Learning-Based Computing Task Offloading for Autonomous Driving: A Load Balancing Perspective

Qiang Ye*, Weisen Shi†, Kaige Qu†, Hongli He‡, Weihua Zhuang†, and Xuemin (Sherman) Shen†
*Department of Electrical and Computer Engineering & Technology, Minnesota State University, Mankato, USA
†Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada
‡College of Information Science & Electronic Engineering, Zhejiang University, Hangzhou, China

ICC 2021 - IEEE International Conference on Communications | 978-1-7281-7122-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICC42927.2021.9500293

***Abstract*—In this paper, we investigate a computing task offloading problem in a cloud-based autonomous vehicular network (C-AVN), from the perspective of long-term network wide computation load balancing. To capture the task computation load dynamics over time, we describe the problem as an Markov decision process (MDP) with constraints. Specifically, the objective is to minimize the expectation of a long-term total cost for imbalanced base station (BS) computation load and task offloading decision switching, with per-slot computation capacity and offloading latency constraints. To deal with the unknown state transition probability and large state-action spaces, a multi-agent deep *Q*-learning (MA-DQL) module is designed, in which all the agents cooperatively learn a joint optimal task offloading policy by training individual deep *Q*-network (DQN) parameters based on local observations. To stabilize the learning performance, a fingerprint-based method is adopted to describe the observation of each agent by including an abstraction of every other agent's updated state and policy. Simulation results show the effectiveness of the proposed task offloading framework in achieving long-term computation load balancing with controlled offloading switching times and per-slot QoS guarantee.**

I. Introduction

Promoting more autonomous vehicles (AVs) on the roads have been one of the focal points from both academia and industry in developing the next generation intelligent transportation systems (ITS) [1], [2]. AVs help to relieve human driving stress, intelligently navigate vehicle traffic, and regulate driving behaviors. In order to achieve safety and more efficient AV driving patterns, the on-board vehicle sensing is the key to perceive environment information to provide different AV services (e.g., object detection, object tracking and localization, and data fusion [1]) and react to the environment changes precisely and promptly (e.g., lane changing, emergency brake, and curve turning [2]). However, only relying on the vehicle sensing technology to realize high levels of automation is far from enough, considering the limited sensing ranges for local environment information acquisition. Besides, in response to vehicle traffic dynamics, an increasing amount of sensing tasks are required to be processed/computed (e.g., data compression and data fusion) to achieve high automation, which often overwhelms the limited on-board computation capacities.

Recent research endeavors promote the development of advanced computing and networking technologies to enhance the data processing and information sharing capabilities [2]. With edge computing and vehicle-to-infrastructure (V2I) communications, such as long term evolution-vehicle (LTE-V) [3], AVs can offload their sensing tasks, through near-road access points (APs) or macro-cell/small-cell cellular base stations (MBSs/SBSs), to high-performance computing (HPC) servers connected to the communication infrastructures. Some researchers also propose to migrate the computing tasks to other AVs in close proximity with available computation resources using vehicle-to-vehicle (V2V) communications [4]. To further enhance the edge computing performance, existing works mainly focus on how to determine optimal decisions between local computing and task offloading, with the objective of minimizing the edge computing cost, in terms of task transmission latency or power consumption [5], [6], or maximizing the network utility for task offloading under quality-of-service (QoS) constraints [4]. Most of the works consider a single edge computing layer with one BS providing services for a group of vehicles in a certain area. With more AVs generating increased and differentiated task computing demands, a layered vehicular networking architecture, e.g., MBSs underlaid by SBSs, equipped with different levels of HPC capacity, is preferred to provide fast task offloading and high computing efficiency.

Considering differentiated computing capacities and vehicular traffic dynamics, how to make task offloading decisions, from a network operator's perspective, to balance the network wide computation load for improving the resource utilization is an important research issue. Some studies shed light on facilitating computation load balancing among vehicles and BSs, by designing load balanced network utility functions (e.g., logarithm) [7] or by minimizing the difference of computation load levels among BSs [8]. Using optimization frameworks with different load balancing metrics are effective in dealing with one-shot task offloading. However, when the load balancing is expected over time, solving optimization problems in each task scheduling slot (in the scale of milliseconds) is computationally complex. Besides, frequent switching of task offloading decisions among BSs over consecutive slots needs to be avoided.

