MEC server
η^{BS}	Computational cost of BS
S_k^R	Maximum caching storage of RSU

III. System Models

A. Network Architecture

We consider a vehicular network that includes one BS, K RSUs hosting MEC servers and U vehicles. Note that MEC servers are installed at the RSUs for computing, which help reduce the computational load on the BS. Let $\mathcal{K} = \{1, ..., K\}$ and $\mathcal{U} = \{1, ..., U\}$ be the sets of RSUs/MEC servers and vehicles, respectively. We assume that the requesting vehicle is capable of concurrently downloading the requested content and uploading its tasks to the RSUs/BS by employing the radical full-duplex technology [34]–[36]. The network architecture is presented in Fig. 1, where we consider the vehicular mobility, edge caching and computational models. In particular, the yellow vehicle starts to request the contents and their computation from the RSU 1. Then, it moves to the coverage range of RSU 2 and hence, the data offloading and the computation offloading also change from RSU 1 to RSU 2 during its movement. After that this vehicle further moves out of the communication coverages of RSUs 1 and 2. So, it must connect directly to the BS for the data offloading and the computation offloading.

B. Communication Model

The channel between a vehicle and an RSU is time-varying and it is modeled by a finite-state Markov chain (FSMC) [5], [6]. Let γ_k^i denote the receiver SNR of the link between vehicle i and RSU k. We partition and quantize γ_k^i into L discrete levels, each of which corresponds to a state of FSMC. Let T_d be the duration of transporting the content c in terms of the number of time slots. Then, the realization of γ_k^i at t is $\Gamma_k^i(t)$. Let us assume furthermore that OFDMA is used for data communications. All the vehicle-to-RSU links are assigned an orthogonal bandwidth, hence there is no interference from one link to another. Let $\nu_k^i(t)$ denote the spectral efficiency of the link between vehicle i and RSU k. The communication rate of vehicle i can be expressed as $r_k^i(t) = b_k^i(t)\nu_k^i(t), \forall i \in \mathcal{U}$, where $b_k^i(t)$ is the bandwidth assigned to the link.

C. Computing Model

Each task W_c^i of vehicle i consists of two components $\{l_c^w, D_c^w\}$, where l_c^w is the size of the computational task for content c, which includes the software code and the input parameters. Furthermore, D_c^w is the number of CPU cycles required for accomplishing the task. After accomplishing the computational task, RSU k sends the computing results (i.e. the control signals) back to vehicle i. Note that we ignore the transmission duration of the control signals due to the small amount of data. Let f_k^i (CPU cycles/second) be the computational capability of RSU k allocated to vehicle i. Due to the vehicular mobility, multiple vehicles may access the same RSU and share the same MEC server at a given time instant. Hence we do not know exactly the computational capability of vehicle i at the next time instant. Thus, the computational capability f_k^i can be modeled by a random variable. We divide f_k^i into N levels, where N corresponds to the number of available computational capability states. Hence, the computing rate (bits/second) is expressed as $r_{k,c}^{i,w}(t) = \frac{f_k^i(t) l_c^w}{D_c^w}$.

D. Mobility Model and Coded Caching Scheme

This paper uses the contact duration T_{con}^R and contact frequency λ to model the mobility pattern of vehicles. Although the vehicles' positions may change at any time, the moving range of the vehicle is relatively small during a short time. We assume that within each contact time T_{con}^R, the mobile vehicle remains connected to the same RSU. If the vehicles move at a high speed, the contact duration can be short, while it will be longer if the vehicles move at a low speed. Vehicular mobility is typically modeled by discrete random jumps with the corresponding intensity characterized by the average contact duration between jumps [7]. The number of contacts between vehicle i and RSU k can be modeled by the Poisson distribution relying on the parameter λ_k^i, which represents the connection frequency associated with a specific mobility intensity.

We assume that C contents can be requested by the vehicles. The average request rate for content c ($c \in \{1, 2, \ldots, C\}$) at time t can be expressed as $\lambda_c(t) = \beta/(\rho c^\alpha)$. Note that the set $\{1, 2, \ldots, C\}$ is stored 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 request probability can be modeled by the Zipf function of $1/\rho c^\alpha$ [7], where we have $\rho = \sum_{c=1}^{C} 1/c^\alpha$ and α ($0 < \alpha < 1$) is the popularity skew. In a coded caching scheme, each content c is encoded into multiple segments with length l_c by using a rateless Fountain code [7] and a requested content file can then be recovered by collecting any s_c number of encoded segments, which constitute a subset of the complete Fountain-coded file. These segments can be cached in the RSU storage. If a vehicle fails to collect enough encoded segments from the RSUs within the delay-tolerance time T_d, the BS sends the missing data to the vehicle. A Fountain-coded caching scheme is particularly beneficial for vehicles in motion.

Remark 1: For simplicity, we employ the rateless Fountain code of [7] in the coded caching scheme. Although this scheme generally does not work well for a fixed deadline owing to its random delay, it still enhances the system's performance over its un-coded counterpart. Moreover, this simple approach enables us to gain insight into artificial intelligence aided mobility-aware edge caching and computing, while keeping the problem sufficiently tractable. The extension of the model to more advanced coding techniques will be considered in our future works. Relevant coding implementations that were published in some recent contributions [37]–[40] would be useful for these further studies. In particular, the performance is significantly improved, when sophisticated coding and packet-combining are used at the ARQ-aided receiver.

IV. Artificial Intelligence-Based Mobility-Aware Edge Caching and Computing

This section formulates the resource allocation optimization problem considered, where the parameters of caching, computing and communication are optimized jointly. Recall that the network has K RSUs/MEC servers, U vehicles and C content files. The downlink channel conditions, the computational capabilities, the caching states and the vehicles' mobility intensity all change dynamically. We aim for determining the subset of RSUs as well as their resources to serve each requesting vehicle. Due to the large number of system states and actions as well as owing to the system dynamics, it is hard to solve this optimization problem by employing traditional methods. We invoke both deep reinforcement learning [11] and the classic PSO [10], which can efficiently determine a near-optimal action, despite searching through only a small fraction of the potentially excessive total search space at a much reduced complexity.

In this paper, a twin-timescale framework is developed based on [9]. To improve the performance, we consider not only the resource requirements of communications, storage and computing but also the hard deadline for every requested content. So we integrate the resource consumption and the hard deadline into a twin-timescale optimization framework.

The large timescale model is developed at epoch level (corresponding to T_d time slots), while the small timescale model is at the time slot level. Similar to [9], the large timescale model receives the requested content and computes the corresponding tasks within each epoch, while the small timescale model performs actions based on the raw states. In particular, the large timescale model provides resources to guarantee that the vehicle receives a sufficiently high number of Fountain-coded segments and offloads all the tasks within T_d. To do so, the system selects the appropriate sets of potential RSUs eligible for caching as well as the sets of MEC servers suitable for computing. However, the algorithms proposed in [9] do not scale well, when the network size increases. Moreover, the user-mobility imposes excessive operational cost and extensive resource allocation complexity. Hence, new schemes have to be conceived for addressing the associated mobility and scalability challenges. To that end, we propose to use mobility-aware reward estimation for the large timescale model. In particular, we use the PSO [10] for maximizing the estimated reward in the large timescale model. Then, the small time scale model selects the most appropriate actions based on the states for maximizing the immediate reward. The specific differences between the estimated reward in the large scale model and the immediate reward in the small scale model will be elaborated on Sections V and VI, respectively. The algorithm developed in the technical report [41] is used for learning as part of the small timescale models in order to derive the optimal policy. This model is presented in Fig. 2.

Remark 2: In the scenario considered, there are two salient metrics for us to consider, namely the reward and delay constraint. Our proposed twin-timescale framework is designed to address both. By contrast, the large timescale model aims for maximizing the reward of the requesting vehicle under the assumption that it receives a sufficiently high number of Fountain-coded segments and offloads all the tasks within T_d (we term this as a goal). Although not exactly known, the reward is estimated based on a mobility-aware model at the large scale. In the small timescale model, an accurate reward can be calculated, when the agent takes an action. The agent maximizes the reward at every point of action without considering the delay constraint. To deal with this issue, the agent has to check whether the received segments and offloading tasks are completed or not at each time slot.

