2017 15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)

Deep Reinforcement Learning-based Scheduling for Roadside Communication Networks

Ribal Atallah, Chadi Assi, and Maurice Khabbaz

***Abstract*—The proper design of a vehicular network is the key expeditor for establishing an efficient Intelligent Transportation System, which enables diverse applications associated with traffic safety, traffic efficiency, and the entertainment of commuting passengers. In this paper, we address both safety and Quality-of-Service (QoS) concerns in a green Vehicle-to-Infrastructure communication scenario. Using the recent advances in training deep neural networks, we present a deep reinforcement learning model, namely deep Q-network, that learns an energy-efficient scheduling policy from high-dimensional inputs corresponding to the characteristics and requirements of vehicles residing within a RoadSide Unit's (RSU) communication range. The realized policy serves to extend the lifetime of the battery-powered RSU while promoting a safe environment that meets acceptable QoS levels. Our presented deep reinforcement learning model is found to outperform both random and greedy scheduling benchmarks.**

***Index Terms*—Optimization, VANETs, Energy-efficient, Deep Reinforcement Learning**

I. Introduction

A. Preliminaries:

The road traffic crashes and consequent injuries and fatalities, traditionally regarded as random and unavoidable accidents, are recently recognized as a preventable public health problem. Indeed, as more countries (*e.g.* USA and Canada) are taking remarkable measures to improve their road safety situation, the downward trend for the number of fatalities and serious injuries due to motor vehicle crashes continues, dropping between 7 and 10% yearly between 2010 and 2014 [1]. Although progress in the transportation industry is being made to get us to a safe, more sustainable and more comfortable transport, there still exists substantial challenges that need to be addressed for the purpose of establishing a full-fledged Intelligent Transportation System (ITS). Consequently, researchers and policy makers joined forces in order to realize a fully connected vehicular network that will help prevent accidents, facilitate eco-friendly driving, and provide more accurate real-time traffic information.

Today, Vehicular Ad-Hoc Networks (VANETs) offer a promising way to achieve this goal. VANETs support two types of communications; namely, *a)* Vehicle-to-Vehicle (V2V) communications where messages are transmitted between neighbouring vehicles, and *b)* Vehicle-to-Infrastructure (V2I) communications where messages are transmitted between vehicles and Road-Side Units (RSUs) deployed along side the roads. VANETs highly rely on real-time information gathered from sensing vehicles in order to promote safety in an ITS. Such information is particularly delay sensitive, and its rapid delivery to a large number of contiguous vehicles can decrease the number of accidents by 80% [2].

Certainly, VANET communications offer safety related services such as road accident alerting, traffic jam broadcast, and road condition warnings. However, on the other hand, through V2I communications, mobile users are able to obtain a number of non-safety Internet services such as web browsing, video streaming, file downloading, and online gaming. As such, a multi-objective RSU scheduling problem arises whose aim is to meet the diverse QoS requirements of various non-safety applications while preserving a safe driving environment. At this point, it is important to mention that the unavailability of a power-grid connection and the highly elevated cost of equipping RSUs with a permanent power source set a crucial barrier to the operation of a vehicular network.

Indeed, it has been reported that energy consumption of mobile networks is growing at a staggering rate [3]. The U.S. Department of Energy is actively engaged in working with industry, researchers, and governmental sector partners through the National Renewable Energy Laboratory (NERL) in order to provide effective measures to reduce the energy use, emissions, and overall transportation system efficiency [4]. Furthermore, from the operators' perspective, energy efficiency not only has great ecological benefits, but also has significant economic benefits because of the large electricity bill resulting from the huge energy consumption of a wireless base station [5]. Following the emerging need for energy-efficient wireless communications as well as the fact that grid-power connection is sometimes unavailable for RSUs, [6], it becomes more desirable to equip the RSUs with large batteries rechargeable through renewable energy sources such as solar and wind power [7] and [8]. Hence, it becomes remarkably necessary to schedule the RSUs' operation in such a way that efficiently exploits the available energy and extends the lifetime of the underlying vehicular network.

B. Motivation:

This current work focuses on a V2I communication scenario where vehicles have non-safety download requests to be served by a battery-powered RSU. The objective of this paper is to realize an efficient RSU scheduling policy that meets acceptable QoS levels, preserves the battery power and prolongs the network's lifetime while prioritizing the safety of the driving environment.

In a previous study presented in [9], the authors developed a Markov Decision Process (MDP) framework with discretized

R. Atallah and C. Assi are with CIISE at Concordia University, Montreal, Canada. E-Mail Addresses: {*r.atalah@gmail.com*, *assi@ciise.concordia.ca*}, M. Khabbaz is with the ECCE department at Notre-Dame University, Shouf, Lebanon. E-Mail Address: *mkhabbaz*@ndu.edu.lb.

978-3-901882-90-6/17 ©2017 IFIP

states in order to establish an optimal RSU-controlled scheduling policy whose objective was to satisfy the maximum number of vehicle-generated download requests. Therein, the resolution of the MDP was realized using reinforcement learning techniques [10]. However, Q-learning with discretized states and actions scales poorly [11]. In this present work, the state space is continuous. Hence, classical reinforcement learning techniques are no longer feasible in this case. Recent advances in training deep neural networks are exploited herein in order to characterize the system state and, thus, promote the feasibility of instantiating a fictitious artificial agent that will: *a)* learn a scheduling policy from high-dimensional continuous inputs using end-to-end deep reinforcement learning, *b)* derive efficient representations of the environment, and *c)* progress towards the realization of a successful RSU scheduling policy which meets multiple objectives.

