Multi-Timescale Actor-Critic Learning for Computing Resource Management With Semi-Markov Renewal Process Mobility

Tan Le, *Member, IEEE*, Martin Reisslein, *Fellow, IEEE*, and Sachin Shetty, *Senior Member, IEEE*

***Abstract*— This paper studies artificial intelligence (AI) aided communication and computing resource allocation in a vehicular network that supports blockchain-enabled video streaming. Our study aims to improve the operating efficiency and to maximize the transcoding rewards for blockchain based vehicular networks. Our resource allocation policy considers the vehicular mobility, which is modelled with a highly-realistic Semi-Markov renewal process, as well as the real-time video service delay constraints. We propose a multi-timescale actor-critic-reinforcement learning framework to tackle these grand challenges. We also develop a prediction model for the vehicular mobility by using analysis and classical machine learning, which alleviates the heavy signaling and computation overheads due to the vehicular movement. A mobility-aware reward estimation for the large timescale model is then proposed to mitigate the complexity due to the large action space. Finally, numerical results are presented to illustrate the developed theoretical findings in this paper and the significant performance gains due to our proposed multi-timescale framework.**

***Index Terms*— Deep reinforcement learning, edge computing, user-mobility, vehicular network.**

I. Introduction

VIDEO streaming has been found very useful in vehicular networks for safety and control purposes. Vehicular networks must employ advanced communications technologies and data collecting/processing techniques to enhance the capacity and QoS for video streaming applications [1]. Supporting video streaming among all types of local devices requires effective transcoders. In particular, transcoding is required whenever the users request video chunks with different quality levels or characteristics (such as different bitrates, resolutions, aspect ratios and encoder mechanisms) depending on their current communication bandwidths and computational capabilities.

Moreover, the information and communications technology industries are under tremendous pressure to seek new architectures and new deployment models for the next generation wireless networks, which consist of massive sets of intelligent devices and new applications [2], [3], [4], [5], [6]. Existing cloud computing technology, despite its tremendous benefits, still faces problems, such as latency, fronthaul/backhaul bandwidth limitations, and centralized processing. Recently, mobile edge computing has been proposed to efficiently improve the QoS for applications with intensive computations and hard deadlines [2], [3], [4], [5], [7]. In particular, edge computing is an alternative to cloud computing, which moves the storage and computation from the centralized cloud to the edge of the network. Edge computing can alleviate the challenges and overcome the disadvantages of cloud computing by locating edge servers in close proximity of the end users. Furthermore, the emerging blockchain technologies provide promising solutions to overcome the challenges of low latency, heavy burden of centralized communication, computation, and storage, as well as the corresponding resource allocations [8]. Based on advanced edge computing and blockchain technologies, some providers on the internet have designed decentralized video streaming systems, e.g., [9]. In this paper, we exploit these advanced supporting technologies, including blockchain mechanisms [8], [10], [11], [12] in vehicular networks, to design an edge computing resource management framework.

A. Contributions

This paper focuses on designing, analyzing, and optimizing the communication and computing resource allocations in a vehicular network that supports blockchain-enabled video streaming by considering the constraints of vehicular mobility and hard delay deadlines. Importantly, in contrast to preceding studies, which considered simplistic mobility models that poorly reflect the user mobility in practical vehicular networks, we consider the highly realistic Semi-Markov renewal process mobility model. Specifically, in the context of the Semi-Markov renewal process mobility model, the contributions of this paper can be summarized as follows.

1) We develop mobility-aware transcoder selection for blockchain networks by modeling mobility with Semi-Markov processes. We also propose classical artificial intelligent classifiers [13], [14], [15], namely a generative learning model, to enhance the prediction performance of the users' future locations.

2) We formulate the joint optimal communication and computational resource allocation problem to minimize the transcoding cost under the constraints of limited and dynamic computation resources at the vehicles, as well as the constraints of the vehicular mobility and the delay budget.

Manuscript received 15 September 2022; revised 24 March 2023; accepted 7 August 2023. Date of publication 22 August 2023; date of current version 17 January 2024. This work was supported in part by the DoD Center of Excellence in AI and Machine Learning (CoE-AIML) under Contract W911NF-20-2-0277 with the U.S. Army Research Laboratory, and in part by the National Science Foundation under Grant CNS 2219742. The Associate Editor for this article was C. Long. *(Corresponding author: Martin Reisslein.)*

Tan Le was with the Old Dominion University, Suffolk, VA 23435 USA. He is now with the Department of Electrical and Computer Engineering, Hampton University, Hampton, VA 23669 USA (e-mail: tan.le@hamptonu.edu).

Martin Reisslein is with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: reisslein@asu.edu).

Sachin Shetty is with the Virginia Modeling, Analysis and Simulation Center, Old Dominion University, Suffolk, VA 23435 USA (e-mail: sshetty@odu.edu).

Digital Object Identifier 10.1109/TITS.2023.3303953

Moreover, we incorporate the main parameters of blockchain networks, such as the cumulative stakes and the reputation values.

3) We develop a multi-timescale framework based on actor-critic-learning (ACL) [16] to allocate the communication and computation resources as well as to determine the set of possible connecting vehicles for every Road Side Unit (RSU), the set of video chunk (segment) transcoding tasks assigned to each RSU, and the selected transcoders. In particular, we propose a mobility-aware reward estimation for the large timescale model to reduce the complexity due to the large action space. Our framework predicts the set of possible vehicles in the coverage range of each RSU and determines the sets of transcoding tasks assigned for each RSU. In the small timescale model, the system takes actions to determine the transcoders, the set of transcoding tasks assigned to each transcoder, as well as the resource allocation for both communication and computation.

4) Numerical results are presented to illustrate the performance of the proposed multi-timescale framework by using the optimal parameter configuration for communication, computation, vehicular mobility, and transcoder selection.

The paper is structured as follows. The following subsections give the related literature and the applications. Section II describes the system models. Section III formulates the resource allocation optimization problem based on the actor-critic reinforcement learning algorithm. The large timescale actor-critic-learning is presented in Section IV, while the small timescale actor-critic-learning is provided in Section V. Section VI gives our performance results, followed by the concluding remarks in Section VII.

B. Related Studies

Video streaming is gaining ever-increasing attention, relying on computing at both the central cloud servers and the fog/edge nodes [6]. Lately, blockchain technologies and smart contracts have become key entities in a variety of applications due to their powerful capabilities in interoperability, privacy, security, reliability, and scalability [11]. Although the existing approaches can in principle be used for video streaming in the vehicular networks, they do not handle the vehicular mobility well, as mobility has not previously been explicitly incorporated into video transcoding resource management. We contribute to filling this gap in the literature by explicitly incorporating mobility into our model.

Although user-mobility has been widely studied in wireless networks, especially in vehicular networks, the related studies on resource management commonly employ simplistic mobility models. For instance, Wang et al. [17] modeled the node-mobility pattern in terms of the inter-contact time between different users. Similarly, the studies [4], [5], [7] considered a simplistic random jump process, which is a relatively poor mobility model for vehicular network applications that tends to degrade the performance of vehicular network applications. To address this significant shortcoming of the mobility modeling, this study considers the practically relevant Semi-Markov renewal process to realistically model the user-mobility. Furthermore, we propose a multi-timescale actor-critic reinforcement learning that predicts the realistic mobility more accurately than traditional reinforcement learning.