Remark 3: We focus on the centralized algorithm as the first step in this paper. According to the machine learning perspective, it is not the right time to develop it due to the following reasons. To implement the distributed machine learning system efficiently, the system must satisfy the impossible strict requirements such as *1)* the consistency across all the machines, *2)* the fault tolerance due to the equipment breakdowns in large scale distributed system, *3)* the perfect communication between machine nodes within the computing cluster, *4)* the efficient storage mechanism tailored to different environments, *5)* the good coordination in resource management mechanism for all the joint machine nodes and *6)* the excellent designation of programming interfaces to support multiple programming languages. Therefore, the extension of the model to develop the distributed algorithm will be considered in our future work. Relevant results that were published in some recent works, e.g., in [30]–[33], [42] and references therein, would be useful for these further studies. Of course, these works still suffer the obstacle that requires the good design to support the different interfaces including operating systems, programming languages and libraries. Hence, it is hard to enable training and evaluation of the deep reinforcement learning model across the distributed servers concurrently with high efficiency and low overhead.

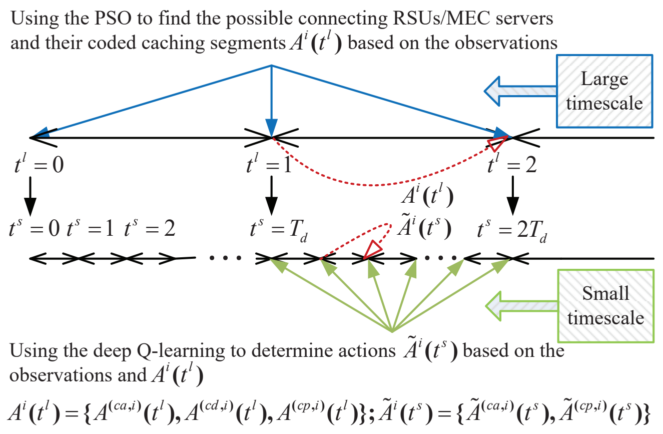
Using the PSO to find the possible connecting RSUs/MEC servers and their coded caching segments $A^i(t^l)$ based on the observations
Large timescale
$t^l=0$
$t^l=1$
$t^l=2$
$t^s=0$ $t^s=1$ $t^s=2$
$t^s=T_d$
$A^i(t^l)$
$\tilde{A}^i(t^s)$
$t^s=2T_d$
Small timescale
Using the deep Q-learning to determine actions $\tilde{A}^i(t^s)$ based on the observations and $A^i(t^l)$
$A^i(t^l)=\{A^{(ca,i)}(t^l),A^{(cd,i)}(t^l),A^{(cp,i)}(t^l)\};\tilde{A}^i(t^s)=\{\tilde{A}^{(ca,i)}(t^s),\tilde{A}^{(cp,i)}(t^s)\}$

Fig. 2. Graphical illustration of multi-timescale model. The large timescale is denoted as t^l, while the small timescale is denoted as t^s.

V. Large Timescale Model with PSO-based Reward Maximization

In this section, we formulate and solve the reward maximization problem for the large timescale model. The best sets of caching RSUs and MEC severs are selected to serve the requesting vehicle. Recall that the requested content has to be received within T_d, i.e. both the computation and content downloading tasks must be completed before the deadline of T_d.

A. Definitions

The system has to appoint the serving RSU for each vehicle, regardless whether or not the requested content should be cached at the RSU. It also has to decide how many coded packets should be cached and whether or not the computational tasks should be offloaded to the RSU/MEC server. An action set $A^i(t^l)$ is defined as $A^i(t^l)=\left\{A^{(ca,i)}(t^l),A^{(cd,i)}(t^l),A^{(cp,i)}(t^l)\right\}$ and discussed in the following.

For the caching action of RSUs, define a $K\times C$ matrix $A^{(ca,i)}(t^l)$, whose (k,c) entry $a^{(ca,i)}_{k,c}(t^l)$ represents the cache control of the c-th content cached at RSU k for vehicle i and $a^{(ca,i)}_{k,c}(t^l)$ is a binary variable with a value of either 1 or 0. Value 0 indicates that the c-th content is not cached at RSU k at t^l or RSU k is out of the communication range of vehicle i during time slot t. By contrast, the value 1 indicates that the c-th content is cached and RSU k is a contact candidate for vehicle i at t^l. At each epoch, $A^{(ca,i)}(t^l)$ determines the subset of RSUs that are contact candidates for vehicle i as well as the subset of contents that are cached at these RSUs.

Let us now define another $K \times C$ action matrix $A^{(cd,i)}(t^l)$, whose (k,c) entry $a_{k,c}^{(cd,i)}(t^l)$ represents the number of coded packets of the c-th content cached at RSU k and the value $a_{k,c}^{(cd,i)}(t^l) \in \{0, s_c\}$. Here, s_c is the minimum number of coded packets that is required for successfully decoding content c. Then, let us define the $(K \times C)$-element computational action matrix of RSUs (MEC servers) as $A^{(cp,i)}(t^l)$, whose (k,c) entry $a_{k,c}^{(cp,i)}(t^l)$ represents the offloading decision of the k-th MEC server for vehicle i, where $a_{k,c}^{(cp,i)}(t^l)$ is either 1 or 0. Value 0 means that the task is not offloaded to MEC server k at time t^l, or RSU k is out of the communication range of vehicle i during time slot t^l, while value 1 indicates that the task is offloaded and RSU k is a contact candidate for vehicle i at time t^l.

B. Reward Function

To maximize the reward, we aim for minimizing the cost of communication, storage and computation. The cost of communication between vehicles and RSUs is defined as δ^R; δ^{BS} is the cost of communications between the vehicles and BS; ξ_c is the caching cost. Finally, η^R and η^{BS} are the computational costs of MEC servers and BS, respectively. The reward for vehicle i is defined as

$$\mathcal{R}^i(t^l) = R^{(cp,i)}(t^l) + R^{(ca,i)}(t^l). \tag{1}$$

Then, $R^{(cp,i)}(t^l)$ and $R^{(ca,i)}(t^l)$ are calculated as follows.

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

Next, $\Pr(U_c^i(\mathbb{X}) = a)$ is calculated, where a is the number of Fountain-coded segments that vehicle i obtains from RSUs. Let M_k^i denote the number of contacts within T_d between vehicle i and RSU k, where M_k^i is a random variable obeying the Poisson distribution with λ_k^i. Then, the total size of the file that vehicle i can obtain from RSU k within T_d is $V_{k,c}^i(\mathbb{X}) = \sum_{n=1}^{M_k^i} \mathcal{B}_k^{i,n}$. Here, $\mathcal{B}_k^{i,n}$ is the maximum number of contents received at vehicle i from RSU k, which follows an exponential distribution with parameter B_k^i. Explicitly, B_k^i can be calculated by $B_k^i = T_{con,k}^R b_k^{i,R} \nu_k^{i,R}$, where $T_{con,k}^R$, $b_k^{i,R}$ and $\nu_k^{i,R}$ are the average duration of every contact, the bandwidth and the achievable spectral 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 represented by $\lambda_k^i T_d$. With this simplification, we arrive at $V_{k,c}^i(\mathbb{X}) = \sum_{n=1}^{\lambda_k^i T_d} \mathcal{B}_k^{i,n}$. Furthermore, $\mathcal{B}_k^{i,n}$ $(n = 1, 2, \ldots, \lambda_k^i T_d)$ is a collection of independent and identically distributed random variables. Thus, according to [43], we have $V_{k,c}^i(\mathbb{X}) \propto \Gamma(\lambda_k^i T_d, B_k^i)$. The size of the file that vehicle i can obtain is expressed as $V_c^i(\mathbb{X}) = \sum_{k\in\mathcal{K}} V_{k,c}^i(\mathbb{X})$.

Let us now proceed to determine the probability distribution function (pdf) of $f_{V_c^i(\mathbb{X})}(v)$. Let $f_{V_{k,c}^i(\mathbb{X})}(v)$ denote the pdf of variable $V_{k,c}^i(\mathbb{X})$. According to the above analysis, the pdf of $f_{V_c^i(\mathbb{X})}(v)$ is the discrete convolution of $f_{V_{k,c}^i(\mathbb{X})}(v)$ $(k = 1, 2, \ldots, K)$ formulated as

$$f_{V_c^i(\mathbb{X})}(v) = f_{V_{1,c}^i(\mathbb{X})}(v) \otimes \ldots \otimes f_{V_{K,c}^i(\mathbb{X})}(v), \tag{2}$$

where $\otimes$ is the discrete convolution. To circumvent the complexity of the convolution, the Welch-Satterthwaite estimation of [43] is used,

$$f_{V_c^i(\mathbb{X})}(v) \approx \frac{v^{\gamma_1 - 1} e^{-v\sigma_1}}{\sigma_1^{-\gamma_1}\Gamma(\gamma_1)}, \tag{3}$$

$$\gamma_1 = \frac{(\sum_{k\in\mathcal{K}} \lambda_k^i T_d B_k^i)^2}{\sum_{k\in\mathcal{K}} \lambda_k^i T_d (B_k^i)^2}, \sigma_1 = \frac{\sum_{k\in\mathcal{K}} \lambda_k^i T_d (B_k^i)^2}{\sum_{k\in\mathcal{K}} \lambda_k^i T_d B_k^i}. \tag{4}$$