C. Problem Statement and Novel Contributions:

This paper proposes the exploitation of a deep reinforcement learning technique, namely Deep Q-Network, for the purpose of efficiently utilizing the available RSU energy while promoting a safe driving environment and offering acceptable QoS levels. A battery-powered RSU is privileged with a deep learning agent which will observe the vehicular environment and steadily learn the optimal scheduling policy. The need for a safe, smart, and green vehicular environment motivates the establishment of an intelligent scheduling policy which meets multiple objectives.

The following points highlight the identifying contributions of this paper:

1) Unlike the work presented in [12]–[16], this work realizes a scheduling policy, which recognizes that the RSU is equipped with a limited-lifetime power source. A deep reinforcement learning agent is deployed at the level of the energy-limited RSU, which first performs random scheduling decisions and gradually learns an adaptive dynamic policy that serves to extend the battery lifetime, minimize the reception latency of safety messages, and achieve acceptable QoS levels.
2) The work presented in [17] proposed energy-efficient online optimization problems which are solved whenever the characteristics of the vehicular environment changes. The usability of this technique is limited given the highly mobile facet of a vehicular network and the frequent topology changes. In fact, the resolution of a complex optimization problem results in further delaying the decision of the RSU. However, with the use of a deep learning model, and once the optimal policy is established, the RSU performs optimal scheduling decisions instantly without the need to continuously solve time-consuming and complex optimization equations.

To the best of our knowledge, this paper is the first to investigate the feasibility of deep reinforcement learning methods, particularly, deep Q-networks, in a vehicular environment.

D. Paper Organization:

The remainder of this paper is structured as follows. Section II presents a brief overview of the related work. A description of the V2I communication scenario is presented in Section III. Section IV presents the vehicular traffic model. Section V lays out a detailed presentation of the MDP formulation. The deep reinforcement learning framework is presented in Section VI. The performance of the proposed model is examined and compared to other scheduling heuristics in Section VII. Finally, concluding remarks are presented in Section VIII.

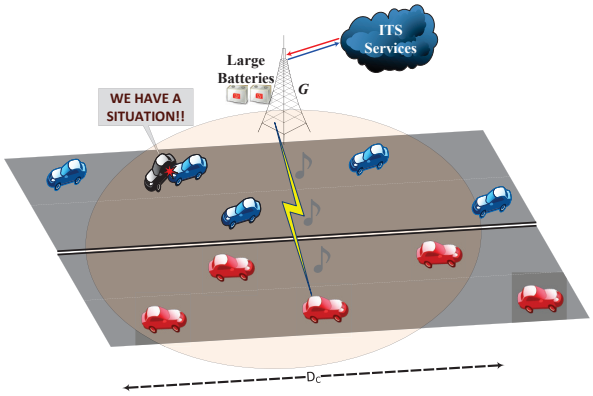


Fig. 1: Energy-Limited VANET

II. Related Work

In [13], the authors proposed a basic low-complexity V2I access scheme called $D*S$ where the RSU stored the Service Requests (SRs) and the request with the least $D*S$ was served first. D is the SR's deadline and S is the data size to be uploaded to the RSU. The authors then studied the uplink MAC performance of a Drive-Thru Internet (DTI) scenario in [14]. Both the contention nature of the uplink and the realistic traffic model were taken into consideration. The authors of [15] proposed two complexity-minimal V2I access schemes and modelled the vehicle's on-board unit buffer's queue as an *M*/*G*/1 queueing system and captured the V2I system's performance from a vehicle's perspective. The authors of [16] proposed a vehicular context-aware downlink scheduling algorithms which use the vehicular information such as the vehicles' positions and cumulated bytes received for the purpose of maintaining a high system throughput and achieving fairness for the contending vehicles. The RSU first collected the required information, modified the transmission rate, selects a vehicle and admits it to service. The major shortcoming of the proposed algorithm is the delay overhead required to make a scheduling decision. Furthermore, its applicability is limited to vehicular drive-thru non-safety-oriented scenarios.

The algorithms proposed in [13]–[16] overlooked the RSU energy consumption pertaining to the adopted scheduling discipline. Given the increasing concern over the energy consumption in wireless networks as well as the highly likely unavailability of permanent power sources in vehicular networks, the conventional design approaches may not be feasible to green communications and should be revisited. Recent research studies presented optimization techniques which address problems such as efficient resource allocation as well as energy-efficient scheduling [17]. The formulation of such

models together with their related optimization techniques are characterized by an elevated level of complexity and require high processing power thus inducing significant latency overhead.

It is worthwhile mentioning that, thus far, the literature overlooks the possibility of establishing scheduling policies using machine learning techniques. Particularly, deep reinforcement learning augments the RSU with the ability to observe and analyse the environment, make decisions, learn from past experience, and eventually, perform optimal actions. In fact, recently, the authors of [18] presented the Deep Q-Network (DQN) algorithm which was tested in a challenging framework composed of several Atari games. DQN achieved dramatically better results than earlier approaches and professional human players and showed a robust ability to learn representations from very high-dimensional input. The authors of [18] and [19] exploited an experience replay mechanism ([20]) and a batch reinforcement learning technique ([21]), which made the convergence and stability of the proposed DQN model possible. DQN is foreseen to address the major long-standing challenge of RL by learning to control agents directly from high-dimensional inputs and state spaces. At this level, it becomes necessary to investigate the feasibility of similar techniques in reinforcement learning scenarios where an agent makes decisions that affect the state of the environment. An example is RSU scheduling in a vehicular network.