C. Potential Applications

The proposed blockchain-enabled video streaming can be applied in a smart city, where transportation safety and real-time navigation services are critical features [6], [18], [19], [20]. In particular, the roads are usually equipped with many monitoring devices, which collect environmental conditions, such as wind, humidity, road surface, air quality and traffic. Furthermore, these surveillance systems also record the scenes of nearby accidents, heavy traffic jams, and/or dangerous situations. All of these special events must be provided to the drivers in an accurate and timely manner. Also, to make autonomous vehicular driving operate properly, large amounts of data (captured video) can be transmitted to powerful cloud/edge servers for further processing and analysis. Then, salient information is extracted and is forwarded to the demanding vehicles [7]. Thereby, the servers must transcode the video data to different versions that are used to serve different target devices.

Another application is the on-board infotainment service, which may, for instance, offer multiplayer games, multimedia applications, video-conference, and video streams of sport games for the vehicle passengers. Importantly, the video based infotainment services are provided in a mobile vehicular network. To provide these video based infotainment services, the network infrastructure should facilitate fast mobile edge computing so that the moving vehicles can conveniently obtain their required videos in a timely manner. There are many challenges in the implementation of achieving both the video based infotainment services for mobile vehicles and the transportation safety.

II. System Models

A. Network Architecture

We consider a blockchain network that includes one base station (BS), K Road Side Units (RSUs), and U vehicles (users). Computational processing units are installed at both the RSUs and the vehicles to reduce the workload at the BS. Let $\mathcal{K} = \{1, \ldots, K\}$ and $\mathcal{U} = \{1, \ldots, U\}$ be the sets of the RSUs, or the Multi-access Edge Computing (MEC) servers, and the vehicles, respectively. The original video is divided into multiple time-sliced chunks (segments) of video, and these original video segments are broadcasted ahead of time (offline) via the RSUs to the users [10]. Usually, these original video segments need to be transcoded to new video segments to achieve a high streaming quality (e.g., by adapting the aspect ratio and resolution to match the display device) [10], [21].

We note that the recently emerged marketplaces for computing infrastructures as well as edge computing, which brings the computing resources into close proximity to the user devices, can readily support the mining and the transcoding. For example, the Livepeer[1] open marketplace can easily absorb the low- to moderate-complexity blockchain mining for determining the selected transcoders, making the energy cost

[1] https://livepeer.org/

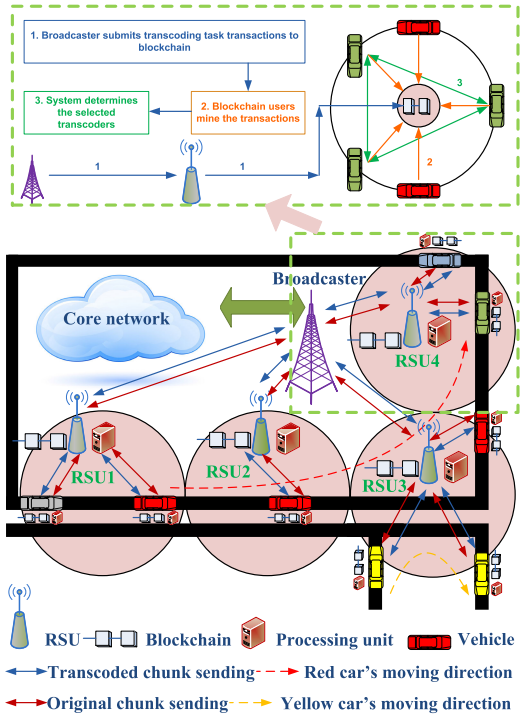

Fig. 1. Mobility-aware blockchain-enabled transcoding model: The top part illustrates the operation of the blockchain network architecture, which includes three steps: submitting transcoding task transactions to blockchain, hashing, and transcoding. The bottom part illustrates one example case of vehicular mobility, in which the tagged red car moves from one wireless coverage area to another. Therefore, the communication link changes during its movement, e.g., the red vehicle successively connects to RSU1, RSU2, and RSU3 as it travels.

of mining negligible relative to the energy cost for the highly demanding video transcoding. In particular, blockchain miners (e.g., cryptocurrency miners) can rent out idle capacities on their GPU mining infrastructures for the video transcoding. The low- to moderate-complexity blockchain mining can run side-by-side with the highly demanding video transcoding because the chips in graphic cards for video encoding (NVDEC+NVENC) are separate from the chips used for general-purpose computing and cryptomining (CUDA cores). Thus, the blockchain mining computing energy expenditures are negligible compared to the energy expenditures for the video transcoding, as also reflect by current monetary costs which are commonly two orders of magnitude higher for video transcoding compared to cryptocurrency hashing.

We closely consider the impact of vehicular mobility on the system performance. Let us consider an example of the mobility-aware blockchain-enabled transcoding model in the bottom part of Fig. 1. The red vehicle first moves from RSU 1 to RSU 2. Therefore, the RSU must carefully assign the amount of chunks to this vehicle to make sure that it can complete its tasks before entering the coverage of RSU 2.

B. Communication Model

We now present the communication model for the transmission of the transcoded video chunks. We investigate the system behavior for both communication and computation in every epoch. We define the epoch as the duration of the processing and communication reserved for one video chunk, i.e., the delay-tolerance time. Each epoch can be divided into T_{out} time slots. The time-varying channel parameters of the user-to-RSU links are modeled as finite-state Markov chains (FSMCs) [5], [7]. We denote γ_u^k for the received SNR of the link between user u and RSU k. We employ the Markov chain to model the γ_u^k, which is partitioned and quantized into L discrete levels (each of which corresponds to a state of the Markov chain). The realization of γ_u^k at time slot t is denoted as $\Gamma_u^k(t)$. The channel state transition probability matrix for the link of user u-RSU k, with h_s and g_s representing two states, can be written as

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

whereby $\psi_{g_s h_s}(t) = \Pr\left(\Gamma_u^k(t+1) = h_s \,\middle|\, \Gamma_u^k(t) = g_s\right)$. We assign the OFDM based orthogonal bandwidth b_u^k to the link between user u and RSU k. Hence, there is no interference from one link to another. Let $\nu_u^k(t)$ denote the spectral efficiency of the link between user u and RSU k. The communication rate of vehicle u can be expressed as

$$R_u^k(t) = b_u^k(t)\nu_u^k(t), \forall u \in \mathcal{U}. \tag{2}$$

C. Blockchain-Enabled Transcoding Model

We now introduce the video chunk transcoding task and the computation needed to process the video transcoding and we present the transcoding evaluation mechanism based on a blockchain [8], [10], [11], [12], [21]. A video chunk transcoding task of length L_O [bytes] consists of the program code and the input parameters that govern (specify) the transcoding processing for the video chunk. The other important parameter for every video chunk transcoding task is the required workload/intensity r for the computing node. Hence, the computing node requires rL_O CPU cycles to accomplish the task processing. Let f_u^k (CPU cycles per second) be the computation capability allocated to user u. The computational capability f_u^k can be modeled as a random variable. We divide f_u^k into M levels, where M corresponds to the number of available computation capability states. The realization of f_u^k is $F_u^k(t)$ at time slot t and the transition probability matrix is $\Theta_u^k(t) = \left[\theta_{x_s y_s}(t)\right]_{M\times M}$, where $\theta_{x_s y_s}(t) = \Pr\left(F_u^k(t+1) = y_s \,\middle|\, F_u^k(t) = x_s\right)$, while x_s and y_s are the states. The computing rate (bits computed per second) is expressed as $R_u^{w,k}(t) = f_u^k(t)/r$. The transcoded video with the length of L_T bytes is sent back to the RSU.