Moreover, the random variable $V_c^i(\mathbb{X})$ can be modeled by the Gamma distribution, i.e. $V_c^i(\mathbb{X}) \propto \Gamma(\gamma_1, \sigma_1)$. Note that we have $U_c^i = \min(\lfloor V_c^i / l_c \rfloor, s_c)$, where l_c is the length of each coded segment. Hence, the probability $\mathcal{P}_1(a) = \Pr(U_c^i(\mathbb{X}) = a)$ that vehicle i receives a coded segments is

$$\mathcal{P}_1(a) = \begin{cases} \int_{l_c a}^{l_c(a+1)} f_{V_c^i(\mathbb{X})}(v)dv & 0 \le a < \sum_{k\in\mathcal{K}} x_{k,c}, \\ \int_{l_c a}^{\infty} f_{V_c^i(\mathbb{X})}(v)dv & a = \sum_{k\in\mathcal{K}} x_{k,c}, \\ 0 & otherwise. \end{cases} \tag{5}$$

Thus, $R^{(ca,i)}(t^l)$ is expressed as

$$R^{(ca,i)}(t^l) = -\sum_{c\in\mathcal{C}} \xi_c \sum_{k\in\mathcal{K}} x_{k,c} + (1 - \sum_{a=1}^{s_c} \mathcal{P}_1(a)) s_c \delta^R b^R \nu^R + \sum_{a=1}^{s_c} \mathcal{P}_1(a)(a\delta^R b^R \nu^R - (s_c - a)^+ \delta^{BS} b^{BS} \nu^{BS}), \tag{6}$$

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

2) Calculation of $R^{(cp,i)}(t^l)$: Remind that the computations associated with content c have to be completed within T_d. Let us denote the number of tasks to be completed for content c by s_c^w. For every task w of content c, l_c^w denotes the length of data that includes the software code and the input parameters, while D_c^w represents the number of CPU cycles required for completing task w. The computational capacity of RSU k reserved for vehicle i is f_k^i (cycles per second), which can be translated to the rate $r_{k,c}^{i,w} = f_k^i l_c^w / D_c^w$ (bits per second). At the beginning of each epoch, we reserve a subset of RSUs for vehicle i, denoted by $A^{(cp,i)} = [a_{k,c}^{(cp,i)}]$. Then the computational capacity of K RSUs is $\mathbb{Z}_{K\times C} = [r_{k,c}^{i,w} \times a_{k,c}^{(cp,i)}]$. It may be readily observed that after making the translation of the computational capacity, the scheme of Section V-B1 can be applied for the following derivations.

Let $U_c^i(\mathbb{Z})$ denote the total number of tasks that can be computed by the RSUs within T_d. Next, we have to determine $\Pr(U_c^i(\mathbb{Z}) = a^w)$, where a^w is the number of tasks that are computed by the RSUs. The number of contacts between vehicle i and RSU k within T_d is M_k^i. Hence, the total size of the contents that vehicle i can offload to RSU k can be calculated as $V_{k,c}^i(\mathbb{Z}) = \sum_{n=1}^{M_k^i} \mathcal{H}_k^{i,n}$. Here, $\mathcal{H}_k^{i,n}$ is the maximum number of contents offloaded to RSU k, which is modeled by an exponential distribution with parameter H_k^i calculated as follows. For a single contact having the duration $T_{con,k}^R$, vehicle i has to transmit H_k^i bits to RSU k and RSU k has to accomplish the computation of this amount of data. Hence, we have $T_{con,k}^R = H_k^i / r_k^i + H_k^i / r_{k,c}^{i,w}$, where $r_k^i = b_k^{i,R} \nu_k^{i,R}$ is the communication rate, while $r_{k,c}^{i,w}$ is the computing rate. Then, we arrive at $H_k^i = T_{con,k}^R \left[(r_{k,c}^{i,w})^{-1} + (r_k^i)^{-1} \right]^{-1}$.

Since the number of contacts follows the Poisson distribution, the average number of contacts between vehicle i and RSU k can be represented by $\lambda_k^i T_d$. Through such a simplification, we can obtain $V_{k,c}^i(\mathbb{Z}) = \sum\limits_{n=1}^{\lambda_k^i T_d} \mathcal{H}_k^{i,n}$. Since $\mathcal{H}_k^{i,n}$ $(n = 1, 2, \ldots, \lambda_k^i T_d)$ is a collection of independent and identically distributed random variables, according to [43], we obtain $V_{k,c}^i(\mathbb{Z}) \propto \Gamma(\lambda_k^i T_d, H_k^i)$. Thus, vehicle i can offload a fraction of the file c given by $V_c^i(\mathbb{Z}) = \sum\limits_{k \in \mathcal{K}} V_{k,c}^i(\mathbb{Z})$.

We now determine the pdf of $f_{V_c^i(\mathbb{Z})}(v)$. Let $f_{V_{k,c}^i(\mathbb{Z})}(v)$ be the pdf of variable $V_{k,c}^i(\mathbb{Z})$. According to the above analysis, the pdf of $f_{V_c^i(\mathbb{Z})}(v)$ is the discrete convolution of $f_{V_{k,c}^i(\mathbb{Z})}(v)$ $(k = 1, 2, \ldots, K)$. Then,

$$f_{V_c^i(\mathbb{Z})}(v) = f_{V_{1,c}^i(\mathbb{Z})}(v) \otimes \ldots \otimes f_{V_{K,c}^i(\mathbb{Z})}(v). \tag{7}$$

Note that the random variable $V_c^i(\mathbb{Z})$ obeys the Gamma distribution, i.e. $V_c^i(\mathbb{Z}) \propto \Gamma(\gamma_2, \sigma_2)$. So we have

$$f_{V_c^i(\mathbb{Z})}(v) \approx \frac{v^{\gamma_2 - 1} e^{-v\sigma_2}}{\sigma_2^{-\gamma_2} \Gamma(\gamma_2)}, \tag{8}$$

$$\gamma_2 = \frac{\left(\sum_{k \in \mathcal{K}} \lambda_k^i T_d H_k^i\right)^2}{\sum_{k \in \mathcal{K}} \lambda_k^i T_d (H_k^i)^2}, \tag{9}$$

$$\sigma_2 = \frac{\sum_{k \in \mathcal{K}} \lambda_k^i T_d (H_k^i)^2}{\sum_{k \in \mathcal{K}} \lambda_k^i T_d H_k^i}. \tag{10}$$

Note that $U_c^i = \min(\lfloor V_c^i / l_c^w \rfloor, s_c^w)$, where l_c^w is the length of each task. So the probability $\mathcal{P}_2(a^w) = \Pr(U_c^i(\mathbb{Z}) = a^w)$ that vehicle i offloads a^w tasks is

$$\mathcal{P}_2(a^w) = \begin{cases} \int\limits_{l_c^w a^w}^{l_c^w (a^w+1)} f_{V_c^i(\mathbb{Z})}(v) dv & 0 \le a^w < \frac{T_d}{K} \sum\limits_{k \in \mathcal{K}} z_{k,c}, \\ \int\limits_{l_c^w a^w}^{\infty} f_{V_c^i(\mathbb{X})}(v) dv & a^w = \frac{T_d}{K} \sum\limits_{k \in \mathcal{K}} z_{k,c}, \\ 0 & otherwise. \end{cases} \tag{11}$$

Finally, $R^{(cp,i)}(t^l)$ is expressed as

$$R^{(cp,i)}(t^l) = -\sum_{c \in \mathcal{C}} \sum_{a^w=1}^{s_c^w} \mathcal{P}_2(a^w) \left[a^w \mathcal{C}^R - (s_c^w - a^w)^+ \mathcal{C}^{BS} \right] + \left[1 - \sum_{a^w=1}^{s_c^w} \mathcal{P}_2(a^w) \right] s_c^w \mathcal{C}^R, \tag{12}$$

where $(a^w)^+ = \max(0, a^w)$. $\mathcal{C}^R = \delta^R b^R \nu^R + \eta^R D_c^w e^R$ is the total offloading cost for a single task, including the communication cost $\delta^R b^R \nu^R$ and the computational cost $\eta^R D_c^w e^R$. Furthermore, $\mathcal{C}^{BS} = \delta^{BS} b^{BS} \nu^{BS} + \eta^{BS} D_c^w e^{BS}$ is the total cost of offloading to the BSs for a single task, which includes the communication cost $\delta^{BS} b^{BS} \nu^{BS}$ and the computational cost $\eta^{BS} D_c^w e^{BS}$.