III. V2I Communication Scenario

As illustrated in Figure 1, RSU *G* is equipped with large batteries that are periodically recharged using energy harvesting techniques (wind/solar) or, when required, human intervention (physical battery recharging or replacing). Each vehicle is equipped with an OnBoard Unit (OBU) through which it communicates with nearby vehicles as well as RSUs. Vehicles arrive to *G*'s communication range with a non-safety-related download service request (*e.g.* email download or media streaming). In the event where a vehicle senses hazardous road conditions (*e.g.,* slippery road or traffic collision), it raises a safety flag in order to notify the RSU about the existence of a safety message, which should be communicated to that RSU as well as other neighbouring vehicles. This current work adopts a slotted Medium Access Control protocol, similar to the IEEE 802.11*p* standard, where time is divided into equal time slots of length τ. However, our proposed scheduling algorithm is centralized as opposed to the distributed nature of the IEEE 802.11*p* protocol. It is assumed that there is perfect synchronization between the RSU and the vehicles residing within that RSU's communication range with the use of a Global Positioning System (GPS). At the beginning of each time slot, the RSU becomes aware of the characteristics of all the vehicles residing within its communication range. Based on the collected information, the RSU signals to a single vehicle the permission to either upload the safety message it is carrying or continue downloading the requested data. In case the RSU chooses to receive a safety message, the RSU notifies the selected vehicle to transmit the carried safety message, and then the RSU will broadcast the received safety message in the next time slot. Note that, vehicles will drop their safety flag in the case where their carried safety message has been communicated to the RSU by another vehicle. During the time the RSU is receiving a safety message, its energy consumption is minimal. However, when serving a vehicle's download service request, the RSU's consumed energy increases as the distance to the receiving vehicle increases. Truly, according to [22], the power consumption increases exponentially as the distance between the transmitter and receiver increases. The next section lays out the vehicular traffic model adopted in this work.

IV. Vehicular Traffic Model

Consider a multi-lane bidirectional highway segment of length D_C as illustrated in Figure 1. This considered segment is assumed to be experiencing steady free-flow traffic. According to [23]–[25], vehicle arrivals to a particular lane l of the considered segment follow a Poisson process with parameter λ_l. Consequently, the overall vehicles' arrival process to the entire segment is also a Poisson process with parameter $\lambda_s = \sum_{l=1}^{L} \lambda_l$, where L is the number of lanes [26]. The per-vehicle speeds are i.i.d. random variables in the range $[V_{\min}; V_{\max}]$. These speeds are drawn from a truncated Normal distribution with average $\overline{V}$ and standard deviation σ_V. It is assumed that vehicles maintain their respective speeds constant during their entire navigation period over the considered roadway segment [24], [25]. Let J_i be a discretized version of the vehicle's residence time within the considered segment. The p.m.f. of J_i has been derived in [25], and is given by:

$$f_{J_i}(j) = \frac{\xi}{2}\left[\text{erf}\left(\frac{\frac{D_C}{j\tau} - \overline{V}}{\sigma_V\sqrt{2}}\right) - \text{erf}\left(\frac{\frac{D_C}{(j-1)\tau} - \overline{V}}{\sigma_V\sqrt{2}}\right)\right] \quad (1)$$

where $J_{\min} \leq j \leq J_{\max}$, τ is the duration of a time slot and ξ is a normalization constant.

An arriving vehicle communicates its speed, direction of travel, and download requirements as soon as it enters the coverage range of the RSU *G*. Consequently, *G* keeps record of all vehicles within its range as well as their associated service requirements. In this work, a vehicle i's download service request size is a uniformly distributed Random Variable H_i between H_{min} and H_{max}. H_i is expressed in bits.

V. Markov Decision Process Model

In this work, a deep reinforcement learning agent deployed at the RSU interacts with the vehicular environment in a sequence of actions, observations, and rewards. At each time step, the agent selects an action from the set of feasible actions at that time, and correspondingly, the RSU will either receive an announced safety message, or transmit data to a vehicle with a download service request. The agent then observes the changes in the environment and receives a reward accordingly. The received reward depends on the whole previous sequence of actions and observations. As such, the impact of an action may only be seen after several hundreds/thousands of time-steps ahead. This current section is dedicated to present

the system state representation as well as the rewards/costs associated with the agent's actions.

A. Input From the Environment:

At the beginning of an arbitrary time slot (time t_n), the agent observes the vehicular environment and collects all the parameters that define the system state. The agent's input from the environment at time t_n is:

- T_n: the time elapsed since the last RSU battery recharge.
- P_n: the remaining power in the RSU's battery, $0 \leq P_n \leq P_t$, where P_t is the battery capacity.
- N_n: the number of vehicles residing within *G*'s communication range, $0 \leq N_n \leq N_{\max}$.
- $\overline{J_n} = \{J_1^n, J_2^n, \cdots, J_{N_n}^n\}$: a vector of size N_n containing the remaining discrete sojourn times of each vehicle v_i, $i \in (1, 2, \cdots N_n)$ and $0 \leq J_i^n \leq J_{\max}$
- $\overline{H_n} = \{H_1^n, H_2^n, \cdots, H_{N_n}^n\}$: a vector of size N_n containing the remaining request sizes for each vehicle v_i, $0 \leq H_i^n \leq H_{\max}$.
- $\overline{W_n} = \{W_1^n, W_2^n, \cdots, W_{N_n}^n\}$: a vector of size N_n containing the waiting times of the safety messages in the vehicles' buffers. In case vehicle v_i has no safety message to upload, W_i^n is set to a negative value (-1).
- $\overline{d_n} = \{d_1^n, d_2^n, \cdots, d_{N_n}^n\}$: a vector of size N_n containing the separation distances between *G* and each of the in-range vehicles, $0 \leq d_i^n \leq D_C/2$.

The agent fully observes the current network situation and is able to realize the system state representation at time t_n, denoted herein by x_n. The system state x_n is therefore a vector of size $(3 + 4N_n)$. The next subsections will define the set of feasible actions as well as the reward/cost associated with selected actions.