We proceed to describe the detailed blockchain-enabled video transcoding model as illustrated in Fig. 1, top. The BS plays the role of the broadcaster, while the RSUs assist to relay information between the users and the BS. Furthermore, the RSUs and the BS cooperatively determine the resource allocation, transcoder selection, and chunk assignment. We now present the operation step-by-step. Firstly, the broadcaster publishes the specifications of the video segments of the original video stream needed for transcoding and submits the corresponding transcoding task transactions to the blockchain network. All the vehicles with available computing resources attend the hashing (as indicated by the red lines in the top part of Fig. 1), i.e., they mine these transactions on the blockchain and we refer to this step as the mining/hashing step. The system determines the set of blockchain users, that will accomplish the transcoding tasks with a blockchain-enabled

transcoder evaluation mechanism that relies on two factors, namely the cumulative stake and the reputation value, which are updated periodically. The cumulative stake is defined as the trustful factor of the candidate's transcoding capability [12]. The higher a candidate's cumulative stake, the more important is the candidate in the transcoding ecosystem. The cumulative stake includes the candidate's own stake and the amount that is delegated from others. The reputation value is important for the transcoding selection; in particular, the reputation value represents the characteristics of seniority, successful transcoding history, and trustworthiness. After the hashing step, the system finds the set of selected transcoders (which are highlighted by the green lines in the top part of Fig. 1) to complete the requested jobs/tasks. Finally, the selected users perform the transcoding, and send the results back to the RSUs.

D. Mobility Model

The vehicular network has a set of RSUs $\mathcal{K} = \{1, \ldots, K\}$, where K is the number of RSUs. We employ the Semi-Markov renewal process, $(X_n, T_n) : n \geq 0$, to model the vehicular mobility. Let $C = 1, 2, \ldots, K$ denote the discrete state space, T_n denote the time of n-th transition, and X_n for $t \in [T_n; T_{n+1})$ denote the system state after the n-th transition for the Semi-Markov renewal process. The state of the Semi-Markov renewal process represents the coverage of each RSU. Hence, the state transition is the handover from the current coverage of one RSU to the coverage of its neighbor RSU. We define the probability of transition from RSU coverage i to the coverage of its neighbor j as $p_{i,j}$. It is assumed that there are a finite number of time homogenous distinct values in the process. Consider user u that has stayed at the i-th RSU coverage for a duration of t. The probability of transition from RSU coverage i to the coverage of its neighbor j is

$$\begin{aligned}\Phi(u)_{i,j}(t) &= \Pr\left(X^u_{n+1} = j, T^u_{n+1} - T^u_n \leq t \,\middle|\, X^u_n = i\right) \\ &= p^u_{i,j} S^u_{i,j}(t), \end{aligned} \tag{3}$$

where $p^u_{i,j}$ is the probability of handover from the coverage of RSU i to that of RSU j for user u. Moreover, $S^u_{i,j}(t)$ is the sojourn time distribution of user u, where its current coverage and its next coverage are RSU i and RSU j, respectively. These quantities $p^u_{i,j}$ and $S^u_{i,j}(t)$ can be calculated as

$$\begin{aligned} p^u_{i,j} &= \lim_{t\to\infty} \Phi(u)_{i,j}(t) \\ &= \Pr\left(X^u_{n+1} = j \,\middle|\, X^u_n = i\right), p^u_{i,j} \in \mathbf{P}^u, \end{aligned} \tag{4}$$

$$S^u_{i,j}(t) = \Pr\left(T^u_{n+1} - T^u_n \leq t \,\middle|\, X^u_{n+1} = j, X^u_n = i\right). \tag{5}$$

III. Blockchain-ACL Empowered Video Streaming

In this section, we introduce the ACL based *multi-timescale* framework for solving the joint communication and computation resource allocation problem. In the multi-timescale ACL framework, the large timescale model estimates the user-mobility and then performs the transcoder selection and resource allocation for both communication and data processing; thus reducing the burden on the small timescale model by reducing the spaces of actions and states.

A. System Operations

The ACL is used to formulate the resource allocation optimization problem, where the parameters of computing and communication are optimized jointly. Recall that the network has K RSUs, U vehicles, and N video chunks. The downlink channel conditions, the computation capabilities, and vehicular mobility intensity all change dynamically. One needs to determine the subsets of transcoding tasks assigned to the RSUs and their nearby vehicles, as well as to allocate their resources to serve the requested transcoding based on the current states. Due to the dynamic changes as well as the large space of the system states and actions, it is very hard to solve this optimization problem by employing traditional methods. Instead, we exploit ACL, which can effectively and efficiently determine the optimal action with a large amount of input data. The system states that consist of the vehicular mobility, communication channel information, and the computational resources at the vehicles are firstly constricted. The actor-critic-network (or agent) determines the optimal action $\mathcal{A}^*$ based on the current system states. This action includes the sets of transcoding tasks assigned to the RSUs and the vehicles, as well as the communication and computing resources allocated for every selected vehicle.

B. Multi-Timescale ACL Framework

There are many recent studies, e.g., [22] and [23], which propose methods to deal with large-scale networks and big data with dynamic variations. A promising candidate is using multi-timescale models, which can improve the system performance as well as make the implementation closely approach the practical application. Based on [22], [23], and [24], we propose a multi-timescale system framework, whereby the large timescale actor-critic-learning model is operated at the epoch level (corresponding to T_{out} time slots) and the small timescale actor-critic-learning model is operated at the time slot level. The large-timescale model is used to range the values of network parameters, while the small-timescale model determines the optimal solution. This implies that the large-timescale model roughly determines the solutions, i.e., these solutions may not be optimal; and the small-timescale model needs to correct and refine the rough solutions.

The proposed algorithms in [5], [7], [22], and [23] work efficiently for the small-size networks. However, they generally do not scale well for large networks with massive numbers of RSUs and vehicles. The resource allocation can be complicated in a mobility scenario. In this study, the action space is very large and causes a high complexity in the implementation. To realize a low-complexity algorithm, we propose a mobility-aware reward estimation for the large-timescale model. In order to obtain the optimal policy, we use the ACL model for learning in both the large-timescale and small-timescale models. The reward is estimated when we take the large-timescale model based action, while the actual reward is realized, when we take the small-timescale model based action.

IV. Large Timescale ACL Model

This section presents the details of the large-timescale transcoder selection framework, which jointly selects the

transcoders and allocates the resources based on the ACL process. There are two phases in this framework: 1) The offline Deep Neural Network (DNN) construction phase, which approximates the action-value function with the corresponding states and actions; and 2) the online dynamic actor-critic-learning phase, which selects the action, controls the system, and updates the dynamic network.

A. Actor-Critic-Based Learning Process

The BS is the agent that identifies the state space, action space, and reward function of the actor-critic-learning-based framework as follows.

1) System State: For RSU k, the system states include the workload of the transcoding job (the total number of video chunk transcoding tasks N), the QoS requirement (timeout T_{out}), the characteristics of the users $u \in \mathcal{U}$ (the cumulative stake CS_u^k, reputation value RV_u^k, available computational resource F_u^k), the number of available sub-channels B^k, and the user candidate set Ω_c^k.