C. Problem Formulation

In this treatise, reward is the key performance metric. We are interested in finding both the optimal caching placement and the resource allocation that maximize the reward. Specifically, the reward $\mathcal{R}^i$ is a function of the caching control of the RSUs $\left(A^{(ca,i)}, A^{(cd,i)}\right)$ and the computing control of the MEC servers $A^{(cp,i)}$. Then, the reward maximization problem can be formulated as follows:

[P1]:

$$\begin{aligned} \max_{A^{(ca,i)}, A^{(cd,i)}, A^{(cp,i)}} \quad & \mathcal{R}^i \\ \text{s.t.} \quad & a_{k,c}^{(ca,i)}, a_{k,c}^{(cp,i)} \in \{0,1\}, a_{k,c}^{(cd,i)} \in [0, s_c], \\ & \textstyle\sum_{c \in [1,\ldots,C]} a_{k,c}^{(cd,i)} \le S_k^R, \end{aligned} \tag{13}$$

where S_k^R is the maximum caching capacity of RSU k. Here the first and second constraints represent the limited cached capacities and computational resources of the RSU. The next section presents our solution to problem **P1**.

D. Caching Placement and Computing Resource Allocation

Problem **P1** in (13) is non-convex, since the reward in the objective function defined in (1) is a complex non-linear function. Given this observation, we have devised Alg. 1 for finding the solution of this optimization problem based on the PSO [10]. We normalize $A^{(cd,i)}$ to arrive at $\breve{A}^{(cd,i)}$, where $\breve{a}_{k,c}^{(cd,i)} = a_{k,c}^{(cd,i)} / s_c$. Let us reshape the $K \times C$ matrices $A^{(ca,i)}$, $\breve{A}^{(cd,i)}$ and $A^{(cp,i)}$ to the vectors $\mathbf{X}_1$, $\mathbf{X}_2$ and $\mathbf{X}_3$, respectively. The variable vector is defined as $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3\}$, where the length of vector $\mathbf{X}$ is $\breve{K} = 3K \times C$. Then, we aim for finding the solutions of $\mathbf{X}$, where $X_i \in [0,1]$ for maximizing the reward $\mathcal{R}^i(\mathbf{X})$. Note that the solutions $\mathbf{X}^*$ can be translated to the optimal solutions of $\left\{A^{(ca,i)*}, A^{(cd,i)*}, A^{(cp,i)*}\right\}$.

For brevity, we describe the PSO in the context of our problem as follows. Explicitly, the PSO is a swarm intelligence based technique inspired by the collective behavior of social swarms of bees or birds. Here, each single solution defined as a "particle" may be viewed as a "bird" in the search space. A swarm of these particles moves through the search space at a specified velocity in order to find an optimal position. Generally, each particle, which is a member of the population, has two components including the position of $\mathbf{X}$ and the velocity of $\mathbf{V}$. Note that the velocities decide the movement directions of the particles, when they roam around the search space to find the optimal position. In our particular problem, the position vector of particle j is presented as $\mathbf{X}_j = \left[X_{j1}, X_{j2}, \ldots, X_{j\breve{K}}\right]$ and its velocity vector is represented as $\mathbf{V}_j = \left[V_{j1}, V_{j2}, \ldots, V_{j\breve{K}}\right]$. Here, j denotes particle j, while $\breve{K}$ represents the number of unknown variables given above.

Firstly, a group of random particles (solutions) is used for initializing the PSO and then the optimal solution is sought by updating the consecutive generations as follows. At every iteration, there are two "best" values: *1)* the particle's best known position $\mathbf{p}^{best}$ is the position vector of the best solution (fitness) of this particle achieved so far (we term it as the personal best position); *2)* the swarm's best known position $\mathbf{g}^{best}$ is the position vector of the global "best" solution, which is tracked by the particle swarm optimizer (we term it as the global best position). In particular, the fitness value of each particle is compared to that of its corresponding $\mathbf{p}^{best}$. If the fitness value of the j-th particle is larger than that of $\mathbf{p}_j^{best}$, $\mathbf{p}_j^{best}$ is then replaced by the j-th particle. Next, $\mathbf{g}^{best}$ is selected as the best $\mathbf{p}_j^{best}$ among all the particles. Recall that the velocities of the particles represent the rate of change for the current position and all particles continuously move in the search space.

In summary, each particle j at iteration t can be defined by three values: 1) the position, X_{jk}^t, is used to evaluate the quality of particle j; 2) the velocity, V_{jk}^t, represents the direction and magnitude per iteration of the particle j's movement; and 3) the personal best position, $p_{jk}^{best,t}$, is the particle j that has been visited up to the iteration t. Note that the personal best position represents the knowledge of the quality solutions. The superscript t and the subscript k represent the iteration index and the particle index, respectively. So the velocity of particle j at iteration $t+1$ is updated according to the current position, X_{jk}^t, the current velocity, V_{jk}^t, the personal best position, $p_{jk}^{best,t}$ and the global best position, $g_{jk}^{best,t}$. We summarize this procedure in Alg. 1.

So both the position and velocity of the particle j are updated as follows:

$$V_{jk}^{t+1} = \psi V_{jk}^t + c_1\epsilon_1\left(p_{jk}^{best,t} - X_{jk}^t\right) + c_2\epsilon_2\left(g_{jk}^{best,t} - X_{jk}^t\right), \quad (14)$$

$$X_{jk}^{t+1} = X_{jk}^t + V_{jk}^t, \quad (15)$$

where c_1 and c_2 are the cognitive and social learning factors (also called acceleration coefficients), which are positive constants. These parameters are among the most important parameters and are utilized for accelerating the algorithm. In fact, they control the balance between exploration and exploration tendencies, which can be explained as follows [44], [45]. If c_1 is set to the larger value, the particle tends to move closer to the personal best position. If c_2 is set to the higher value, it will result in faster convergence to the global best position. It implies that enables the particle to move toward to the optimum solution. In our scenario, we set c_1 and c_2 as $c_1 = c_2 = 2.05$ [45]. Still referring to (14), ϵ_1 and ϵ_2 are a pair of independent random variables with uniform distribution between 0 and 1, which are generated at every update for each individual dimension. These parameters are used for ensuring diversity of the group particles. Furthermore, ψ is the inertia weight, which shows the effect of the previous velocity vector on the new vector. Usually, ψ is set as the constant value in the range of $[0.9, 1.2]$. However, ψ in our algorithm decreases linearly from 0.9 to 0.4 to enhance the convergence [46]. When the termination conditions are satisfied, the final $\mathbf{g}^{best}$ will be the output as the optimal solution, $\mathbf{X}^*$.

Note that the velocity vector in the regular PSO may grow to infinity if the values of ψ, c_1 and c_2 are not set correctly. In this case, the particle leaves the search space (we call the swarm explosion) and hence, its objective value is not considered for updating personal and global best vectors [44], [45]. Therefore, the fundamental solution to the swarm explosion restricts the velocity to the range of $[-v_{max}, v_{max}]$, where v_{max} is the maximum allowed velocity [44]. However, this restriction may prevent the particles from going to the boundaries of the search space. Hence, the search scheme does not direct to the optimal solutions. To improve the stability and ensure the convergence of PSO, we employ the constriction factor [44]. In particular, the velocity of the constriction factor based PSO can be expressed as follows:

$$V_{jk}^{t+1} = \overline{\psi}\left[V_{jk}^t + c_1\epsilon_1\left(p_{jk}^{best,t} - X_{jk}^t\right) + c_2\epsilon_2\left(g_{jk}^{best,t} - X_{jk}^t\right)\right], \quad (16)$$

$$\overline{\psi} = \frac{2}{\left|2 - \xi - \sqrt{\xi^2 - 4\xi}\right|}, \quad (17)$$

where $\xi = c_1 + c_2$ and $\xi > 4$. The stability and convergence requirements would be satisfied by controlling the value of ξ. When ξ increases, the constriction factor $\overline{\psi}$ decreases. This results in reducing the diversity of the population and hence, it takes longer to reach convergence. Moreover, the value of ξ has to be larger than 4 to guarantee stability. In this paper, we set $c_1 = c_2 = 2.05$, $\xi = 4.1$.

VI. Small Timescale Deep Q-Learning Model

In the small timescale model, the operation of the control system is as follows. First, the control system collects the status of each RSU/MEC server, of each vehicle and of each vehicle's mobility. Next, it constructs the system states, which are the vehicle's mobility and communication channel information as well as the caching contents and the computational resources at the RSU/MEC servers. Then, the deep Q-network (or agent) receives the system states and determines the optimal action A^*. This action includes the set of RSUs as well as their caching and computational resources for the requesting vehicle. Finally, the control system receives this action and forwards it to the vehicle.