B. Impact of RSU's Action: Rewards and Costs:

Let a_n denote RSU's action at time step t_n. Let A_n be the set of valid (admissible) actions time at t_n given that the system state is x_n, therefore, $a_n \in A_n$. At time t_n, the RSU either chooses to receive a safety message (whose existence has been announced), hence $a_n = 0$, or to transmit data to a vehicle v_i (where $1 \leq i \leq N_n$), and hence $a_n = i$. It is clear that the set of valid actions at time t_n depends on the system state x_n.

Now, at each time-step, the agent selects an action and observes its impact on the environment. The RSU receives a scalar value pertaining to the reward/cost associated with the selected action. At time t_n, whenever the RSU chooses to transmit data to a particular vehicle $v_i \in N_n$, the reward received is the number of transmitted bits during that time step. In this case, the cost paid at time t_n is composed of two components, namely, a) the power consumed by the RSU to serve vehicle v_i, and b) the waiting time of a safety message whenever it exists. When the RSU chooses to listen to an announced safety message, the induced cost pertains to the amount of power required for the RSU to receive the safety message's data. Furthermore, in this model, the event of the departure of a vehicle from the RSU's range with an incomplete download request is considered an undesired event which causes remarkable penalties on the system's returns. Whenever a vehicle departs from the RSU's coverage range with an incomplete download request, the agent is penalized by a value corresponding to the remaining number of bits which need to be downloaded in order to fulfill that vehicle's request. Note that, even if the impact of the occurrence of such event unveils in a single time-step during which a vehicle departs with an Incomplete Service Request (ISR), the agent realizes that the sequence of all its previous actions lead to this event. This is a clear example that the feedback about an action may sometimes by received after many thousands of time steps have elapsed. All of the above is accounted for through the formulation of a large, but finite MDP-based model, whose characteristics are laid out next.

C. MDP Resolution:

The above definitions of the system state, RSU action, and reward/cost gives rise to an MDP whose transition probability kernel is unknown. Therefore, the resolution of this MDP requires the use of Reinforcement Learning techniques for the purpose of maximizing the RSU's total rewards. Let r_n be the single-step RSU reward at t_n, and let R_n be the total future discounted rewards. R_n is given by: $R_n = \sum_{t=t_n}^{T} \gamma^{t-t_n} r_n$, where T is the time step at which the RSU battery is either drained or recharged, indicating the start of a new discharge period.

Recall that the agent's objective is to realize an optimal scheduling policy which will serve to achieve three goals, namely: *a)* minimize the reception delay of safety messages, *b)* maximize the number of completed service requests, and *c)* extend the RSU's battery lifetime. Now, this problem has been formulated as an MDP whose states are modelled as a Markov chain, and a state-action dependent cost is incurred at each stage. Classically, MDPs can be addressed using the methods of dynamic programming [10]. However, in our cases, the state space dimension is enormously large, hence, the required computational effort to realize an optimal RSU scheduling policy is prohibitively large, a phenomenon commonly referred to as the "curse of dimensionality" [27]. As a result, function approximation techniques are used to overcome this widely known limitation of MDPs. In this work, a non-linear function approximator is used, specifically a neural network. The next section lays out the neural network training process using a deep reinforcement learning algorithm, which will serve to realize an optimal policy for the above-presented MDP.

VI. Deep Reinforcement Learning Resolution

A. Reinforcement Learning Background:

The optimal control of a MDP requires the determination of a stationary policy π defining which action a_n should be applied at time t_n in order to maximize/minimize an aggregate objective function of the immediate rewards/costs. As such, a policy π induces a stationary mass distribution over the realizations of the stochastic process (x_n, a_n). A sequence

of functions $\pi = \{a_1, a_2, \cdots, a_T\}$, with each $a_n : E \rightarrow A, 1 \leq n \leq T$, is said to be an admissible policy if $a_n \in A_n$, $\forall x_n \in E$. Let Π denote the set of all admissible policies. The goal of this current study is to find an optimal policy π^* which will maximize the total discounted rewards. Whenever the RSU is following the scheduling policy π, the action at t_n is $a_n = \pi(x_n)$. Therefore, the single step reward becomes $r_n = r(x_n, \pi(x_n))$ and the optimal policy is hence given by:

$$\pi^* := \underset{\pi \in \Pi}{\operatorname{argmax}} \quad R_n^{\pi} \tag{2}$$

and

$$R_n^{\pi} = \sum_{t=t_n}^{T} \gamma^{t-t_n} r\left(x_n, \pi(x_n)\right) \tag{3}$$

Let $Q^*(x_n, a_n)$ be the optimal action-value function defined by the maximum expected return when being in state x_n and taking optimal action a_n. $Q^*(x_n, a_n)$ obeys the following Bellman optimality equation [10]:

$$Q^*(x_n, a_n) = \mathbb{E}_{x_{n+1}}\left[r_n + \gamma \max_{a_{n+1}}\Big[Q^*(x_{n+1}, a_{n+1}|x_n, a_n)\Big]\right] \tag{4}$$

Classical reinforcement learning algorithms estimate the action-value function by using the Bellman equation as an iterative update as follows:

$$Q_{n+1}(x_n, a_n) = \mathbb{E}_{x_{n+1}}\left[r_n + \gamma \max_{a_{n+1}}\Big[Q_n(x_{n+1}, a_{n+1}|x_n, a_n)\Big]\right] \tag{5}$$

This technique is known as the value iteration algorithm and it converges to the optimal action-value function, $Q_n \rightarrow Q^*$ as $n \rightarrow \infty$. However, the usability of this approach in this work is impractical since the system state is a large size vector, hence, the state space is far too large to enumerate. As a result, the time needed for the Q-table to converge increases immeasurably. Consequently, it has become common to use function approximation techniques to estimate the action-value function. The next subsection lays out the adoption of a non-linear function approximation method to represent the Q-function using a neural network, which takes the state vector as input and outputs the Q-value for each possible action.