In this paper, we propose a learning-based computing task offloading framework in a multi-tier autonomous vehicular network (AVN) with layered edge computing. The objective is to achieve long-term computation load balancing among BSs and, at the same time, reduce the offloading switching times. To capture the task computation load dynamics over sequential scheduling slots, we formulate the task offloading problem as a Markov decision process (MDP) with per-slot offloading delay and BS computation capacity constraints. Due to the large prob-

978-1-7281-7122-7/21/$31.00 ©2021 IEEE

lem size and unavailability of state transition probabilities, we employ a multi-agent deep Q-learning (MA-DQL) technique to obtain the optimal policy for task offloading through interacting with the network environment [9], [10]. Each agent distributively learns its own policy based on local observation, but cooperatively maximizes a common reward, which substantially reduces the complexity in solving the problem. To facilitate the learning convergence, a fingerprint-based method is adopted to customize the observation description of each agent. Computer simulations are conducted based on real vehicle traffic trace to demonstrate the network wide computation load balancing under capacity and latency constraints.

II. System Model

Consider a two-tier uplink AVN, as shown in Fig. 1, where a single MBS, denoted by B_0, is deployed at the center of a macro-cell in the first network tier to provide a wide-area communication coverage for AVs moving on a road segment. The macro-cell is underlaid by n SBSs, denoted by $\{B_1, B_2, ..., B_n\}$, within its coverage area in the second tier. The SBSs covering small communication areas are placed near the road to support increased task offloading demands from AVs. We use set $\mathcal{B} = \{B_0, B_1, ..., B_n\}$ to denote all the BSs in the considered network scenario. The AVN under consideration

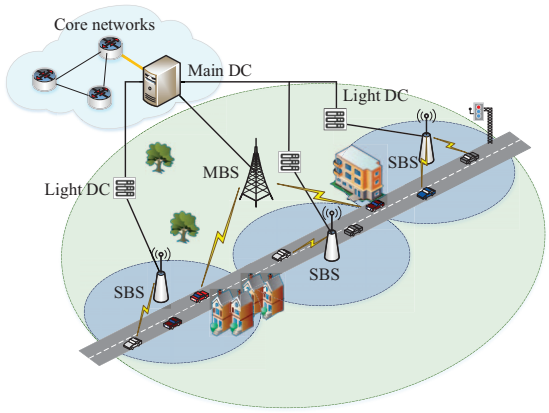


Fig. 1: A two-tier AVN with layered edge computing.

is cloud enabled, also named cloud-AVN (C-AVN), where the baseband processing and radio resource management (RRM) functionalities on each BS are virtualized and migrated to a two-layer edge computing architecture [11], as shown in Fig. 1. The master-level computing layer is provisioned in a main DC node hosting intensive computing resources at the edge of core networks, which is physically connected to the MBS and provides different virtualized processing functionalities that need to be centralized, including mobility management and centralized RRM. The slave-level computing layer consists of a number of light DC nodes with less computing capacities, each connected to one SBS. Every light DC node provides processing/computing functionalities for vehicle access control and distributed RRM under each SBS, and is controlled by the main DC node through wired links.

AVs are moving on the road segment and enter or leave the road as time elapses. Assume that each AV on the road segment is always under the coverages of the MBS and one of the SBSs, and connects to both BSs. Offloading decisions for different tasks on an AV can be switched between the BSs at the cost of extra waiting time and consumed power. Suppose that time is partitioned in a sequence of fixed-duration (T) scheduling slots, and T is set same as the delay upper bound for single task execution. For analysis tractability, we assume that the road segment under the coverage of B_0 is divided into a number of disjoint road zones, with the set of zone indexes denoted by $\mathcal{Z}_0 = \{1, 2, ..., Z\}$, and the AVs with generated tasks (if any) in one zone make a unified offloading decision to one BS covering the zone (either an SBS or the MBS). Denote the subset of zone indexes under the coverage of SBS B_k as $\mathcal{Z}_k$ ($\subset \mathcal{Z}_0$). If an AV generates a task in its buffer at a slot, the vehicle is termed as an *active AV*. The number of active AVs in zone z ($\in \mathcal{Z}_k$) under B_k at slot t is denoted by $N_k^z(t)$. At each time slot, BSs collect updated AV transmission buffer status and make task offloading decisions for active AVs in each zone.