A. Deep Q-Learning

In the following, we provide a brief description of deep Q-networks. Interested readers can find further detailed information in [11]. 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

Algorithm 1 PSO-BASED EDGE CACHING AND COMPUTING IN VEHICULAR NETWORKS

```
1: for each particle $j = 1, 2, \ldots, D$ do
2:   Initialize the position $\mathbf{X}_j$ with a random number $\in [0,1]$, the velocity $V_{ji}$ with a random number $\in [-v\max, v\max]$ and the particle's best known position $\mathbf{p}_j^{best} = \mathbf{X}_j$.
3:   if $\mathcal{R}^i\left(\mathbf{p}_j^{best}\right) > \mathcal{R}^i\left(\mathbf{g}_j^{best}\right)$ then
4:     Update the swarm's best known position $\mathbf{g}_j^{\text{best}} = \mathbf{p}_j^{best}$.
5:   end if
6: end for
7: repeat
8:   for each particle $j = 1, 2, \ldots, D$ do
9:     for $d = 1, 2, \ldots, N$ do
10:      Pick random numbers, $\epsilon_1$ and $\epsilon_2 \in [0,1]$. Update the velocity using Eq. (14).
11:    end for
12:    Update the particle's position $\mathbf{X}_j = \mathbf{X}_j + \mathbf{V}_j$.
13:    if $\mathcal{R}^i\left(\mathbf{p}_j^{best}\right) < \mathcal{R}^i\left(\mathbf{X}_j\right)$ then
14:      Update the particle's best known position $\mathbf{p}_j^{best} = \mathbf{X}_j$.
15:      if $\mathcal{R}^i\left(\mathbf{g}_j^{best}\right) < \mathcal{R}^i\left(\mathbf{p}_j^{best}\right)$ then
16:        Update the swarm's best known position $\mathbf{g}_j^{best} = \mathbf{p}_j^{best}$.
17:      end if
18:    end if
19:    $\mathbf{g}^{best} = \underset{\mathbf{g}_j^{best}}{\operatorname{argmax}} \mathcal{R}^i\left(\mathbf{g}_j^{best}\right)$.
20:  end for
21: until Convergence
22: $\mathbf{X}^* = \mathbf{g}^{best}$.
```

takes an action $a(t) \in A$ based on the current state $x(t) \in X$. After that the system evolves to a new state $x(t+1) \in X$ with the transition probability $P_{x(t)x(t+1)}(a)$ and obtains the immediate reward $r\left[x(t), a(t)\right]$.

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

$$V^{\pi}(x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \epsilon^t r(x(t), a(t))\,|x(0) = x\right], \qquad (18)$$

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

Recall that the environment is modeled by a Markov decision process (MDP), hence, the value function can be rewritten as

$$V^{\pi}(x) = R\left[x, \pi(x)\right] + \epsilon \sum_{x' \in X} P_{xx'}\left[\pi(x)\right] V^{\pi}(x'), \qquad (19)$$

where $R(x, \pi(x))$ is the mean value of the immediate reward $r\left[x, \pi(x)\right]$, while $P_{xx'}\left[\pi(x)\right]$ is the transition probability from x to x', when the action $\pi(x)$ is executed.

Let us now consider model-free reinforcement learning, where both R and P are unknown. Q-learning is one of the strategies capable of determining the best policy π^*. From (19), a state-action function, namely, Q-function is defined as

$$Q^{\pi}(x, a) = R(x, a) + \epsilon \sum_{x' \in X} P_{xx'}(a) V^{\pi}(x'), \qquad (20)$$

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

$$Q^{\pi^*}(x, a) = R(x, a) + \epsilon \sum_{x' \in X} P_{xx'}(a) V^{\pi^*}(x'), \qquad (21)$$

The discounted cumulative state function can be written as

$$V^{\pi^*}(x) = \max_{a \in A} Q^{\pi^*}(x, a). \qquad (22)$$

So we now aim for estimating the best Q-function instead of finding the best policy. The Q-function can be obtained by using the recursive method of [47].

$$Q_{t+1}(x,a) = Q_t(x,a) + \alpha\Big(r + \epsilon \max_{a'} Q_t(x', a') - Q_t(x,a)\Big), \qquad (23)$$

where α is the learning rate. Note that $Q_t(x,a)$ definitely converges to $Q^*(x,a)$, when an appropriate α is selected [11]. One way to estimate the Q-function is to use a function approximation, such as a neural network $Q(x,a;\theta) \approx Q^*(x,a)$, where the parameter θ is the weight of the neural network [11], which is adjusted at each iteration during the Q network training in order to reduce the mean square error.

To make reinforcement learning applicable to real applications, deep Q-learning was developed as follows, which requires two improvements for transforming 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, $D(t) = \{e(1), ..., e(t)\}$. In contrast to traditional Q-learning, where the arriving samples are used for training the neural network's parameters, deep Q-learning randomly selects the sample 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

$$Loss(\theta) = \mathbb{E}\left[\left(y - Q(x, a, \theta)^2\right)\right], \qquad (24)$$

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

Given the action A^i at the large timescale model, we perform the actions $\tilde{A}^{(cp,i)}(t)$ and $\tilde{A}^{(ca,i)}(t)$ according to the states in the time slot level. The vehicle's mobility has an impact on the actions of $\tilde{A}^{(cp,i)}(t)$ and $\tilde{A}^{(ca,i)}(t)$ as well as on the immediate reward. 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 for balancing the exploration and exploitation. It implies that this policy balances the reward maximization based on the knowledge already acquired, while attempting new actions to further increase knowledge [11]. So the remaining tasks define the system states, system actions and immediate reward functions, which are presented in the following sections.

B. System States

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

C. System Actions

In each time slot, the agent has to decide which RSU is assigned to the vehicle and whether or not the computational task should be offloaded to the RSU/MEC server. Note that the caching placement is performed as part of the large timescale model. However, the size of each caching content dynamically changes in the small timescale model [5], [6]. The current action $\tilde{A}^i(t^s)$ is denoted by

$$\tilde{A}^i(t^s) = \left\{\tilde{A}^{(cp,i)}(t^s), \tilde{A}^{(ca,i)}(t^s)\right\}, \tag{25}$$

where $\tilde{A}^{(cp,i)}(t^s)$ and $\tilde{A}^{(ca,i)}(t^s)$ are defined as follows.

For the communication control, let us define the $K \times C$ matrix $\tilde{A}^{(ca,i)}(t^s)$, whose (k, c) entry $\tilde{a}_{k,c}^{(ca,i)}(t^s)$ represents the connection control of RSU k and vehicle i for the c-th content cache, where $\tilde{a}_{k,c}^{(ca,i)}(t^s)$ is a binary variable with a value of either 1 or 0. Value 0 implies that the content c is not cached at RSU k at time slot t^s or that RSU k is out of the communication range of vehicle i during time slot t^s, while value 1 means that the content is cached and RSU k is the best contact for vehicle i during time slot t^s. For the computing control, let us define the $K\times C$ matrix $\tilde{A}^{(cp,i)}(t^s)$, whose (k, c) entry $\tilde{a}_{k,c}^{(cp,i)}(t^s)$ represents the connection between the MEC server k and vehicle i for the data offloading. Each element $\tilde{a}_{k,c}^{(cp,i)}(t^s)$ is either 1 or 0. Value 0 means that the task is not offloaded to the MEC server k at time slot t^s, or RSU k is out of the communication range of vehicle i during time slot t^s, while value 1 indicates that the task is offloaded and RSU k is the best contact with vehicle i during time slot t^s.

D. Reward Functions

The exact immediate rewards for the actions are defined as follows:

$$\tilde{\mathcal{R}}^i(t^s) = \tilde{R}^{(cp,i)}(t^s) + \tilde{R}^{(ca,i)}(t^s). \tag{26}$$

$\tilde{R}^{(cp,i)}(t^s)$ and $\tilde{R}^{(ca,i)}(t^s)$ can be respectively calculated as

$$\begin{aligned}\tilde{R}^{(ca,i)}(t^s) = &-\sum_{c\in\mathcal{C}}\xi_c \sum_{k\in\mathcal{K}} x_{k,c} + \sum_{k\in\mathcal{K}} \tilde{a}_{k,c}^{(ca,i)} \delta^R b_k^{i,R} \nu_k^{i,R} \\ &+ (1 - \sum_{k\in\mathcal{K}} \tilde{a}_{k,c}^{(ca,i)}) \delta^{BS} b^{BS} \nu^{BS}, \end{aligned}\tag{27}$$

$$\tilde{R}^{(cp,i)}(t^s) = -\sum_{c\in\mathcal{C}} \sum_{k\in\mathcal{K}} \tilde{a}_{k,c}^{(cp,i)} \tilde{\mathcal{C}}^R + (1 - \sum_{k\in\mathcal{K}} \tilde{a}_{k,c}^{(cp,i)}) \mathcal{C}^{BS}, \tag{28}$$

where $\tilde{\mathcal{C}}^R = \delta^R b_k^{i,R} \nu_k^{i,R} + \eta^R D_c^w e^R$ is the cost of offloading tasks to RSU, including the communication cost $\delta^R b_k^{i,R} \nu_k^{i,R}$ and the computational cost $\eta^R D_c^w e^R$. Similarly, $\mathcal{C}^{BS} = \delta^{BS} b^{BS} \nu^{BS} + \eta^{BS} D_c^w e^{BS}$ is the cost of offloading the tasks to BS, where $\delta^{BS} b^{BS} \nu^{BS}$ and $\eta^{BS} D_c^w e^{BS}$ are the communication and computational costs. Here, e^R and e^{BS} are the energy consumption per cycle of the MEC server and the BS, respectively.