B. Deep Reinforcement Learning Approach

This paper presents a model-free deep reinforcement learning method where the action-value function is represented using a non-linear function approximator, namely a neural network. Let $Q(x_n, a_n; \theta)$ be an approximate action-value function with parameters θ. Hence, $Q(x_n, a_n; \theta) \approx Q^*(x_n, a_n)$. In the literature, a neural network function approximator with weights θ is referred to as a Q-network [18]. A Q-network can be trained in order to learn the parameters θ of the action-value function $Q(x_n, a_n; \theta)$ by minimizing a sequence of loss functions, where the i^{th} loss function $L_i(\theta_i)$ is given by:

$$L_i(\theta_i) = \mathbb{E}\left[r_n + \max_{a_{n+1}} Q(x_{n+1}, a_{n+1}; \theta_{i-1}) - Q(x_n, a_n; \theta_i)\right]^2 \tag{6}$$

Note that, θ_i are the neural network's parameters at the i^{th} update, and the parameters from the previous update, θ_{i-1} are held fixed when optimizing the loss function $L_i(\theta_i)$. Thus, the term $r_n + \max_{a_{n+1}} Q(x_{n+1}, a_{n+1}; \theta_{i-1})$ is the target for iteration i, which depends on the neural network's parameters from the last update. Differentiating the loss function with respect to the neural network's parameters at iteration i, θ_i gives the following gradient:

$$\bigtriangledown_{\theta_i} L_i(\theta_i) = \mathbb{E}\left[\left(r_n + \max_{a_{n+1}} Q(x_{n+1}, a_{n+1}; \theta_{i-1}) - Q(x_n, a_n; \theta_i)\right) \bigtriangledown_{\theta_i} Q(x_n, a_n; \theta_i)\right] \tag{7}$$

The use of batch methods, which utilize the full training set to compute the next update to parameters at each iteration tend to converge very well to local optima. However, often in practice, computing the cost and gradient for the entire training set is extremely slow and sometimes intractable on a single machine, especially when the training dataset is large. Therefore, rather than computing the full expectations in the above gradient equation, it is computationally desirable to optimize the loss function using the Stochastic Gradient Descent (SGD) method. Based on the learning rate α, SGD updates the neural network's parameters after seeing only a single or a few training examples. The use of SGD in the neural network setting is motivated by the high cost of running back propagation over the full training set. Reinforcement learning methods tend to diverge when used with non-linear function approximators such as a neural network. In order to avoid the divergence of deep reinforcement learning algorithms, two techniques were introduced in [18], namely *a)* Experience Replay and *b)* Fixed Target Network. Theses techniques are utilized herein, and as a result, deep reinforcement learning is used to minimize the following loss function:

$$L_i(\theta_i) = \mathbb{E}_{x_n, a_n, r_n, x_{n+1} \sim \mathbb{D}}\left[r_n + \max_{a_{n+1}} Q(x_{n+1}, a_{n+1}; \theta^-) - Q(x_n, a_n; \theta_i)\right]^2 \tag{8}$$

where $\mathbb{D}$ is the experience replay memory and θ^- are the parameters of the target Q-network. By using the above two techniques, the convergence of the underlying deep reinforcement learning algorithm has been empirically proven in [18] and [19]. On the other hand, the drawback of using the experience replay is the substantial memory requirements.

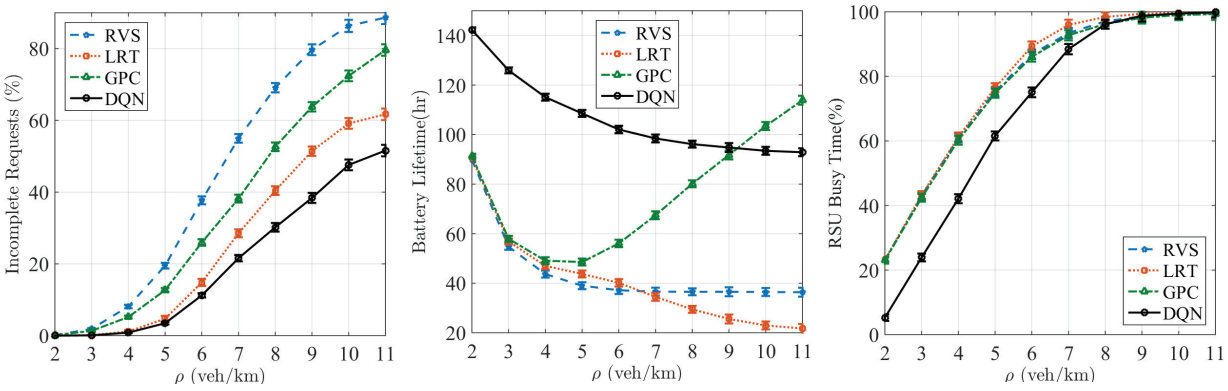
The proposed algorithm herein is an off-policy algorithm as it learns an optimal action $a_n = \max_a Q(x_n, a_n; \theta)$ while still choosing random actions to ensure adequate exploration of the state space. The widely used technique for off-policy algorithms is the ϵ-greedy strategy where the agent selects the current optimal action with a probability $1 - \epsilon$, and selects a random action with probability ϵ. The deep Q-learning algorithm is presented in Algorithm 1.