A. Task Model

Computing tasks are consistently generated on AVs for object detection service [1]. Each task needs to be offloaded to a selected BS for further processing and the task execution result is sent back to trigger different AV operations (e.g., lane changing, acceleration/deceleration). Each task offloading process needs to be completed within certain latency constraint to ensure timely response. If the computing output is not transmitted back within the delay bound, the task will be discarded. By setting T as the delay bound for task offloading, which is a small value, we assume that at most one task is generated on an AV at a time [12]. The computing task generation at slot t on each vehicle follows Bernoulli distribution with parameter p [5]. Therefore, at every time slot, the transmission buffer of each active AV contains at most one generated task to be offloaded. As the computation capacity on BSs are high and the computation output is of small size, both task computing delay and transmission delay of computation result are negligible [5], [12]. The size of each generated task (in unit of bits) from an active AV is assumed identical, and H_k denotes the maximum task size from AVs under B_k. The computing speed (in unit of CPU cycles per second) at a DC node connected to B_k is C_k, and the computation intensity (i.e., number of cycles required to process one information bit) is ϕ, same for all tasks.

B. Communication Model

Suppose that radio resource allocation is determined in each resource planning period (in a time scale of minutes or hours) based on the long-term task arrival statistics and network information (e.g., vehicular traffic load, wireless channel status). We assume that the bandwidth resources on the MBS and SBSs are orthogonal to avoid inter-tier interference, and AVs under B_k ($\in \mathcal{B}$) are allocated an uplink channel with equal bandwidth and use a fixed uplink transmit power, denoted by W_k and P_k, respectively [12]. The bandwidth resources among the SBSs

can be reused with controlled interference to exploit resource multiplexing.

According to the Shannon capacity formula, we calculate the average uplink transmission rate from each AV in zone $z\,(\in \mathcal{Z}_k)$ to B_k at slot t as

$$r_k^z(t) = W_k \log_2\left(1 + \overline{I_k^z(t)}\right) \tag{1}$$

where $\overline{I_k^z(t)}$ denotes average signal-to-noise power ratio (SNR) from an AV in zone z to B_0 (when $k = 0$) or average signal-to-interference-plus-noise power ratio (SINR) at SBS B_k (when $k = 1, 2, ..., n$). $\overline{I_k^z(t)}$ is calculated by considering the uplink transmit power P_k, the squared uplink channel gain $\overline{G_k^z(t)}$ averaged over a group of active AVs in zone z of B_k at slot t, and the average background noise power σ^2. Instantaneous uplink channel gain at each time slot consists of the path loss and log-normal shadowing components, which stay unchanged over a number of sequential slots, and the fast fading component (assumed to be exponentially distributed with unit mean) [10]. An upper bound of inter-cell interference is also included in $\overline{I_k^z(t)}$ for the SINR calculation, assuming bandwidth resources are allocated in a way that the interference in one zone (e.g., the first zone) under the coverage of one SBS only comes from corresponding zones (e.g, the first zones) under the coverages of the other SBSs.

III. Problem Formulation

The objective of the problem under consideration is to determine how AVs in each road zone offload computing tasks among BSs to balance the network wide computation load and reduce the offloading decision switching times in a long term, under offloading delay and computation capacity constraints. At slot t, the computation load at B_k is calculated as

$$L_k(t) = \frac{\sum\limits_{z \in \mathcal{Z}_k} N_k^z(t) a_k^z(t) \phi H_k}{C_k T} \tag{2}$$

where $a_k^z(t)$ denotes a unified task offloading decision for AVs in zone z under B_k. We have $a_k^z(t) = 1$, if active AVs in zone z offload the computing tasks to B_k; Otherwise, $a_k^z(t) = 0$.

The cost of imbalanced task computing load among BSs at slot t is interpreted as the maximum instantaneous BS computation load [13], denoted by $C_1(t) = \max_{B_k \in \mathcal{B}} \{L_k(t)\}$. To avoid frequent switching of task offloading decisions in each road zone, we design a switching cost by counting the number of offloading decision changes between consecutive time slots over the road zones under all BSs, given by

$$C_2(t) = \sum_{B_k \in \mathcal{B}} \sum_{z \in \mathcal{Z}_k} \sum_{j \in \mathcal{B}' \setminus B_k} a_k^z(t) a_j^z(t-1) \tag{3}$$

where $\mathcal{B}'$ is the set including the MBS and the SBS covering zone z. The total cost of imbalanced computation load and offloading decision switching is a weighted sum of $C_1(t)$ and $C_2(t)$, expressed as

$$C(t) = \lambda C_1(t) + (1 - \lambda) C_2(t) \tag{4}$$

where λ is a weighting factor within the interval $(0, 1)$.