2) System Action: When the transcoding tasks (jobs) are broadcasted, the system agent determines the number of transcoding tasks assigned to every RSU k. This action is denoted by $\mathbf{A}^{(Cv,k)}(t) = \left\{a^{(Cv,k)}\right\}$, where $a^{(Cv,k)} = N^k \in [1, N]$; whereby, we employ the notation convention that superscripts in parentheses specify the parameters of an action, whereas the indices of other parameters, e.g., N are written as superscripts without parentheses. For RSU k, the action $\mathbf{A}^{(U,k)}(t) = \left\{a_u^{(U,k)}\right\}$ is to select the transcoders, where $a_u^{(U,k)} = 1$ if the user u is selected as the transcoder; otherwise, $a_u^{(U,k)} = 0$. So RSU k determines the transcoder set Ω_s^k, which is formulated from the users in the candidate set Ω_c^k, i.e., $\Omega_s^k = \left\{u \,\middle|\, a_u^{(U,k)} = 1, u \in \Omega_c^k\right\}$. Moreover, action $\mathbf{A}^{(B,k)}(t) = \left\{a_u^{(B,k)}\right\}$ allocates bandwidth for every transcoder, where $a_u^{(B,k)} \in \left[1, B^k\right]$. Finally, action $\mathbf{A}^{(Cp,k)}(t) = \left\{a_u^{(Cp,k)}\right\}$ allocates computing resources for every selected transcoder, where $a_u^{(Cp,k)} \in \left[1, F^k\right]$.

3) Reward Function: We aim for maximizing the transcoding reward under the consideration of the deadline constraints of the video chunks by designing effective mechanisms for the transcoder selection and the resource allocation. In the following, we derive an immediate reward that is obtained by the blockchain users when the transcoding tasks are completed successfully. In particular, the reward function consists of the reward due to the completion of tasks and the energy consumption costs. The reward due to the completion of tasks can include two parts, namely the reward from the stakeholder (or equivalently the block reward), which is proportional to the cumulative stake, and the reward from the earning, which is proportional to the reputation value. The cost is caused by the energy usage for both the computation and the communication.

Specifically, the indicator I_k represents the compliance with the time budget constraint at the RSU k. Note that the large timescale model is the coarse allocation, while the small timescale model gives the fine allocation. Hence, we assume that an equal chunk allocation is implemented, i.e., every selected transcoder is assigned the same number of video chunks $a^{(Cv,k)}/\left|\Omega_s^k\right|$, where $\left|\Omega_s^k\right|$ is the cardinality of the set Ω_s^k and $a^{(Cv,k)} = N^k$. The computational time for each chunk is $L_O r/a_u^{(Cp,k)}$, where L_O is the length of the program code and input parameters for the video chunk, r (cycles per bit) is the workload/intensity for every video chunk, and $a_u^{(Cp,k)} = f_u^k$ is the computational frequency, i.e., the CPU processing rate. The communication time is the time duration for transferring the transcoded video chunk back to the RSU, which is defined as L_T/R_u^k, where L_T is the length of the transcoded video chunk and $R_u^k = a_u^{(B,k)} \log(1 + \gamma_u^k)$ is the communication rate. We neglect the communication time for transferring the program code and input parameters of the small size $L_O \ll L_T$ to the users (and the original video segments are transferred to the users offline); also, the upload delays from the RSUs to the BS are neglected due to the typically high-speed RSU-BS communication. Thus, the total time taken is $D_u^k = \left(L_O r/a_u^{(Cp,k)} + L_T/R_u^k\right) a^{(Cv,k)}/\left|\Omega_s^k\right|$. If $D_u^k \le T_{out}$, $\forall u \in \Omega_s^k$, $I_k = 1$; otherwise, $I_k = 0$.

The reward for RSU k is thus

$$\mathcal{R}^k(t) = \sum_{u \in \Omega_c^k} a_u^{(U,k)} \left(CS_u^k \theta_{CS} + \frac{a_u^{(Cv,k)}}{\left|\Omega_s^k\right|} \overline{RV}_u^k \theta_{RV} - \frac{a_u^{(Cv,k)}}{\left|\Omega_s^k\right|} C_u^k \theta_{EN} \right). \tag{6}$$

The first term in parentheses is the reward from the stakeholder (or equivalently the block reward), whereby θ_{CS} is the weight of the cumulative stake. The second term in parentheses is the reward of the reputation value, which is the benefit of the transcoding task, whereby θ_{RV} is the weight of the reputation value, i.e., the transcoding revenue. The third term in parentheses is the energy consumption cost expended for the computation and communication, where θ_{EN} is the weight of the energy consumption. $\overline{RV}_u^k = RV_u^k/RV_{\max}$ is the normalized reputation value. C_u^k can be derived as $C_u^k = \mu \left(a_u^{(Cp,k)}\right)^2 L_O r + P_u^k L_T/R_u^k$, whereby the first term (with the squared $a_u^{(Cp,k)}$ and the energy efficiency coefficient μ) is the computational power consumption, the second term is the communication energy. Thus, the total system reward can be expressed as $\mathcal{R}(t) = \sum_{k \in \mathcal{K}} \mathcal{R}^k(t) I_k$. Next, we determine the candidate set Ω_c^k.

4) Calculation of Ω_c^k With Mobility Prediction Model: We now determine the future location of a specific user in the next epoch. Consider the network with the set of RSUs $\mathcal{K} = \{1, \ldots, K\}$, where K is the number of RSUs. The probability that user u in the coverage area of RSU i is handed over before or at time t can be evaluated as

$$\begin{aligned} \Lambda_i^u(t) &= \Pr\left(T_{n+1}^u - T_n^u \le t \,\middle|\, X_n^u = i\right) \\ &= \sum_{j \in \mathcal{K}} \Phi(u)_{i,j}(t). \end{aligned} \tag{7}$$

Recall that $\Phi(u)_{i,j}(t)$ is defined in Eqn. (3) as the probability of transition from the coverage of RSU i to the coverage of its neighbor RSU j if user u has already stayed in the current coverage for time t.

The time-homogeneous Semi-Markov process of user u is defined as $X = (X_t, t \in \mathbf{R}_0^+)$. Its state transients are

$\Psi^u_{i,j}(t) = \Pr\left(X^u_t = j \,\middle|\, X^u_0 = i\right)$. The continuous-time process is now transformed into the discrete-time process $\Psi^u_{i,j}(l)$:

$$\Psi^u_{i,j}(l) = h^u_{i,j}(l) + \sum_{m\in\mathcal{K}}\sum_{\tau=1}^{k} \Psi^u_{m,j}(k-\tau)\sigma^u_{i,m}(\tau), \quad (8)$$

where $h^u_{i,j}(l) = \left[1-\Lambda^u_i(t)\right]\theta_{i,j}$; $\sigma^u_{i,m}(l) = \Phi^u_{i,m}(1)$ if $l = 1$, $\sigma^u_{i,m}(l) = \Phi^u_{i,m}(l) - \Phi^u_{i,m}(l-1)$. $\Psi^u_{i,j}(l)$ is the probability of the event that the user u is in coverage of RSU j after l amount of time from the time instant when it switches from somewhere else to the coverage of RSU i.

The location of a user at every l' time steps needs to be predicted. This event is that the user u is in the current coverage of RSU i and stays for the sojourn time $t_{soj} = s$ and then is in the coverage of RSU j after l' amount of time. The probability of the event $\hat{\Psi}^u_{i,j}(l',s) = \Pr\left(X^u_{s+l'} = j \,\middle|\, X^u_0 = i, t_{soj} = s\right)$ can be expressed as [25] and [26]:

$$\begin{aligned}\hat{\Psi}^u_{i,j}(l',s) &= \frac{\Pr\left(X^u_{s+l'} = j, t_{soj} = s \,\middle|\, X^u_0 = i\right)}{\Pr\left(t_{soj} = s \,\middle|\, X^u_0 = i,\right)} \\ &= \frac{h^u_{i,j}(l'+s) + \sum\limits_{m\in\mathcal{K}}\sum\limits_{\tau=s+1}^{l'+s}\Psi^u_{m,j}(l'+s-\tau)\sigma^u_{i,m}(\tau)}{1-\Lambda^u_i(s)}. \quad (9)\end{aligned}$$

The RSU with the highest probability $\hat{\Psi}^u_{i,j}(l',s)^* = \max\limits_{j\in\mathcal{K}_i}\hat{\Psi}^u_{i,j}(l',s)$ is chosen as the predicted future destination, where $\mathcal{K}_i$ is the set of neighboring RSUs of RSU i. We perform the similar steps for all the users to obtain the predicted set Ω^k_c of users in the coverage area of RSU k, $k \in \mathcal{K}$.