VII. Results and Discussion

A. Simulation Setup:

In the simulation setup of this work, the vehicular traffic model presented in Section V is adopted. Realistic mobility traces were generated by SUMO (Simulation for Urban MObility) and fed as a mobility input for the simulation. The presented results herein were averaged over multiple runs of the simulations. Table I lists the simulator's input parameters. In this section, the performance of the proposed DQN algorithm is evaluated in terms of:

- Incomplete request percentage.



(a) Incomplete Request Percentage (b) Battery Lifetime (c) RSU Busy Time

Fig. 2: Performance Evaluation with Variable Vehicular Density

Algorithm 1 Deep Q-Learning with Experience Replay and Fixed Target Network

Initialize replay memory $\mathbb{D}$ to capacity $\boldsymbol{C}$
Initialize Q-network with random weights $\boldsymbol{\theta}$
Initialize target Q-network with random weights $\boldsymbol{\theta^- = \theta}$
for episode = 1, M **do**
  Collect network characteristics to realize state $\boldsymbol{x_0}$
  for n = 0, T **do**
    $\boldsymbol{a_n} = \text{argmax}_a\ \boldsymbol{Q(x_n, a_n; \theta)}$ with probability $\boldsymbol{1-\epsilon}$.
    Otherwise, $\boldsymbol{a_n}$ is selected randomly.
    Execute $\boldsymbol{a_n}$ and observe $\boldsymbol{r_n}$ and $\boldsymbol{x_{n+1}}$
    Store transition $\boldsymbol{(x_n, a_n, r_n, x_{n+1})}$ in $\mathbb{D}$
    Sample random minibatch of transitions from $\mathbb{D}$
    Set the target to $\boldsymbol{r_n}$ if episode terminates at $\boldsymbol{n+1}$, otherwise, target is $\boldsymbol{r_n + \max_{a_{n+1}} Q(x_{n+1}, a_{n+1}; \theta^-)}$
    Perform a SGD update on $\boldsymbol{\theta}$
    Every $\boldsymbol{C}$ steps, update: $\boldsymbol{\theta^- = \theta}$
  end for
end for

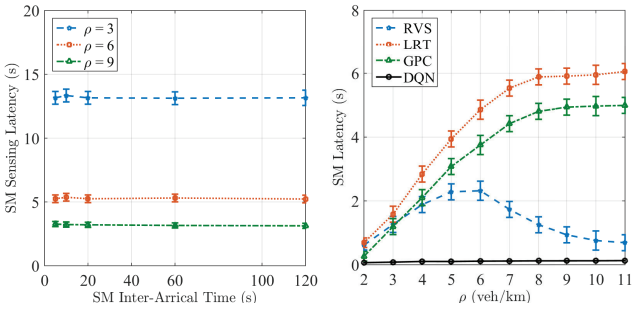


Fig. 3: SM Sensing and Receiving Latencies

- RSU battery lifetime.
- RSU busy time.

The proposed algorithm herein is compared with three other scheduling algorithms namely:

1) RVS: Random Vehicle Selection algorithm where, at time t_n, the RSU randomly chooses a vehicle $v_i \in I_n$ to be served [15].
2) LRT: Least Residual Time algorithm where, at time t_n, the RSU chooses the vehicle $v_i \in I_n$ whose remaining sojourn time $J_i^n < J_j^n$, $\forall j = 1, \cdots N_n$ and $j \neq i$ [15].
3) GPC: Greedy Power Conservation algorithm where, at time t_n, the RSU chooses the closest vehicle $v_i \in I_n$ which contributes to the lowest energy consumption compared to the remaining vehicles residing within G's communication range.

Under all the above scheduling algorithms, if the selected vehicle happens to carry a safety message, the vehicle will transmit it to the RSU, which will, in turn, broadcast it to the set of in range vehicles.

TABLE I: Simulation Input Parameters

Parameter	Value
RSU Battery Capacity	$5 \times 50Ah$ batteries
Time slot length	$\tau = 0.1$ (s)
Vehicular densities	$\rho \in [2; 11]$ (veh/km)
Min and Max vehicle speed	$[60; 140]$ (km/h)
Min and Max request size	$[2; 10]$ (MB)
RSU covered segment	$D_C = 1000$ (m)
Vehicles and RSU radio range	500 (m)
Channel data bit rate	$B_c = 9$ (Mbps)
Learning rate	$\alpha(n) = 1/n$
Discount factor	$\gamma = 0.5$
Replay Memory Capacity	1 million transitions
Minibatch Size	100 transitions

B. Simulation Results:

Figure 2 evaluates the performance of the proposed DQN algorithm when compared with the three previously described scheduling algorithms, namely, RVS, LRT and GPC. Figure 2(a) plots the percentage of vehicles leaving G's communication range with an incomplete service request as a function of the vehicular density. It is clear that the percentage of incomplete requests increases as more vehicles are present within the RSU's coverage range under all scheduling algorithms. In fact, as ρ increases, the likelihood of selecting a