As our objective is to strike the balance between the network wide computation load and the task offloading switching cost over time, we describe the problem as an MDP formulation with per-slot constraints to capture the network dynamics over sequential time slots and model the interaction between network states and policies. An MDP formulation is often described by a four dimensional tuple, composed of state $\mathcal{S}$, action $\mathcal{A}$, state transition probability $P(\mathcal{S}'|\mathcal{S}, \mathcal{A})$, and reward $R(\mathcal{S}, \mathcal{A})$. In the considered scenario, the system state at time slot t is defined as $\mathcal{S}(t) = \{\mathcal{N}(t), \mathcal{I}(t), \mathcal{A}(t-1)\}$, where $\mathcal{N}(t)$ indicates the set of active AV numbers in each road zone, $\mathcal{I}(t)$ is the set of average SNR and SINR between an AV in a zone to a BS, and $\mathcal{A}(t-1)$ denotes a combination of task offloading decisions for AVs in each zone under each BS at slot $t-1$. The system action at slot t is denoted by $\mathcal{A}(t)$.

The state transition from t to $t+1$ consists of two parts: 1) the change of $\mathcal{N}(t)$ and $\mathcal{I}(t)$ due to dynamics of AV active status and vehicle movement over time, and 2) the update of $\mathcal{A}(t-1)$ to $\mathcal{A}(t)$ triggered by the action taken at t. A stationary policy $\mathbf{\Pi}(\mathcal{A}|\mathcal{S})$ in MDP is defined as the steady-state probability of taking action $\mathcal{A}$ under state $\mathcal{S}$. Therefore, we aim at finding an optimal policy $\mathbf{\Pi}^*(\mathcal{A}|\mathcal{S})$ that minimizes the expectation of a long-term total cost for achieving network wide load balancing (i.e., maximizes a long-term total reward) under BS computation capacity constraints and task offloading delay bound in each time slot. The MDP-based problem formulation is given as $(\mathrm{P1})$:

$$\min_{\Pi} \mathbb{E}\left[\lim_{T \to +\infty} \frac{1}{T} \sum_{t=1}^{T} C(\mathcal{S}(t)) \middle| \Pi\right]$$

$$\text{s.t.} \begin{cases} \sum\limits_{B_j \in \mathcal{B}'} a_j^z(t) \le 1, \quad z \in \mathcal{Z}_k,\, B_k \in \mathcal{B} & \text{(5a)} \\ \sum\limits_{z \in \mathcal{Z}_k} N_k^z(t) a_k^z(t) \le \dfrac{C_k T}{\phi H_k}, \quad B_k \in \mathcal{B} & \text{(5b)} \\ \dfrac{H_k}{r_k^z(t)} a_k^z(t) \le D_b, \quad z \in \mathcal{Z}_k,\, B_k \in \mathcal{B} & \text{(5c)} \end{cases}$$

where $C(\mathcal{S}(t))$ is the per-slot total cost calculated by (4), and D_b is the task offloading delay bound. In $(\mathrm{P1})$, constraint (5a) indicates that active AVs in each zone take a unified action to offload tasks to either the MBS or the SBS providing the coverage. Constraint (5b) guarantees the instantaneous BS computation load at each time slot not exceed the computation capacity. Constraint (5c) indicates the maximum task transmission delay from active AVs in each time slot should be less than or equal to D_b.

The formulated problem has high state and action dimensions that reach to $3\sum\limits_{k=0}^{n} |\mathcal{Z}_k|$ and $\sum\limits_{k=0}^{n} |\mathcal{Z}_k|$, respectively, where $|\cdot|$ determines set cardinality. Thus, both state and action spaces are normally large. Besides, due to the dynamic computing task load, the state transition probability function $P(\mathcal{S}'|\mathcal{S}, \mathcal{A})$ is often implicit and is hard to be known as a priori. Therefore, conventional MDP algorithms, e.g., value iteration [5], for solving the stochastic optimization problem may not be applied.

Instead, we consider to use a DQL-based approach to deal with the problem complexity caused by large state and action spaces, and learn the optimal steady-state task offloading policy.

IV. Cooperative Multi-Agent Deep Q-Learning Approach

DQL is a model-based deep reinforcement learning (DRL) method that uses trial-and-error interaction with the environment to accumulate state-action-reward transition samples to train a deep Q-network (DQN) [14], a structured deep neural network (DNN) with state as input and Q value as output. An established DQN output can well approximate the action-value function $Q\left(\mathcal{S}(t), \mathcal{A}(t)\right)$ described by the Bellman equation, given by

$$\begin{aligned} Q\left(\mathcal{S}(t), \mathcal{A}(t)\right) &= (1-\alpha) Q\left(\mathcal{S}(t), \mathcal{A}(t)\right) \\ &+ \alpha \left[R(\mathcal{S}(t), \mathcal{A}(t)) + \gamma \max_{\mathcal{A}(t+1)} Q(\mathcal{S}(t+1), \mathcal{A}(t+1)) \right] \end{aligned} \tag{6}$$