VII. Numerical Results

This section presents our numerical results for illustrating the performance of the proposed scheme. The key parameters are chosen as follows, unless stated otherwise: K = 10; U = 50; $b^R = 4b^{BS}$ = 1 MHz; ξ_c = 2 units/MB; $\delta^{BS} = 10\delta^R$ = 20 units/MHz; $\eta^{BS} = 10\eta^R$ = 100 units/J; $e^{BS} = e^R$ = 1W/GHz; D_c^w = 100 Mcycles; l_c/l_c^w = 20; f_k^i in the range of [10, 20] GHz; time slot is set 20s, $T_d = 240$s. Note that the cost of using the services of the BS (for downloading cached contents and computing their corresponding tasks) is assumed to be ten times the cost of using the services of the RSUs.

The wireless channels of vehicle-to-RSU links all follow the Markov model. During the contact duration between vehicle i to RSU k, ν_k^i has two states, i.e. $\nu_k^i = \{1, 4\}$, with 1 corresponding to the worst channel and 4 to the best channel. The probabilities of remaining in the same state and that of traversing from one state to the other are set to 0.7 and 0.3, respectively. Similarly, the probabilities of caching are set to 0.7 and 0.3. The computational states of MEC servers are assumed to obey 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 detailed results have to be omitted here owing to the space limit, but motivated readers can find the details in [41].

Fig. 3 shows the success probability versus the coded packet size l_c for $\lambda T_d = 0.1$. Here, "*success*" means that the tagged vehicle i completes downloading the requested content and offloads its corresponding tasks for computation within the hard deadline T_d without assistance from the BS. We compare our proposed scheme against two benchmark cases: *1) random resource allocation; and 2) equal resource allocation*. Similar to our proposed scheme, both the random resource allocation and the equal resource allocation select the sets of eligible RSUs at every epoch in order to reduce the number of reserved RSUs for reducing the cost of their caching storage. However, they pre-select these sets in a random manner. In every time slot, the random resource allocation allocates caching contents and computational resources, while the equal resource allocation performs caching and

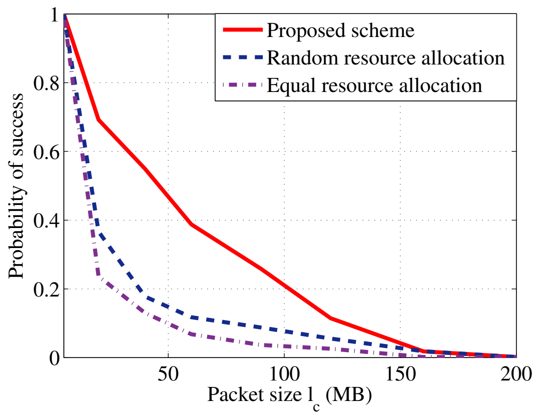

Fig. 3. Probability of success vs coded packet size l_c for $\lambda T_d = 0.1$.

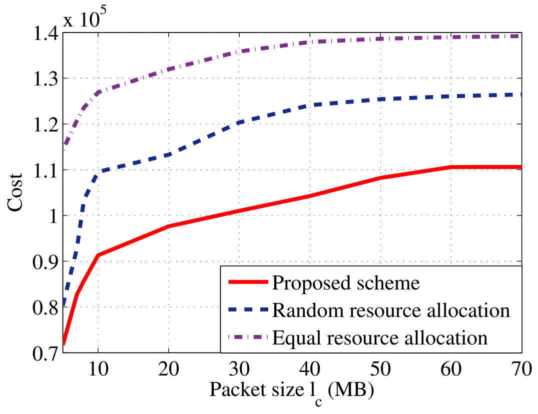

Fig. 4. Cost vs coded packet size l_c for $\lambda T_d = 0.05$.

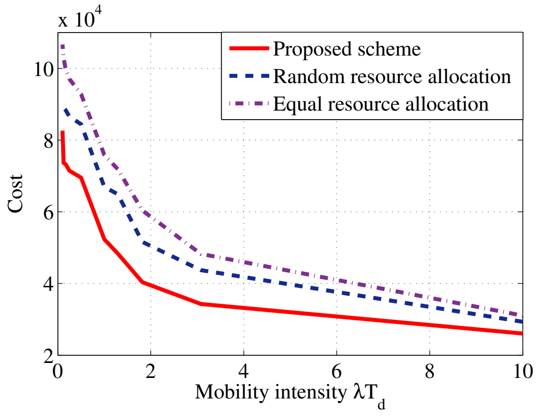

Fig. 5. Cost vs vehicle mobility intensity, λT_d for $l_c = 30MB$.

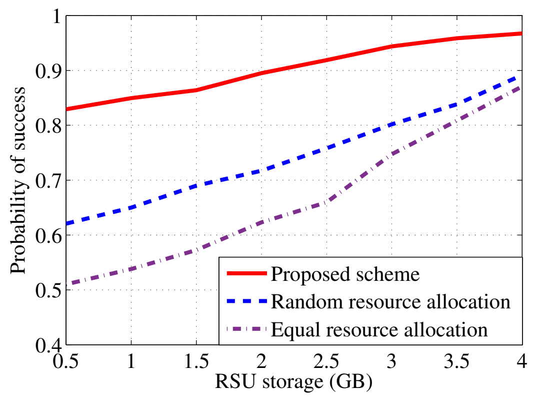

Fig. 6. Probability of success vs RSU storage, λT_d for $l_c = 30MB$.

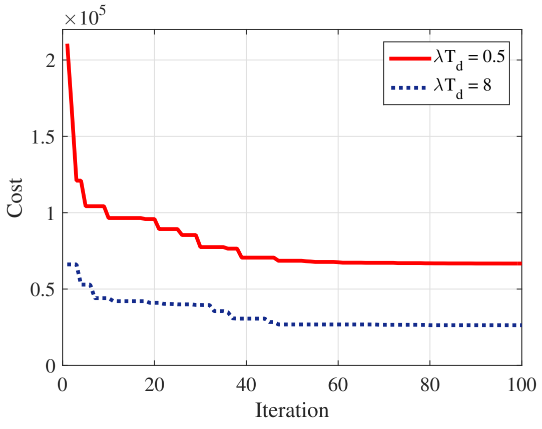

Fig. 7. Convergence performance of the PSO algorithm (Alg. 1) for $\lambda T_d = (0.5, 8)$ and $l_c = 30MB$.

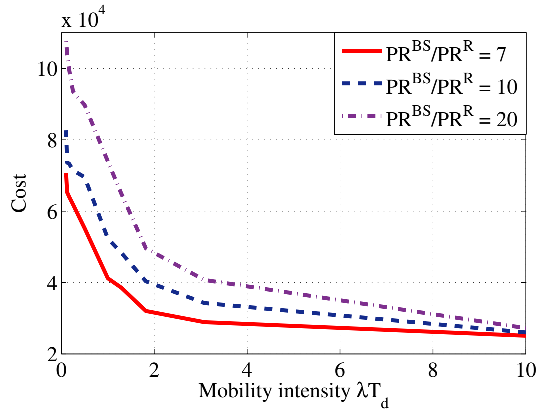

Fig. 8. Cost vs vehicle mobility intensity, λT_d, the backhaul capacities and cloud computing resources for $l_c = 30MB$.

computing allocation equally. As expected, the probability of success decreases upon increasing the coded packet size for all the schemes, because it becomes easier to transmit data within the contact duration, when the size of the coded segment is small. 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 for every epoch. Therefore, the traditional method of pre-selecting eligible RSUs does not work well in the presence of hard end-to-end deadline delay constraint and in the scenario of vehicular mobility. Note that 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 the number of coded packets required for successfully processing the requested content. As a result, it fails to accomplish its corresponding tasks within the hard deadline of T_d. Our proposed small timescale model provides the excellent performances compared with the two benchmark schemes because our proposal can adapt well with the dynamic change of parameters for communications, storage and computing as well as vehicular mobility.