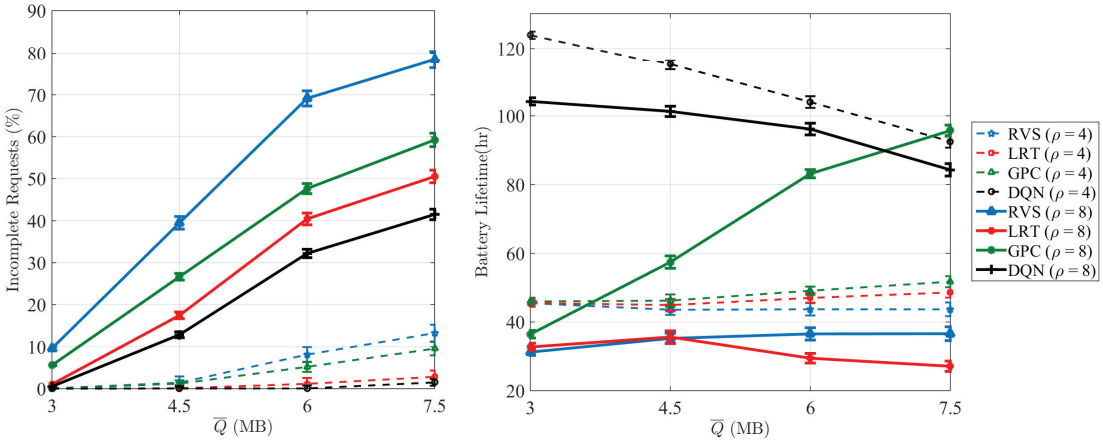


Fig. 4: Performance Evaluation with Variable Average Request Size

certain vehicle will decrease, independent of the scheduling discipline. Consequently, a vehicle will spend less time receiving service and the total number of vehicles departing from G's communication range with incomplete service requests will increase. Figure 2(a) also shows that DQN outperforms RVS, LRT as well as GPC in terms of incomplete service requests. Under RVS, the selection method is random, and no service differentiation is applied, and therefore, the number of vehicles whose associated download request is not fulfilled increases remarkably as more vehicles are present within G's communication range. Now, for GPC, G is serving the closest vehicle which resides in the minimal energy consumption zone compared to the set of in range vehicles. Whenever ρ is small, a large portion of the vehicles have enough time to complete their download request during the time they are the closest to the G. However, when ρ increases, the time during which a certain vehicle is closest to the RSU is not enough to complete the download request. As a result, the percentage of vehicles departing with an incomplete service request increases. Under LRT, the vehicle with the least remaining download residence time is selected. Whenever the vehicular load is small, (*i.e.,* $\rho < 6\ veh/km$), LRT performs relatively well. In fact, under LRT, a vehicle having the least residual residence time will be granted a prioritized continuous access to the channel allowing it to download its entire service request. Under LRT, similar to GPC, when the traffic load increases, the time a vehicle is prioritized for channel access is not enough to completely download its service request. Finally, the well-trained DQN scheduling algorithm results in a smaller number of vehicles with incomplete service requests compared to RVS, LRT and GPC. This is especially true since the departure of a vehicle with an incomplete service request is considered an undesirable event, and during the exploration phase, the DQN algorithm learns to prevent the occurrence of such events in order to avoid penalizing its total rewards.

Figure 2(b) plots the battery lifetime when the RSU is operating under different scheduling algorithms. Under RVS, the battery lifetime decreases as more vehicles are present within the RSU's coverage range. Now, as the vehicular density further increases, the RSU battery lifetime becomes constant. This is due to the fact that, under RVS, the RSU becomes continually busy (as illustrated in Figure 2(c)), and randomly chooses a vehicle to serve without accounting for the energy consumption. Under LRT, the battery lifetime decreases dramatically as ρ increases. In fact, as more vehicles are present within G's coverage range, the vehicle with the least residual residence time is most probably located at the edge of the departure point from the RSU's communication range. That point is the farthest from G and requires the highest amount of energy from the RSU to serve that distant vehicle. Now, under GPC, the battery lifetime decreases as ρ increases from 2 to 5 vehicles per meter, and then increases as ρ increases. Note that, under GPC, G serves the closest vehicle. So when ρ increases from 2 to 5, the RSU becomes busier (as illustrated in Figure 2(c)), and since there is a small number of vehicles within the RSU's range, the closest vehicle with an incomplete service request may not reside in low energy consumption zones, causing the RSU to consume larger amounts of energy. As such, the battery lifetime decreases. Now, as ρ increases further, more vehicles reside within the RSU's coverage range, and the closest vehicle to the RSU resides in lower energy consumption zones, allowing G to consume less amounts of energy and extend its battery lifetime. Finally, under DQN, as ρ increases and the RSU becomes busier, the battery lifetime decreases. It is clear that DQN outperforms RVS and LRT in terms of battery lifetime. DQN also shows longer battery lifetime than GPC whenever the vehicular density is less than 9 vehicles per km. When ρ increases beyond 9 veh/km, GPC results in longer battery lifetime. This is due to the fact that DQN not only tries to spend the least amount of energy, but also, to serve as much vehicles as possible. It is true that GPC outperforms DQN in terms of battery lifetime as the traffic conditions experience larger vehicular densities, however, this is at the expense of deteriorated QoS levels revealed by the large percentage of incomplete service requests.

Figure 3 plots the Safety Message (SM) sensing and receiving delays. The sensing delay is the time it takes any vehicle residing within G's communication range to sense the existence of a SM. The receiving delay is the time elapsed from the first sensing of the SM until it is disseminated in the

network by the RSU. The SM sensing delay is independent from the RSU scheduling discipline. In fact, SM sensing delay depends on traffic density as well as the vehicles' sensing range. Now, it is clear from Figure 3(a) that the sensing delay is also independent from the SM inter-arrival time, but decreases as the vehicular density increases. As more vehicles are present within G's coverage range, it becomes more likely that a vehicle is close to location of the hazardous event, and consequently, the SM sensing delay decreases. Figure 3(b) plots the SM receiving delay, and shows that, the three scheduling algorithms RVS, LRT and GPC do not account for the existence of a safety-related message in the network. Under RVS, the SM receiving delay increases at first since the chances of choosing a vehicle that carries a SM are small. However, this delay decreases again as more vehicles reside in G's communication range and more vehicles have sensed the existence of a SM and raised a safety flag. Under LRT, a vehicle senses a SM and waits until it becomes the vehicle with the least remaining residence time in order to transmit that SM to the RSU, which will, in turn broadcast it to the set of in range vehicles. This results in high SM receiving delays as illustrated in Figure 3(b). Similarly under GPC, a vehicle raising a safety flag has to wait until it becomes the closest to the RSU in order to transmit the SM it is holding. Under DQN, the algorithm realizes the significant importance of safety messages, and as such, whenever a vehicle raises a safety flag, the RSU listens to the carried SM immediately and broadcasts it to the set of in range vehicles thereafter, hence promoting a safer driving environment.