where α is the learning rate that balances learning time with accuracy, and γ is the discount factor indicating how much the future reward is discounted in each learning time step. We use DQL with experience replay to handle complex learning tasks in an environment with high-dimension state-action representations and a discrete action space [14]. The DQL has much higher learning efficiency than conventional Q-table based reinforcement learning which may experience "curse of dimensionality" in a high-dimension state-action problem structure. However, if we adopt a single-agent DQL and put the learning module in the DC node connected to the MBS which has a global view on the network environment, it would be difficult to obtain good learning performance due to increased state and action spaces and huge amount of real-time information exchanged between AVs and the DC node.

With the two-layer edge computing architecture, we design an MA-DQL module where each SBS acts as a learning agent to interact with its local network environment to gain experience and obtain its own decision policy. All agents share a same reward function as a unified learning objective, i.e., balancing the BS computation load with reduced task offloading switching times, which turns the multi-agent learning process as a cooperative game [9], [10]. A joint reward is obtained after actions are taken from all the agents in each time slot to collaboratively explore the global network environment. As actions are taken only based on local observations, the state and action spaces of each agent are substantially reduced. A detailed design of the MA-DQL module is in the following.

1) Observation with fingerprint: The local observation $\mathcal{O}_k(t)$ from agent k (SBS B_k) at slot t is described as a function of global state $\mathcal{S}(t)$, given by

$$\mathcal{O}_k(t) = \mathcal{F}(\mathcal{S}(t), k) = \{\mathcal{N}_k(t), \mathcal{I}_k(t), \mathcal{A}_k(t-1)\} \tag{7}$$

where $\mathcal{F}(\cdot)$ describes the mapping relation, $\mathcal{N}_k(t) = \{N_k^z(t), z \in \mathcal{Z}_k\}$, $\mathcal{I}_k(t) = \{\overline{I_0^z(t)}, \overline{I_k^z(t)}, z \in \mathcal{Z}_k\}$, and $\mathcal{A}_k(t-1) = \{a_k^z(t-1), z \in \mathcal{Z}_k\}$. Relying on local observation to take actions works well for conventional multi-agent Q-learning, which is also termed as independent Q-learning [9]. However, for the cooperative MA-DQL, all agents share the same reward function but each one trains its own DQN parameter, denoted by $\boldsymbol{\theta}_k$ for agent $k\,(k = 1, 2, ..., n)$. Thus, each agent would face a non-stationary network environment while the other agents adjusting their actions at the same time. The problem could become more severe when experienced replay is employed to improve the learning convergence [9]. To stabilize the learning performance, say for agent k, we include a low dimensional fingerprint reflecting the update process of other agents' polices including the latest DQN training iteration steps from all the agents excluding agent k, denoted by $\boldsymbol{e}_{-k}$, and the rate of exploration $\boldsymbol{\epsilon}_{-k}$ used in ϵ-greedy policy [10]. We also customize the observation design for agent k in slot t by including the abstracted state information from the other agents and the MBS in slot $t-1$ to further stabilize the learning process. That is the set of computation load shared by other BSs, denoted by $\mathcal{L}_{-k}(t-1)$. With the layered edge computing architecture, the computation load on each BS can be synchronized at the main DC node within slot $t-1$ and shared among SBSs at the beginning of slot t. The customized observation space for agent k is expressed as

$$\mathcal{O}_k^{(f)}(t) = \{\mathcal{O}_k(t), \mathcal{L}_{-k}(t-1), \boldsymbol{e}_{-k}, \boldsymbol{\epsilon}_{-k}\}. \tag{8}$$

2) Action: The action for agent k includes a set of task offloading decisions in each zone under SBS B_k at time slot t, denoted by $\mathcal{A}_k(t) = \{a_k^z(t), z \in \mathcal{Z}_k\}$. Compared with the single agent DQL, the action space is reduced to $2^{|\mathcal{Z}_k|}$ which only depends on the number of zones under SBS B_k.