Fig. 4 illustrates the cost versus Fountain-coded packet size l_c for $\lambda T_d = 0.05$. We can readily observe that increasing the size of the coded segment increases the cost of the caching storage. Moreover, it is hard for the tagged vehicle i to download the content and to offload its computational tasks under the constraints of vehicular mobility and hard deadline delay, when the coded file size is large. Therefore, the tagged vehicle may fail to receive the required number of coded segments. As a remedy, these missed segments and their corresponding tasks can then be downloaded and computed with the assistance of the BS, but say at ten times higher cost. Again, our proposed scheme consistently provides significant performance gains over the other two schemes. Similarly, the impact of mobility intensity on the cost is demonstrated in Fig. 5 with $l_c = 30MB$. When the vehicular mobility is low, the cost is high because there are less opportunities for the requesting vehicle to connect to new RSUs. Hence, every RSU has to cache more segments in order to acquire the minimum number of received coded segments required for successful decoding. However, the proposed scheme relying on deep reinforcement learning, PSO and our twin-timescale framework mitigates this detrimental impact.

In Fig. 6, we investigate the influence of RSUs' caching storage, S_k^R, on our proposed scheme. In particular, the relationship between caching storage of RSUs and the probability of success is illustrated. For simplicity, we consider the same caching storage of RSUs, i.e. $S_k^R = S^R$. Again, we compare our proposed scheme with the random resource allocation and the equal resource allocation. As expected, the probability of success increases, when the caching storage of RSUs increases. This observation can be intuitively explained as follows. When the RSUs' caching storage is larger, the RSUs can store more contents and hence, the higher probability of success would be attained. Furthermore, it is clearly observed from the figure that the influence on probability of success of the random resource allocation is greater than that of the equal resource allocation, especially, when the cache storage of RSUs is larger. Because the diversity of contents that can be stored with the random resource allocation is higher than that with the equal resource allocation, when storage capacity of RSUs is larger. Therefore, the random resource allocation achieves a higher probability of success than the equal resource allocation. Finally, our proposed scheme outperforms both the random resource allocation and the equal resource allocation. Especially, when the RSU has a limited resource of caching capacity, our proposed scheme also achieves the high performance in terms of probability of success (all the probabilities of success are higher than 80%). This result validates the significant contributions of the classic PSO in the large timescale model and deep reinforcement learning in the small timescale model. In particular, we beneficially exploit the user-mobility and caching cooperation amongst the RSUs to achieve a high probability of success.

We also evaluate the complexity of our proposed twin-timescale framework. In Fig. 7, we illustrate the convergence performance of the large timescale model for $\lambda T_d = (0.5, 8)$ and $l_c = 30MB$. It is readily observed that the cost performance decreases with the increase of the iteration number. It reaches a relatively stable value at the iteration of 48. This result confirms that our proposed large timescale model is efficient and effective. Furthermore, the performance cost of the small timescale model reaches the relatively stable value, when the number of the episodes goes around 1100. This number is quite small, because it is beneficial from the solution of the large timescale model (i.e. the decision of sets of necessary RSUs for every epoch).

Finally, we study the impact of backhaul capacity and of the amount of cloud computing resources on the cost. For low backhaul capacities and low cloud computing resources, the performance cost of using the limited bandwidth to download cached content from the BS is high and the cloud resources invoked for computing its corresponding tasks are expensive. Hence, the system imposes a cost penalty on the requesting vehicle, whenever it relies on the limited backhaul capacities and cloud computing resources. Again, we consider the scenario of $l_c = 30MB$. For simplicity, we assume that the ratio of the communication costs equals the ratio of computing cost (i.e. $PR^{BS}/PR^R = \eta^{BS}/\eta^R = \delta^{BS}/\delta^R$). To elaborate, PR^{BS}/PR^R is the ratio of the cost of using the BS's resources to the cost of using the RSUs' resources. In Fig. 8, it is set to 7, 10 and 20, which correspond to low, medium and high backhaul and computational capabilities. When PR^{BS}/PR^R increases, the cost increases because poor backhaul and BS computational capabilities degrade the system performance. It is also observed that when mobility intensity λT_d is large at high speed, the performance curves follow a downward trend and the discrepancy between the curves of low and high capabilities becomes small. By contrast, there is a substantial gap between the curves of low and high capabilities at a low mobility intensity, λT_d. These observations indicate that our proposed scheme mitigates the deleterious effects of limited backhaul capacity and low BS computational resources by effectively exploiting the distributed storage and computational capabilities as well as the vehicular mobility.

VIII. Conclusions

In this paper we developed a framework of joint optimal resource allocation for communication, caching and computing in vehicular networks. In order to strike a compelling trade-off between a high QoS and desirable cost efficiency, the vehicular mobility was exploited for enhancing both the caching and computing policies. We formulated an optimization problem for resource allocation and proposed a twin-timescale framework for solving this problem. Due to the potentially excessive complexity of the resultant large action space, it remains an open challenge to operate within a truly multi-timescale framework. Thus, we proposed PSO-based reward maximization for our large timescale model, while we invoked deep reinforcement learning for the small timescale model. Our numerical results provided insights into a number of important theoretical findings and showed significant performance gains as a benefit of using optimal parameter configurations for our proposed scheme.

Acknowledgment

The research of L. T. Tan and R. Q. Hu was supported in part by National Science Foundation under grants NeTS 1423348 and EARS 1547312, in part by Natural Science Foundation of China under grant 61728104, and in part by Intel Corporation. L. Hanzo gratefully acknowledges the financial support of the EPSRC projects EP/Noo4558/1, EP/PO34284/1, of the Royal Society's GRCF as well as of the European Research Council's Advanced Fellow Grant QuantCom.

References

[1] Y. Chen, L. Wang, Y. Ai, B. Jiao and L. Hanzo, "Performance analysis of NOMA-SM in vehicle-to-vehicle massive MIMO channels," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 12, pp. 2653-2666, Dec. 2017.

[2] J. Wang, C. Jiang, Z. Han, Y. Ren and L. Hanzo, "Internet of vehicles: Sensing aided transportation information collection and diffusion," *IEEE Trans. Veh. Tech.*, vol. 67, no. 5, pp. 3813-3825, May 2018.

[3] 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.

[4] J. Hu, L. L. Yang and L. Hanzo, "Energy-efficient cross-layer design of wireless mesh networks for content sharing in online social networks," in *IEEE Trans. Veh. Tech.*, vol. 66, no. 9, pp. 8495-8509, Sept. 2017.

[5] Y. He, N. Zhao and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Tech.*, vol. 67, no. 1, pp. 44-55, Jan. 2018.

[6] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing framework in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Tech.*, vol. 67, no. 11, pp. 10190-10203, Nov. 2018.

[7] R. Wang, J. Zhang, S. H. Song and K. B. Letaief, "Mobility-aware caching in D2D networks," in *IEEE Trans. Commun.*, vol. 16, no. 8, pp. 5001-5015, May 2017.

[8] 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, 2017.

[9] H. S. Chang, P. J. Fard, S. I. Marcus and M. Shayman, "Multitime scale Markov decision processes," in *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 976-987, June 2003.

[10] J. Kennedy, "Particle swarm optimization," *Encyclopedia of machine learning*, Springer US, pp. 760-766, 2011.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski and S. Petersen, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529-533, Feb. 2015.

[12] N. Golrezaei, A. F. Molisch, A. G. Dimakis and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," in *IEEE Commun. Mag.*, vol. 51, no. 4, pp. 142-149, April 2013.

[13] 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.

[14] Y. Chen, M. Ding, J. Li, Z. Lin, G. Mao and L. Hanzo, "Probabilistic small-cell caching: Performance analysis and optimization," in *IEEE Trans. Veh. Tech.*, vol. 66, no. 5, pp. 4341-4354, May 2017.

[15] 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," in *IEEE J. Sel. Areas Commun*, vol. 34, no. 8, pp. 2115-2129, Aug. 2016.

[16] J. Li, Y. Chen, Z. Lin, W. Chen, B. Vucetic and L. Hanzo, "Distributed caching for data dissemination in the downlink of heterogeneous networks," in *IEEE Trans. Commun.*, vol. 63, no. 10, pp. 3553-3568, Oct. 2015.

[17] M. Gregori, J. Gomez-Vilardebo, J. Matamoros and D. Gunduz, "Wireless content caching for small cell and D2D networks," in *IEEE J. Sel. Areas Commun*, vol. 34, no. 5, pp. 1222-1234, May 2016.