B. Improvement of Mobility Prediction

We improve the mobility predicting by developing a novel form of the Naive Bayes classifier: We incorporate convolutional neural network encoder (CNN), variational auto-encoder (VAE), and semi-supervised learning into the conventional existing Naive Bayes classifier, so as to exploit both the spatial and temporal information for improving the prediction accuracy. Due to the space constraints, we only briefly summarize our original development of the novel form of the Naive Bayes classifier; see [27] for details. Firstly, we use results from the Semi-Markov renewal process model, where the a most probable RSUs are selected to associate user u in the next epoch. Then, mobility patterns are generated depending on *1)* the mobility history data, *2)* the set of well-visited landmarks, and *3)* the distances between the current location and these possible future locations (i.e., the well-visited landmarks).

The patterns are first formulated according to the historical data and each of them is identified by a class label and time stamp. According to the given location and time stamp, the proposed model determines the posterior probability of each class. The class with the highest probability is selected as the recognized pattern, based on which the future location is determined. According to the historical data, the mobility pattern (including the location of the user for every epoch T_{out} within N_T epochs) is created. These data are accumulated to form the time-series data, which cause the arising problem of big data management, when considering large-scale networks.

For training in large-scale networks, the traditional supervised learning is not suitable because it requires a manually labelled sample for every class. This requirement is too costly and impractical in our considered scenario. In practical applications, there can exist a class with an unknown label, e.g., due to insufficient data or the arrival of a novel class. Furthermore, some of the values of observed labels may be missing and some of the labels of the training data may be completely unobserved. Therefore, the employed training and testing mechanism would be a semi-supervised learning instead of supervised learning to perform the data augmentation. From the perspective of the data augmentation, our proposed semi-supervised learning artificially increases the training set by creating modified copies of a dataset using existing labelled data. Furthermore, our data augmentation updates the set of labelled data after training for the unlabelled data, i.e., making minor changes to the dataset and/or generating new data using deep learning mechanisms.

Specifically, our proposed semi-supervised learning introduces criteria to train the novel unlabelled classes. These criteria help to determine whether the unlabelled classes belong to the already labelled classes or not, as well as to spawn novel classes. In particular, we develop a semi-supervised training algorithm for the Naive Bayes classifier, which can perform training from both labeled and unlabeled data. This proposed method is different from the original Naive Bayes classifier, which only trains from labelled data.

Next, the VAE [6], [15], [28] is used to extract the salient characteristics of the time-series data and to reliably provide additional useful criteria for the reconstruction objective. Different from the classifier based on a dataset with labeled data, the model must perform training to learn the hidden parameters (or the latent codes). The Bayesian support vector machine is used for leveraging the available class labels. Based on that, the model can formulate the precise decision boundaries for future classification. The next step is to construct the necessary latent codes by obtaining the discriminative features of the data. There are many ways to do so, among these, the CNN is a highly efficient method as it not only extracts the discriminative features, but also produces the latent codes in lower dimensions, i.e., the CNN functions as a fast encoder for the distribution [15], [18]. For the data reconstruction, the deep generative deconvolutional network is used in the decoder, whereby the latent code is the input. The training of the classifier terminates when the reconstruction loss reaches a small threshold.

V. Small Timescale ACL Model

Given the actions $\mathbf{A}^{(Cv,k)}, \mathbf{A}^{(U,k)}, \mathbf{A}^{(B,k)}$, and $\mathbf{A}^{(Cp,k)}$ decided by the large timescale actor-critic-learning model, the small timescale actions $\tilde{\mathbf{A}}^{(Cv,k)}, \tilde{\mathbf{A}}^{(U,k)}, \tilde{\mathbf{A}}^{(B,k)}$, and $\tilde{\mathbf{A}}^{(Cp,k)}$ are furthermore decided in every time slot. The user-mobility has an impact on the actions $\tilde{\mathbf{A}}^{(Cv,k)}, \tilde{\mathbf{A}}^{(U,k)}, \tilde{\mathbf{A}}^{(B,k)}$, and $\tilde{\mathbf{A}}^{(Cp,k)}$, as well as on the small timescale reward.

A. System States

Recall that there are T_{out} time slots, each having the duration of ΔT, in each epoch. At the beginning of every time slot t, $t \in [1, T_{out}]$, the state of the remaining number of chunks at every RSU k is updated, i.e., $N_t^k = N_{t-1}^k - \Delta N_{t-1}^k$, where ΔN_{t-1}^k is the number of video chunks that are completely transcoded in time slot $t-1$. The system states at time slot t include the cumulative stake CS_u^k, the reputation value RV_u^k, the computational resource F_u^k, the bandwidth B^k, and the user candidate set Ω_c^k.

B. System Actions

Within the time slot t, the agent needs to decide what users are selected as the transcoders. This action is denoted by $\tilde{\mathbf{A}}^{(U,k)} = \left\{\tilde{a}_u^{(U,k)}\right\}$, where $\tilde{a}_u^{(U,k)} = 1$ if the user u is selected as the transcoder, otherwise, $\tilde{a}_u^{(U,k)} = 0$. Furthermore, the communication action is $\tilde{\mathbf{A}}^{(B,k)} = \left\{\tilde{a}_u^{(B,k)}\right\}$, $\tilde{a}_u^{(B,k)} \in \left[1, B^k\right]$, where $\tilde{a}_u^{(B,k)}$ determines the number of subchannels assigned to user u for communications. The computational action is $\tilde{\mathbf{A}}^{(Cp,k)} = \left\{\tilde{a}_u^{(Cp,k)}\right\}$, $\tilde{a}_u^{(Cp,k)} \in \left[1, F^k\right]$, where $\tilde{a}_u^{(Cp,k)}$ determines the amount of computational resource allocated to user u. Finally, the job assignment action is $\tilde{\mathbf{A}}^{(Cv,k)} = \left\{\tilde{a}_u^{(Cv,k)}\right\}$, $\tilde{a}_u^{(Cv,k)} \in \left[1, N_t^k\right]$, where $\tilde{a}_u^{(Cv,k)}$ determines the number of video chunks assigned to the user u for transcoding.

Note that we perform the actions every time slot t and require that each assigned user must complete its transcoding job within the current time slot, i.e., within the time duration ΔT. This is different from the large timescale model, where the deadline for the video is T_{out}. This distinct operation on the small timescale vs. the large timescale can be explained as follows. In the mobility prediction model, we predict the location of a user by selecting the location with the highest probability. Therefore, the user may not stay within one RSU's coverage for the entire epoch T_{out}. It means that the user may join or leave an RSU's coverage in the middle of an epoch. Under this observation, if we assign the task to the user with the deadline of T_{out}, the user may leave the coverage before completing the task and we would lose the transcoded results from this user. Also, the best candidate can be missed if a user with powerful processing capabilities arrives too late and all the chunks have already been assigned to other users.