3) Reward: The per-slot reward function is designed considering the total cost for BS computation load balancing and penalties against constraint violation, given by

$$\begin{aligned} R(t) = -C(t) &- V_1 \sum_{B_k \in \mathcal{B}} \mathbf{1}\left(\sum_{z \in \mathcal{Z}_k} N_k^z(t) a_k^z(t) > \frac{C_k T}{\phi H_k} \right) \\ &- V_2 \sum_{B_k \in \mathcal{B}} \sum_{z \in \mathcal{Z}_k} \mathbf{1}\left(\frac{H_k}{r_k^z(t)} a_k^z(t) > D_b \right) \end{aligned} \tag{9}$$

where V_1 and V_2 denote the penalties when the BS computation capacity is exceeded and the task offloading delay constraint is violated, and $\mathbf{1}(\cdot)$ is an indicator function which equals 1 if the condition is met and equals to 0 otherwise. According to (9), all the agents cooperatively take actions to maximize the accumulated reward over time, which is equivalent to minimizing the network-wide computation load balancing cost while satisfying the capacity and delay constraints to the maximum extent.

4) MA-DQL algorithm: We consider an episodic learning setting where the network environment is initialized at the beginning of each episode including AV traffic pattens, BS configuration parameters, and channel conditions. Within each episode, each agent (i.e., SBS) updates the local observation information at every time slot (training time step) and learns the offloading decisions for its covering AVs in each road zone. Through multiple episodes of training, the set of optimal policy is converged over different vehicular environment states to maximize the long-term reward. Specifically, each agent has

two DQNs: a target DQN and an evaluation DQN. In time step t, any agent, k, takes action $\mathcal{A}_k(t)$ based on $\mathcal{O}_k^{(f)}(t)$ using ϵ-greedy policy [14], and all agents' actions jointly generate a system reward, $R(t)$. Then, a state transition tuple $\left[\mathcal{O}_k^{(f)}(t), \mathcal{A}_k(t), R(t), \mathcal{O}_k^{(f)}(t+1)\right]$ is observed and stored in the replay memory $\mathcal{M}_k$ of agent k. For each episode, a mini-batch $\mathcal{D}_k$ of state transitions is randomly sampled from replay memory $\mathcal{M}_k$ to train the evaluation DQN parameter $\boldsymbol{\theta}_k$, by minimizing a squared loss function given in (10) based on stochastic gradient descent [10], [14].

$$L(\boldsymbol{\theta}_k) = \sum_{t \in \mathcal{T}_k} [R(t) + \gamma \max_{\mathcal{A}'_k(t+1)} \hat{Q}(\mathcal{O}_k^{(f)}(t+1), \mathcal{A}'_k(t+1); \hat{\boldsymbol{\theta}}_k) - Q(\mathcal{O}_k^{(f)}(t), \mathcal{A}_k(t); \boldsymbol{\theta}_k)]^2 \quad (10)$$

where $\mathcal{T}_k$ is the set of time steps when each of the transition tuples in mini-batch $\mathcal{D}_k$ is collected, $Q(;\boldsymbol{\theta}_k)$ and $\hat{Q}(;\hat{\boldsymbol{\theta}}_k)$ denotes the action-value function approximated by the evaluation DQN and the target DQN, with the parameters being $\boldsymbol{\theta}_k$ and $\hat{\boldsymbol{\theta}}_k$, respectively. This mini-batch sampling process is called experience replay, which is a key feature in DQN to stabilize the learning convergence. After M episodes, the target DQN parameter $\hat{\boldsymbol{\theta}}_k$ of any agent k is replaced by the updated evaluation DQN parameter $\boldsymbol{\theta}_k$. The detailed algorithm is presented in Algorithm 1.

Note that the realization of the algorithm relies on a two-stage learning process: centralized training and distributed execution [10]. In the training stage, each agent train its own DQN parameters by accessing the total system reward; In the execution stage, each agent takes its own action based on the local observation and the trained DQN. After some time (e.g., minutes) when the AV traffic pattern varies significantly, the training stage is trigged again to update the DQN parameters in response to the environment change.

Algorithm 1: MA-DQL algorithm with experience replay for task offloading

Initialize: Vehicular network environment, DQN parameters $\boldsymbol{\theta}_k$ and $\hat{\boldsymbol{\theta}}_k$, replay memory $\mathcal{M}_k$

for *each episode* **do**
  Load AV traffic trace and initialize network state;
  for *any time step* t **do**
    for *any agent* k **do**
      Update $\mathcal{O}_k^{(f)}(t)$;
      Take action $\mathcal{A}_k(t)$ based on $\mathcal{O}_k^{(f)}(t)$ using ϵ-greedy policy;
    end
    Observe the joint system reward $R(t)$;
    Update observation to $\mathcal{O}_k^{(f)}(t+1)$;
    for *any agent* k **do**
      Store the state transition tuple $\left[\mathcal{O}_k^{(f)}(t), \mathcal{A}_k(t), R(t), \mathcal{O}_k^{(f)}(t+1)\right]$ in replay memory $\mathcal{M}_k$;
    end
  end
  for *any agent* k **do**
    Randomly sample a min-batch $\mathcal{D}_k$ of transition tuples from $\mathcal{M}_k$;
    Use the mini-batch samples to train the evaluation DQN parameter $\boldsymbol{\theta}_k$ based on a stochastic gradient descent method;
    Copy $\boldsymbol{\theta}_k$ to $\hat{\boldsymbol{\theta}}_k$ every M episodes;
  end
end