Figure 4 plots the percentage of incomplete requests and the RSU battery lifetime versus the average request size. Figure 4(a) shows that the percentage of vehicles departing from RSU's coverage range with an incomplete service request increases as the average request size increases. The QoS also deteriorates as the vehicular density increases, which emphasizes the result in Figure 2(a). It is clear that DQN outperforms all the other scheduling benchmarks irrespective of the size of the average service request. Figure 4(b) shows that the RSU battery is conserved for longer periods when the RSU is operating under DQN. Similar to the reasoning of Figure 2(b), GPC outperforms DQN in terms of battery lifetime only in situations where either the traffic conditions are light-medium or the average request size is large enough in order to keep serving vehicles in low energy consumption zones. In both situations, GPC is resulting in remarkable degraded QoS levels.

VIII. Conclusion and Future Research Direction

This paper develops an artificial agent deployed at the RSU, which will learn a scheduling policy from high-dimensional continuous inputs using end-to-end deep reinforcement learning. This agent derives efficient representations of the environment, learn from past experience, and progress towards the realization of a successful scheduling policy in order to establish a green and safe vehicular network which achieves acceptable levels of QoS. This work is the first step towards the realization of an artificial intelligent agent, which exploits deep reinforcement learning techniques and governs a set of RSUs connected in tandem on a long road segment in order to promote an efficient and connected green vehicular network.

References

[1] T. Canada, "Canadian motor vehicle traffic collision statistics 2014 - http://www.tc.gc.ca."
[2] S. Pierce, "Vehicle-infrastructure integration (vii) initiative: Benefit-cost analysis: Pre-testing estimates," March 2007.
[3] E. Strinati *et al.*, "Holistic approach for future energy efficient cellular networks," *e & i Elektrotechnik und Informationstechnik*, vol. 127, no. 11, 2010.
[4] U. D. of Energy NREL, "Transportation and the future of dynamic mobility systems," 2016.
[5] D. Lister, "An operators view on green radio," *Keynote Speech, GreenComm*, 2009.
[6] K. Tweed, "Why cellular towers in developing nations are making the move to solar power," *Scientific American*, 2013.
[7] V. Chamola *et al.*, "Solar powered cellular base stations: current scenario, issues and proposed solutions," *IEEE Communications Magazine*, vol. 54, no. 5, 2016.
[8] R. Atallah *et al.*, "Energy harvesting in vehicular networks: A contemporary survey," *IEEE WCM*, vol. PP(99), 2015.
[9] R. Atallah *et al.*, "A reinforcement learning technique for optimizing downlink scheduling in an energy-limited vehicular network," *IEEE Transactions on Vehicular Technology*, 2016.
[10] M. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
[11] C. Gaskett *et al.*, "Q-learning in continuous state and action spaces," in *Australasian Joint Conference on Artificial Intelligence*, Springer, 1999.
[12] M. Cheung *et al.*, "Dora: Dynamic optimal random access for vehicle-to-roadside communications," *IEEE JSAC*, vol. 30, no. 4, 2012.
[13] Y. Zhang *et al.*, "On scheduling vehicle-roadside data access," in *Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks*, pp. 9–18, ACM, 2007.
[14] Y. Zhuang *et al.*, "On the uplink mac performance of a drive-thru internet," *Vehicular Technology, IEEE Transactions on*, vol. 61, no. 4, pp. 1925–1935, 2012.
[15] R. Atallah *et al.*, "Modeling and performance analysis of medium access control schemes for drive-thru internet access provisioning systems," *IEEE T-ITS*, vol. 16, no. 6, 2015.
[16] T. Hui *et al.*, "Vecads: Vehicular context-aware downstream scheduling for drive-thru internet," in *Vehicular Technology Conference (VTC Fall), 2012 IEEE*, pp. 1–6, IEEE, 2012.
[17] A. Hammad *et al.*, "Downlink traffic scheduling in green vehicular roadside infrastructure," *IEEE TVT*, vol. 62, no. 3, 2013.
[18] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
[19] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[20] J. L. Lin, "Programming robots using reinforcement learning and teaching.," in *AAAI*, pp. 781–786, 1991.
[21] S. Lange *et al.*, "Batch reinforcement learning," in *Reinforcement Learning*, pp. 45–73, Springer, 2012.
[22] T. Rappaport *et al.*, *Wireless communications: principles and practice*, vol. 2. Prentice Hall PTR New Jersey, 1996.
[23] S. Yousefi *et al.*, "Analytical model for connectivity in vehicular ad hoc networks," *IEEE TVT*, vol. 57, no. 6, 2008.
[24] M. Khabazian *et al.*, "A performance modeling of connectivity in vehicular ad hoc networks," *IEEE TVT*, vol. 57, no. 4, 2008.
[25] M. Khabbaz *et al.*, "A simple free-flow traffic model for vehicular intermittently connected networks," *IEEE T-ITS*, vol. 13, no. 3, 2012.
[26] E. Cascetta, *Transportation systems engineering: theory and methods*, vol. 49. Springer Science & Business Media, 2013.
[27] W. Powell, "What you should know about approximate dynamic programming," *Naval Research Logistics (NRL)*, vol. 56, no. 3, 2009.