C. Reward Function

In this section, we determine the reward achieved by users at every time slot. $\tilde{I}_u^k$ indicates whether user u completes its assigned task before the deadline (i.e., within the time slot duration ΔT) or not. This indicator is different from that defined in the large timescale model, where the task completion is defined for the entire video. In the small timescale model, RSU k adjusts its resource allocation to complete its assigned tasks. That is, RSU k takes immediate action to allocate the new transcoding candidate for the uncompleted video chunks that have not been successfully transcoded by the selected user in the previous time slot. Note that the time taken at user u for computation and communication is $\tilde{D}_u^k = \tilde{a}_u^{(Cv,k)}\left(\frac{L_O r}{\tilde{a}_u^{(Cp,k)}} + \frac{L_T}{\tilde{R}_u^k}\right)$. The first term $\frac{L_O r}{\tilde{a}_u^{(Cp,k)}}$ is the computational delay, while the second term $\frac{L_T}{\tilde{R}_u^k}$ is the communication delay. Recall that L_O [bytes] is the length of the program code and parameters for transcoding, L_T [bytes] is the length of the transcoded video chunk, $\tilde{a}_u^{(Cv,k)}$ is the number of chunks assigned to user u, $\tilde{a}_u^{(Cp,k)}$ is the computational resource allocated to user u, and r is the required workload/intensity of every video chunk for the computing node. Moreover, $\tilde{R}_u^k$ is the communication rate, which is calculated by $\tilde{R}_u^k = \tilde{a}_u^{(B,k)} \log(1 + \gamma_u^k)$, where $\tilde{a}_u^{(B,k)}$ is the communication resource allocated to user u. We have $\tilde{I}_u^k = 1$ if $\tilde{D}_u^k \leq \Delta T$; $\tilde{I}_u^k = 0$ otherwise.

If user u fails to complete the transcoding of the assigned chunks, then it does not get the reward for neither the cumulative stake nor the reputation value. However, it still pays for its energy consumption. In this case, there are some possible options. The user may give up the transcoding in the middle of a time slot in order to save the energy or it always uses up the entire time slot even though transcoding cannot complete. Hence, it is quite complicated to capture the exact loss at the user. For simplicity, we assign the penalty ΔP per assigned chunk to a failed user. Then, the reward at RSU k is

$$\tilde{\mathcal{R}}^k(t) = \sum_{u \in \Omega_c^k} \tilde{a}_u^{(U,k)} \Big[\Big(CS_u^k \theta_{CS} + \tilde{a}_u^{(Cv,k)} \overline{RV}_u^k \theta_{RV} - \tilde{a}_u^{(Cv,k)} C_u^k \theta_{EN} \Big) \tilde{I}_u^k - (1 - \tilde{I}_u^k) \tilde{a}_u^{(Cv,k)} \Delta P \Big]. \quad (10)$$

The first part of the reward is the block reward or the stakeholder reward, where θ_{CS} is the weight of the cumulative stake. The second part is the reward of the reputation value from transcoding tasks, whereby $\overline{RV}_u^k = RV_u^k / RV_{\max}$ is the normalized reputation value and θ_{RV} is the weight of the reputation value or the transcoding revenue. The third part is the cost of the energy consumption spent for computation and communication, whereby θ_{EN} is the weight of the energy consumption; and $C_u^k = \mu \left(\tilde{a}_u^{(Cp,k)}\right)^2 L_O r + P_u^k L_T / R_u^k$, where the first term is the computational power consumption with μ denoting the energy efficiency coefficient, and the second term is the communication energy. The total system reward is $\tilde{\mathcal{R}}(t) = \sum_{k \in \mathcal{K}} \tilde{\mathcal{R}}^k(t)$.

VI. Numerical Results

A. Simulation Set-up

The key parameters are chosen as follows, unless stated otherwise: $K = 5$ RSUs; $U = 50$ users (70% stationary, 30% moving as per Levy Walk model [29]); $N = 13000$ chunks; $T_{out} = 15$ time slots; $L_O = 0.5$ KB (following [10]); $L_T = 100$ KB; $r = 1.8$ Kcycles/bit; $\mu = 10^{-26}$; $\theta_{RV} = 1$ token/chunk; $\theta_{CS} = 0.05$; $\theta_{RV} = 0.1$. $\left\{CS_u^k\right\}$ is randomly selected in the range of $[100, 300]$ tokens. The normalized reputation value $\left\{RV_u^k / RV_{\max}\right\}$ is randomly chosen in the range of $[0, 1]$. The number of available sub-channels $\left\{B^k\right\}$ is randomly selected in the range of $[5, 30]$. The available computational resources $\left\{F^k\right\}$ is in the range of $[2, 10]$ GHz. We set the time slot duration to $\Delta T = 2$ s in accordance with the Group of Picture (GOP) length in practical video streaming systems.

TABLE I

ACCURACY PERFORMANCE OF MOBILITY PREDICTION INDICATED BY 95% CONFIDENCE INTERVAL (LoB, UpB), WHERE LoB AND UpB ARE THE LOWER BOUND AND UPPER BOUND OF THE PREDICTED RANGE AS A FUNCTION OF THE LENGTH OF THE PREDICTION INTERVAL T_{out}

Methods	T_{out} = 15 slots	T_{out} = 30 slots	T_{out} = 45 slots	T_{out} = 60 slots	T_{out} = 75 slots	T_{out} = 90 slots
Most probable method	(79.9, 82.2) %	(78.4, 81.5) %	(75.3, 78.2) %	(68.5, 73.9) %	(59.6, 68.1) %	(58.0, 66.5) %
Naive Bayes method	(84.7, 87.9) %	(82.9, 86.7) %	(78.9, 83.2) %	(69.6, 76.3) %	(61.0, 70.0) %	(57.4, 68.0) %
Our improved method	(89.9, 93.0) %	(87.6, 90.9) %	(85.0, 88.2) %	(81.0, 86.8) %	(77.3, 86.0) %	(74.4, 83.2) %

The wireless channels, i.e., vehicle-to-RSU links, all follow the Markov model. During the contact time of vehicle i to RSU k, $\nu_{i,k}$ has two states, i.e., the link spectral efficiency $\nu_{i,k} = \{1, 4\}$, with 1 corresponding to the bad channel and 4 to the good channel. The probabilities of remaining in the same state and the transition from one state to the other state are, respectively, set to 0.7 and 0.3. The computation states of the users follow the Markov model, where the transition probabilities $\Theta_{i,j}$ are in the range of [0, 1] and $\sum_{i,j} \Theta_{i,j} = 1$.

To implement reinforcement learning, we employ the open-source software library TensorFlow 0.12.1.

B. Mobility Prediction Accuracy

We consider the Levy Walk model [29] as the mobility trace generation model, with a six-day training data set. We compare the real location and the predicted location of the users for every time interval T_{out}. The accuracy performance is calculated for the whole observation of all the time intervals. Table I presents the 95% confidence intervals (LoB, UpB) of the prediction results.

We compare: 1) the most probable method (widely used, e.g., in [25] and [30]), 2) the Naive Bayes classifier based method (which generally improves on the most probable method), and 3) our proposal. In our proposal, we include the Naive Bayes classifier for classification, the CNN based VAE for extracting the salient characteristics, and the semi-supervised learning for triage training and testing (see Section IV-B). We observe that our improved mechanism achieves higher accuracy performance than the others, whereby the accuracy performance values (i.e., the maximum prediction accuracy values or the upper bounds) are higher than 80% for all considered prediction intervals. This performance enhancement is due to the utilization of advanced and classical machine learning techniques, such as the CNN based VAE, the generative model of Naive Bayes classifier, and semi-supervised learning. We use the prediction results of our proposed method for the subsequent evaluations.