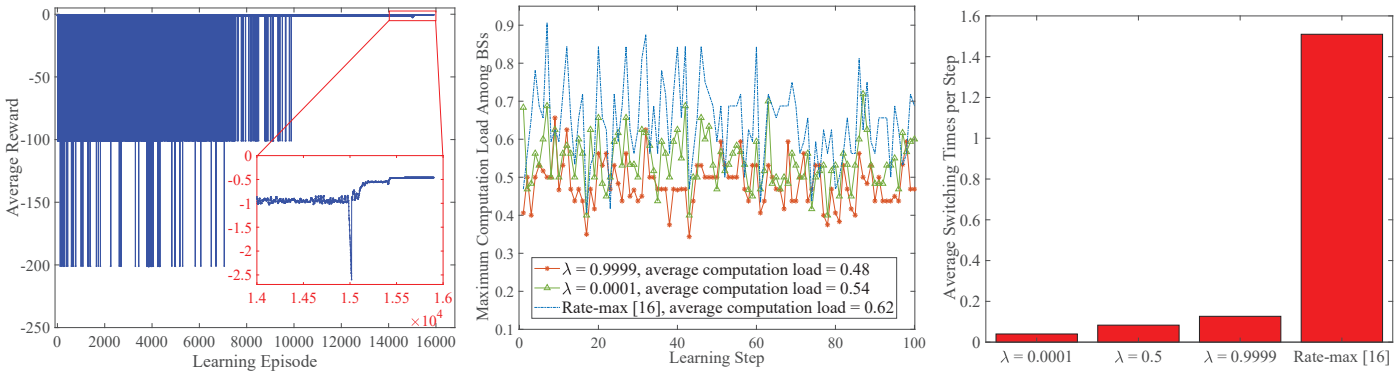
V. Simulation Results

Simulations are conducted to verify the effectiveness of the proposed task offloading framework for computation load balancing. The network scenario and the learning module are created using Python 3.7 IDE with Tensorflow 1.14.0. To simulate vehicle traffic patterns, we use SUMO traffic simulator to build a bi-directional two-line road segment of 1 km with a vehicle traffic trace loaded from a provincial roadway in Xinjiang, China [15]. The average speed of each vehicle is between 23m/s to 28m/s. We deploy 1 MBS underlaid by 2 SBSs, each with the coverage radius of 500 m and 250 m, respectively, to cover the road. The minimum distances from the MBS and each of the SBSs to the road segment are set as 20 m and 10 m, and the main and light DCs are equipped with computation capacities of 6 Mbps and 3.2 Mbps, respectively. The road segment under the MBS coverage is partitioned into 10 disjoint zones, among which each SBS covers 5 zones, and the average active AV number in each zone is around 5. The uplink transmit power from AVs to the MBS and the SBSs are set as 27 dBm and 25 dBm. Each SBS connected to a light DC has a DQL module with two structured DQNs. Each DQN has 3 hidden layers with $(128, 64, 64)$ neurons, respectively, between the input and output layers. Relu nonlinear activation function is used for all the hidden layers. Other system and learning parameters are summarized in Table I.

TABLE I: System and learning parameters

System parameters	Values	Learning parameters	Values
Channel bandwidth	200 kHz	Learning rate	10^{-3}
Noise power	-104 dBm	Discount factor	0.9
Maximum task size	5000 bits	Exploration rate	0.9999
Path loss exponent	3.5	Exploration decay	0.0002
Log-normal shadowing	-30 dB	Replay buffer size	5000
AV active probability	0.5	Mini-batch size	64
delay bound/step time	50 ms	Steps per episode	1000
Penalty (V_1/V_2)	2000/10000	Replace episodes (M)	40

We first demonstrate the performance of the designed learning module in solving (P1). The weighting factor λ in (4) is set as 0.9 to balance computation load more than controlling offloading decision switching. The total reward averaged over 1000 time steps in each episode is shown in Fig. 2(a). It can be seen that the average reward at the end of the learning process (at around 16000 episodes) converges, where no offloading latency violation and computation capacity overload are incurred. As the learning process goes, all the agents cooperatively train their DQN parameters to refine individual task offloading policies based on local observations to eventually obtain a joint optimal stationary policy.