[18] K. Poularakis, G. Iosifidis and L. Tassiulas, "Approximation algorithms for mobile data caching in small cell networks," in *IEEE Trans. Commun*, vol. 62, no. 10, pp. 3665-3677, Oct. 2014.

[19] 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.

[20] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. C. Chen and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Commun.*, vol. 24, no. 2, pp. 98-105, April 2017.

[21] E. Bastug, M. Bennis and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82-89, Aug. 2014.

[22] J. G. Andrews, H. Claussen, M. Dohler, S. Rangan and M. C. Reed, "Femtocells: Past, present, and future," in *IEEE J. Sel. Areas Commun.*, vol. 30, no. 3, pp. 497-508, April 2012.

[23] R. Q. Hu, 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.

[24] J. G. Andrews, F. Baccelli and R. K. Ganti, "A tractable approach to coverage and rate in cellular networks," in *IEEE Trans. Commun.*, vol. 59, no. 11, pp. 3122-3134, Nov. 2011.

[25] J. Wang, "A survey of web caching schemes for the internet," *ACM SIGCOMM Comp. Commun. Review*, vol. 29, no. 5, pp. 36-46, Oct. 1999.

[26] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher and B. Ohlman, "A survey of information-centric networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26-36, Jul. 2012.

[27] M. Cha, H. Kwak, P. Rodriguez, Y. Y. Ahn and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," in *IEEE/ACM Trans. Networking*, vol. 17, no. 5, pp. 1357-1370, Oct. 2009.

[28] L. T. Tan, R. Q. Hu and Y. Qian, "D2D communications in heterogeneous networks with full-duplex relays and edge caching," in *Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4557-4567, Oct. 2018.

[29] Q. Wang, L. T. Tan, R. Q. Hu and G. Wu, "Hierarchical collaborative cloud and fog computing in IoT networks," *Proc. WCSP 2018.*

[30] L. T. Tan and L. B. Le, "Compressed sensing based data processing and MAC protocol design for smartgrids," *Proc. IEEE WCNC2015.*

[31] L. T. Tan and L. B. Le, "Joint data compression and MAC protocol design for smartgrids with renewable energy," *Wirel. Commun. Mob. Com.*, vol.16, no. 16, pp. 2590-2604, July 2016.

[32] L. T. Tan and H. Y. Kong, "A novel and efficient mixed-signal compressed sensing for wide-band cognitive radio," *Proc. IEEE IFOST2010.*

[33] L. T. Tan, H. Y. Kong and V. N. Q. Bao, "Projected Barzilai-Borwein methods applied to distributed compressive spectrum sensing," *Proc. IEEE DySPAN2010.*

[34] Z. Zhang, K. Long, A. V. Vasilakos and L. Hanzo, "Full-duplex wireless communications: challenges, solutions, and future research directions," *Proc. IEEE*, vol. 104, no. 7, pp. 1369-1409, Jul. 2016.

[35] L. T. Tan and L. B. Le, "Design and optimal configuration of full-duplex MAC protocol for cognitive radio networks considering self-interference," in *IEEE Access*, vol. 3, pp. 2715-2729, 2015.

[36] L. T. Tan and L. B. Le, "Multi-channel MAC protocol for full-duplex cognitive radio networks with optimized access control and load balancing," *Proc. IEEE ICC2016.*

[37] H. Chen, R. G. Maunder and L. Hanzo,"A survey and tutorial on low-complexity turbo coding techniques and a holistic hybrid ARQ design example," in *IEEE Commun. Surv. Tutor.*, vol. 15, no. 4, pp. 1546-1566, Fourth Quarter 2013.

[38] H. Chen, R. G. Maunder and L. Hanzo,"Low-complexity multiple-component turbo-decoding-aided hybrid ARQ," in *IEEE Trans. Veh. Tech.*, vol. 60, no. 4, pp. 1571-1577, May 2011.

[39] H. Chen, R. G. Maunder and L. Hanzo,"Lookup-table-based deferred-iteration aided low-complexity turbo hybrid ARQ," in *IEEE Trans. Veh. Tech.*, vol. 60, no. 7, pp. 3045-3053, Sept. 2011.

[40] H. Chen, R. G. Maunder, Y. Ma, R. Tafazolli and L. Hanzo, "Hybrid-ARQ-aided short fountain codes designed for block-fading channels," in *IEEE Trans. Veh. Tech.*, vol. 64, no. 12, pp. 5701-5712, Dec. 2015.

[41] "Twin-timescale artificial intelligence aided mobility-aware edge caching and computing in vehicular networks," technical report. Online: https://www.dropbox.com/s/x3w1wxbkessh10p/MDRLTechrep.pdf?dl=0

[42] E. P. Xing, Q. Ho,W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar and Y. Yu, "Petuum: A new platform for distributed machine learning on big data," *IEEE Trans. Big Data*, vol. 1, no. 2, pp. 49-67, June 2015.

[43] S. M. Ross, Introduction to Probability Models. San Francisco, CA, USA: Academic, 2014.

[44] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58-73, Feb. 2002.

[45] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: A review," in *Evol. Comput.*, vol. 25, no. 1, pp. 1-54, March 2017.

[46] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," *Proc. CEC2000.*

[47] S. Gu, T. Lillicrap, I. Sutskever and S. Levine, "Continuous deep Q-learning with model-based acceleration," *Proc. ICML'2016.*

Le Thanh Tan (S'11–M'15) received his B.Eng. and M.Eng. degrees from Ho Chi Minh University of Technology, Vietnam, in 2002 and 2004, respectively, and PhD degree from Institut National de la Recherche Scientifique, Canada in 2015. He is currently with Department of Electrical & Computer Engineering, Utah State University. From 2002 to 2010, he was a Lecturer with the Ho Chi Minh University of Technology and Education. In 2015, he was a Postdoctoral Research Associate at Ecole Polytechnique de Montreal, Canada. From 2016 to 2017, he was a Postdoctoral Research Associate at Arizona State University, USA. His research interests include artificial intelligence, machine learning, Internet of Things, vehicular networks, 5G wireless communications, edge/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. He is a Member of the IEEE.

Rose Qingyang Hu (S'95-M'98-SM'06) received the B.S. degree from the University of Science and Technology of China, the M.S. degree from New York University, and the Ph.D. degree from the University of Kansas. She is a currently a Professor with the Electrical and Computer Engineering Department and an Associate Dean for Research with the College of Engineering, Utah State University. Besides a decade academia experience, she has more than 10 years of R&D experience with Nortel, Blackberry, and Intel as a Technical Manager, a Senior Wireless System Architect, and a Senior Research Scientist, actively participating in industrial 3G/4G technology development, standardization, system level simulation, and performance evaluation. Her current research interests include next-generation wireless communications, wireless system design and optimization, Internet of Things, cloud computing/fog computing, wireless system modeling, and performance analysis. She has published extensively in top IEEE journals and conferences and holds numerous patents in her research areas. She is an IEEE Communications Society Distinguished Lecturer Class 2015-2018. She was a recipient of Best Paper Awards from the IEEE GLOBECOM 2012, the IEEE ICC 2015, the IEEE VTC Spring 2016, and the IEEE ICC 2016. She served as the TPC Co-Chair for the 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 and the IEEE Wireless Communications.

Lajos Hanzo (http://www-mobile.ecs.soton.ac.uk) FREng, FIEEE, FIET, Fellow of EURASIP, DSc received his degree in electronics in 1976 and his doctorate in 1983. In 2009 he was awarded an honorary doctorate by the Technical University of Budapest and in 2015 by the University of Edinburgh. In 2016 he was admitted to the Hungarian Academy of Science. During his 40-year career in telecommunications he has held various research and academic posts in Hungary, Germany and the UK. Since 1986 he has been with the School of Electronics and Computer Science, University of Southampton, UK, where he holds the chair in telecommunications. He has successfully supervised 112 PhD students, co-authored 18 John Wiley/IEEE Press books on mobile radio communications totaling in excess of 10 000 pages, published 1800+ research contributions at IEEE Xplore, acted both as TPC and General Chair of IEEE conferences, presented keynote lectures and has been awarded a number of distinctions. Currently he is directing an academic research team, working on a range of research projects in the field of wireless multimedia communications sponsored by industry, the Engineering and Physical Sciences Research Council (EPSRC) UK, the European Research Council's Advanced Fellow Grant and the Royal Society's Wolfson Research Merit Award. He is an enthusiastic supporter of industrial and academic liaison and he offers a range of industrial courses. He is also a Governor of the IEEE ComSoc and VTS. During 2008 - 2012 he was the Editor-in-Chief of the IEEE Press and a Chaired Professor also at Tsinghua University, Beijing. For further information on research in progress and associated publications please refer to http://www-mobile.ecs.soton.ac.uk.