C. Video Transcoding Resource Management

Fig. 2 shows the probability of success versus video chunk size L_O, whereby success means that the whole video is completely transcoded by the selected users within the hard deadline T_{out}. The proposed scheme is compared with: *1)* the single timescale model, which is the same as our proposed scheme except that the pre-selection of the set of possible users and the set of chunks assigned to the RSU is omitted; and *2)* the equal scheme, which equally allocates the communication and computational resources to all the users as well as assigns the same number of video chunks to the RSUs. We observe from Fig. 2 that the probability of success decreases when the video chunk length increases for

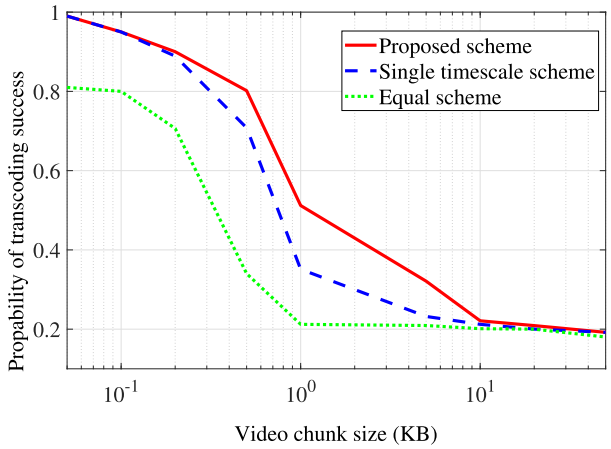

Fig. 2. Probability of success vs. size of video chunk transcoding task L_O: For mid-range task sizes (program code, input parameters) L_O, the proposed multi-timescale framework achieves significantly higher (verified with 95% confidence intervals) transcoding success probabilities than the single-timescale framework.

all the schemes. This is because it is easier to perform the computation and then transmit the transcoded version back to the RSU within the contact duration when the video chunk size L_O is smaller. Furthermore, the success probability of the equal scheme decreases dramatically, when the original chunk size L_O exceeds 0.2 KB, while the proposed scheme achieves a much higher success probability than the other two. This validates the beneficial contribution of the large timescale model design, where we carefully select the sets of essential users as the candidates and assign the right amount of video chunks to every RSU. The proposed scheme fully exploits the computational and communication resource diversity, as well as accounts for the mobility diversity amongst the users.

Fig. 3 illustrates the energy consumption as a function of the video bitrate. In particular, we transcode the video to the new version with the bitrate between 200 Kbps and 32 Mbps. For simplicity, we set the video chunk size proportional to the bitrate. We compare the proposed scheme with *1)* only single timescale model; *2)* equal scheme; and *3)* random scheme. In the random scheme, the communication and computational resources are randomly allocated to the users, while the same number of video chunks are assigned to the RSUs and users. We observe from Fig. 3 that the increase of the bitrate of the required transcoded video version increases the cost due to the energy consumption at the selected transcoders. An increased bitrate of the transcoded video increases the amount of output data and hence increases the power consumption for sending data back to the RSUs. The proposed multi-timescale framework consistently achieves significantly (at 95% confidence level) lower energy consumption, and thus significantly higher energy efficiency, than the single-timescale framework.

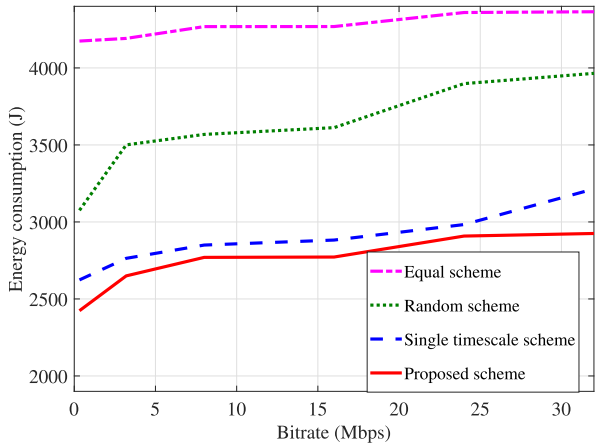

Fig. 3. Energy consumption vs. bitrate for size of video chunk transcoding task $L_O = 500$ B.

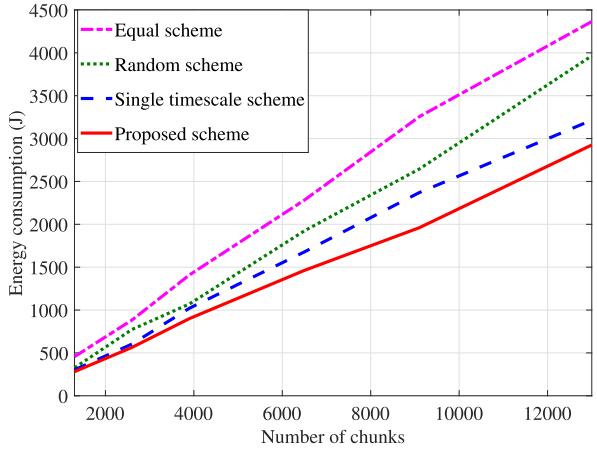

Fig. 4. Energy consumption vs. number of video chunk transcoding tasks N for $L_O = 500$ B and $L_T = 100$ KB.

We furthermore investigate the impact of the number of video chunks on the system cost in Fig. 4. We observe from Fig. 4 that when the number of chunks increases, more computational and communication resources are needed for the selected transcoders. The video chunks are carefully assigned to RSUs and their transcoders to guarantee the QoS requirements (i.e., the hard deadline constraints). The energy consumption increases, when the number of video chunks increases due to the heavy workload. However, our proposed multi-timescale framework is consistently the most energy-efficient method, as verified at the 95% confidence level, compared to the other schemes (including the single-timescale framework).

Finally, we study the transcoding revenue vs the varying bitrate of transcoded video version in Fig. 5. The proposed scheme using actor-critic reinforcement learning and multi-timescale framework achieves a higher revenue gain than the other schemes. In the proposed multi-timescale framework, we only determine the transcoder selection for a pre-selected subset of users for every RSU, i.e., the state and action spaces are smaller, and hence the cost is reduced. Specifically, the action space becomes much smaller in the small timescale model after the pre-selection of the users and chunks for each RSU is performed in the large timescale model. In contrast,

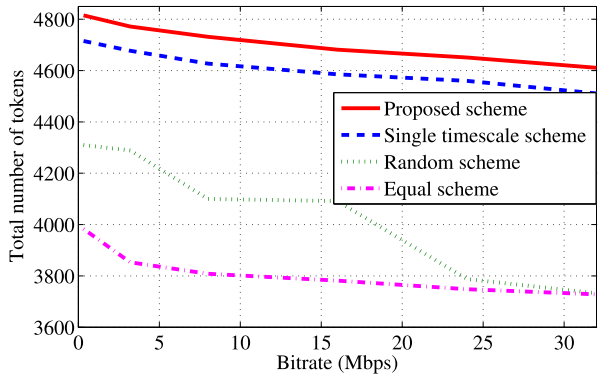

Fig. 5. Revenue vs bitrate.

the single-timescale framework needs to perform resource allocation to all the RSUs and users, leading to a much higher cost. Thus, the single-timescale framework incurs a significantly (at 95% confidence level) higher complexity than the proposed multi-timescale framework.