(a) The MA-DQL performance

(b) Computation load balancing among BSs

(c) Average task offloading switching times

Fig. 2: Performance of the proposed learning-based task offloading framework.

Fig. 2(b) and Fig. 2(c) show the maximum BS computation load (including its average over time) and the average offloading switching times over all BSs per time step, respectively, when the learning convergence is reached (For display clarity, we only show the computation load in the first 100 steps from last training episode). We demonstrate in Fig. 2(b) the cases of maximizing the load balancing and minimizing the offloading switching cost, by setting λ as 0.9999 and 0.0001, respectively. The former case shows better load balancing by sacrificing more task offloading decision changes over consecutive slots in different road zones. An upward trend is observed in Fig. 2(c) on average task offloading switching times per step with the increase of λ. However, the proposed framework balances the BS computation load and task offloading switching cost by well controlling the switching times. A reinforcement learning-based rate maximization scheme in [16] is compared with, as shown in Figs. 2(b) and 2(c). By offloading tasks to a BS with a higher uplink transmission rate, the existing scheme sacrifices more task offloading switching times and certain degree of load balancing to achieve a lower task transmission delay.

VI. Conclusion

In this paper, we have studied a computing task offloading problem in a C-AVN, with the objective of achieving long-term computation load balancing with reduced offloading decision switching cost. Specifically, the problem is formulated as an MDP with per-slot computation capacity and offloading latency constraints. To deal with large state-action spaces and unavailability of state transition probability distribution, we design a multi-agent DQL module where each SBS acts as a learning agent to cooperatively learn a joint optimal task offloading policy based on local observations. To facilitate the learning convergence, a fingerprint-based method is adopted to customize the observation of each agent with state and policy abstraction from other agents. Simulation results demonstrate that the proposed learning-based task offloading framework flexibly balances the BS computation load with controlled offloading decision switching times, compared with a rate-maximization offloading scheme.

Acknowledgment

This work was supported in part by research grants from the Natural Sciences and Engineering Research Council (NSERC) of Canada and in part by the Faculty Research Grant (FRG) 211597 at Minnesota State University, Mankato.

References

[1] S.-C. Lin *et al.*, "The architectural implications of autonomous driving: Constraints and acceleration," in *Proc. ASPLOS' 18*, 2018, pp. 751–766.
[2] W. Zhuang *et al.*, "SDN/NFV-empowered future IoV with enhanced communication, computing, and caching," *Proc. IEEE*, vol. 108, no. 2, pp. 274–291, 2020.
[3] R. Molina-Masegosa and J. Gozalvez, "LTE-V for sidelink 5G V2X vehicular communications: A new 5G technology for short-range vehicle-to-everything communications," *IEEE Veh. Technol. Mag.*, vol. 12, no. 4, pp. 30–39, 2017.
[4] Y. Liu *et al.*, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11 158–11 168, 2019.
[5] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, 2016.
[6] J. Yan *et al.*, "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 235–250, 2020.
[7] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, 2019.
[8] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane, and Y. Liu, "Multi-UAV-enabled load-balance mobile-edge computing for IoT networks," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6898–6908, 2020.
[9] J. Foerster *et al.*, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proc. ICML' 17*, vol. 70, 2017, pp. 1146–1155.
[10] L. Liang, H. Ye, and G. Y. Li, "Spectrum sharing in vehicular networks based on multi-agent reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2282–2292, 2019.
[11] M. Kourtis *et al.*, "A cloud-enabled small cell architecture in 5G networks for broadcast/multicast services," *IEEE Trans. Broadcast.*, vol. 65, no. 2, pp. 414–424, 2019.
[12] H. He, H. Shan, A. Huang, Q. Ye, and W. Zhuang, "Edge-aided computing and transmission scheduling for LTE-U-enabled IoT," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 7881–7896, 2020.
[13] K. Qu, W. Zhuang, X. Shen, X. Li, and J. Rao, "Dynamic resource scaling for VNF over nonstationary traffic: A learning approach," *IEEE Trans. Cogn. Commun. Netw.*, to appear, doi:10.1109/TCCN.2020.3018157.
[14] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[15] "SUMO 1.7.0," [Online]. Available: https://www.eclipse.org/sumo/.
[16] Z. Li, C. Wang, and C. Jiang, "User association for load balancing in vehicular networks: An online reinforcement learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 8, pp. 2217–2228, 2017.