Furthermore, there is a flexible regulation in the proposed scheme and single-timescale scheme for the case that a user arrives to or leaves the RSU's coverage in the middle of epoch. In particular, the deadline constraint is added in the small timescale model for each time slot so that the unexpected user-mobility can be addressed in subsequent time slots. Therefore, this flexible regulation mechanism avoids the loss of revenue due to users' leaving movement. Also, performing chunk assignment in each time slot helps us to reserve more chunks for the better candidates that arrive in the subsequent time slots. Thus, the proposed multi-timescale framework exploits the dynamic user-mobility and the transcoding cooperation amongst the users to achieve a high reward gain.

VII. Conclusion

We developed a transcoder selection mechanism for a blockchain-enabled machine learning aided vehicular network under consideration of the realistic Semi-Markov renewal process mobility. In particular, a multi-timescale framework was proposed based on (instantiated with) actor-critic reinforcement learning (ACL) for solving the joint communication and computation resource allocation problem. To achieve the operational excellence and the cost efficiency, the user-mobility was exploited to enhance the computing policies, while the CNN based VAE and classic semi-supervised learning with naive Bayes classifier were used for predicting the vehicular mobility. To address the complexity caused by the large state and action spaces, we proposed a mobility-aware reward estimation for the large timescale model to effectively reduce these spaces. Numerical results obtained with ACL demonstrated the significant resource allocation improvements and energy consumption reductions achieved by the proposed multi-timescale framework. Adaptations of the multi-timescale framework to other reinforcement learning instantiations, such as policy search or separate value functions, are an interesting direction for future engineering work.

Building on the present study, one future research direction is to develop a twin-timescale mobility-aware AI aided framework for 5G vehicular network slicing, which includes a monitoring system of the slice SLA and slice adaption based on

the traffic dynamics as well as effective admission control that balances 1) infrastructure utilization, 2) service provisioning, and 3) provider revenue. Furthermore, future research should develop an effective scheduling mechanism for isolating and protecting 5G network slicing. Moreover, future research should pursue AI-based resource allocation for integrated network slicing, communication, caching, and computing.

REFERENCES

[1] X. Jiang, F. R. Yu, T. Song, and V. C. M. Leung, "Resource allocation of video streaming over vehicular networks: A survey, some research issues and challenges," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 5955–5975, Jul. 2022.

[2] P. Arthurs, L. Gillam, P. Krause, N. Wang, K. Halder, and A. Mouzakitis, "A taxonomy and survey of edge cloud computing for intelligent transportation systems and connected vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 6206–6221, Jul. 2022.

[3] Q. Wang, L. T. Tan, R. Q. Hu, and Y. Qian, "Hierarchical energy-efficient mobile-edge computing in IoT networks," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11626–11639, Dec. 2020.

[4] L. T. Tan, R. Q. Hu, and L. Hanzo, "Heterogeneous networks relying on full-duplex relays and mobility-aware probabilistic caching," *IEEE Trans. Commun.*, vol. 67, no. 7, pp. 5037–5052, Jul. 2019.

[5] L. T. Tan, R. Q. Hu, and L. Hanzo, "Twin-timescale artificial intelligence aided mobility-aware edge caching and computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3086–3099, Apr. 2019.

[6] K. Muhammad, A. Ullah, J. Lloret, J. D. Ser, and V. H. C. de Albuquerque, "Deep learning for safe autonomous driving: Current challenges and future directions," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4316–4336, Jul. 2021.

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

[8] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of blockchains in the Internet of Things: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1676–1717, 2nd Quart., 2019.

[9] *Transcodium: A Decentralized Peer-to-Peer Media Editing, Transcoding & Distribution Platform*. Accessed: Aug. 12, 2023. [Online]. Available: https://www.allcryptowhitepapers.com/transcodium-whitepaper

[10] M. Liu, Y. Teng, F. R. Yu, V. C. M. Leung, and M. Song, "A deep reinforcement learning-based transcoder selection framework for blockchain-enabled wireless D2D transcoding," *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3426–3439, Jun. 2020.

[11] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 695–708, Jan. 2019.

[12] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, "Anonymous reputation system for IIoT-enabled retail marketing atop PoS blockchain," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3527–3537, Jun. 2019.

[13] A. Haydari and Y. Yilmaz, "Deep reinforcement learning for intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 1, pp. 11–32, Jan. 2022.

[14] Q. Wang, G. M. Garrity, J. M. Tiedje, and J. R. Cole, "Naïve Bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy," *Appl. Environ. Microbiol.*, vol. 73, no. 16, pp. 5261–5267, Aug. 2007.

[15] T. Le and S. Shetty, "Artificial intelligence-aided privacy preserving trustworthy computation and communication in 5G-based IoT networks," *Ad Hoc Netw.*, vol. 126, Mar. 2022, Art. no. 102752.

[16] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor–critic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, 2009.

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

[18] S. Mozaffari, O. Y. Al-Jarrah, M. Dianati, P. Jennings, and A. Mouzakitis, "Deep learning-based vehicle behavior prediction for autonomous driving applications: A review," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 1, pp. 33–47, Jan. 2022.

[19] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 740–759, Feb. 2022.

[20] B. R. Kiran et al., "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, Jun. 2022.

[21] X. Jiang, F. R. Yu, T. Song, and V. C. M. Leung, "A survey on multi-access edge computing applied to video streaming: Some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 871–903, 2nd Quart., 2021.

[22] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, nos. 1–2, pp. 181–211, Aug. 1999.

[23] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. NIPS*, vol. 29, 2016, pp. 1–9.

[24] H. Soo 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.

[25] H. Farooq, A. Asghar, and A. Imran, "Mobility prediction-based autonomous proactive energy saving (AURORA) framework for emerging ultra-dense networks," *IEEE Trans. Green Commun. Netw.*, vol. 2, no. 4, pp. 958–971, Dec. 2018.

[26] J.-K. Lee and J. C. Hou, "Modeling steady-state and transient behaviors of user mobility: Formulation, analysis, and application," in *Proc. ACM MobiHoc*, 2006, pp. 85–96.

[27] T. Le, M. Reisslein, and S. Shetty. *Multi-Timescale Actor-Critic Learning for Computing Resource Management With Semi-Markov Renewal Process Mobility (Extended Version)*. Accessed: Aug. 12, 2023. [Online]. Available: https://www.dropbox.com/s/k4m8d4rs1oyqjtn/ABTCVNTechnicalreport.pdf?dl=0

[28] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.

[29] I. Rhee, M. Shin, S. Hong, K. Lee, S. J. Kim, and S. Chong, "On the Levy-walk nature of human mobility," *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, pp. 630–643, Jun. 2011.

[30] Q. Yuan, I. Cardei, and J. Wu, "An efficient prediction-based routing in disruption-tolerant networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 1, pp. 19–31, Jan. 2012.

Tan Le (Member, IEEE) received the Ph.D. degree from the University of Quebec in Montreal, Montreal, QC, Canada, in 2015. He was a Research Assistant Professor with the Virginia Modeling, Analysis and Simulation Center (VMASC), Old Dominion University, Suffolk, VA, USA, and he is currently an Assistant Professor with Hampton University, Hampton, VA, USA.

Martin Reisslein (Fellow, IEEE) received the Ph.D. degree in systems engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 1998. He is currently a Professor with the School of Electrical, Computer and Energy Engineering, Arizona State University (ASU), Tempe, AZ, USA, and the Program Chair of Computer Engineering (CEN) at ASU.

Sachin Shetty (Senior Member, IEEE) received the Ph.D. degree from Old Dominion University, Suffolk, VA, USA, in 2007. He is currently an Executive Director with the Center for Secure and Intelligent Critical Systems (CSICS), Virginia Modeling, Analysis and Simulation Center (VMASC), and a Professor with the Department of Computational Modeling and Simulation Engineering, Old Dominion University.