Received April 12, 2020, accepted April 29, 2020, date of publication May 26, 2020, date of current version June 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2997729

Virtual Relay Selection in LTE-V: A Deep Reinforcement Learning Approach to Heterogeneous Data

XUNSHENG DU[1], (Graduate Student Member, IEEE), HIEN VAN NGUYEN[1], (Member, IEEE), CHUNXIAO JIANG[2], (Senior Member, IEEE), YONG LI[3], (Senior Member, IEEE), F. RICHARD YU[4], (Fellow, IEEE), AND ZHU HAN[1], (Fellow, IEEE)

[1]Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004, USA
[2]Tsinghua Space Center, Tsinghua University, Beijing 100084, China
[3]Department of Electronic Engineering, Tsinghua University, Beijing100084, China
[4]Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada

Corresponding author: Xunsheng Du (xunshengdu@gmail.com)

This work was supported in part by US MURI AFOSR under Grant MURI 18RT0073, and in part by NSF under Grant EARS-1839818, Grant CNS1717454, Grant CNS-1731424, and Grant CNS-1702850.

ABSTRACT The development of Long Term Evolution (LTE) enables wireless communication with high transmission rate, low latency, and wide coverage area. These outstanding features of LTE support the next generation of vehicle-to-everything (V2X) communication, which is named LTE-V. Among the various technologies in LTE-V, placing relay nodes on vehicles is a promising approach to save power and energy, and extend the transmission range. In this paper, we consider the virtual relay node selection problem. In the problem, a base station transmits data to a vehicle relay (also known as helper) who will further disseminate the received data to the vehicular subscribers nearby. The selection of the vehicle relay node is a challenging issue since the utility of the selection can only be known after this action has been made. Another challenge of this problem is that the traditional pure optimization is inapplicable due to the imperfect information available. Motivated by the recent success of Alpha Go Zero, we employ deep reinforcement learning (DRL) as a powerful tool for facing the above challenges and solving the problem without global information. We build a bridge between the traffic information and decision of relay node selection based on the reality that the utility of vehicle relay is highly correlated with the traffic density. In our work, deep convolutional neural networks are first applied to extract traffic patterns and then learn the traffic and network topology. Deep learning (DL) acts as a role to map features inside traffic and communication topology to the decisions. Then Q-Learning dynamically updates the utilities of decisions by trials in the environment. Finally, the overall rewards can be calculated to measure these decisions, and the Q function is also updated. Simulation results based on real traffic data validate that our proposed approach can achieve high utility performance.

INDEX TERMS Vehicle communication, LTE-V, deep learning, reinforcement learning.

I. INTRODUCTION

Nowadays, vehicular network communication is a compelling technology to provide wireless connectivity among vehicles, roadsides' drivers, passengers, and pedestrians [1]. There exists a potentially promising market for vehicle-to-everything (V2X) [2], yet it has not been put into large-scale inference [3]. A lot of applications are on their way to be implemented in vehicle-to-vehicle (V2V) communication scenarios, such as road safety, traffic efficiency, and infotainment types with different performance requirements [1]. Most of those applications require low delay, high reliability, and high quality of service (QoS). In order to meet the above service requirements, several wireless access technologies have been conceived to provide radio interface including cellular systems, infrared communications, traditional Wi-Fi, and IEEE 802.11p [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Bohui Wang.

Vehicular ad hoc network (VANET) is a part of the novel approach for intelligent transportation system (ITS) with the aim to enhance driver's safety, regulating traffic and improving the whole driving experience [5]. In real-time traffic scenarios, VANET enables inter-vehicle, vehicle-to-roadside, as well as inter-roadside communication. However, IEEE 802.11p suffers from the unbounded delay problem, scalability issue, and lack of high QoS guarantees [6]. Moreover, the transmission range is limited when it comes to huge urban traffic scenarios. Without pervasive roadside equipment, it is difficult for IEEE 802.11p to provide consecutive and long-lived vehicle-to-infrastructure (V2I) connectivity. The disadvantages of IEEE 802.11p demonstrated above activate the emerging development of LTE-V which is a potential solution supporting vehicular communications [7].

LTE has a wide coverage area, high penetration rate and also supports service of high speed. Moreover, LTE enables high data rates and low latency, which can be beneficial to vehicular safety. As a matter of fact, the safety problem on the road is more and more severe nowadays. According to the National Highway Traffic Safety Administration (NHTSA), there were 47,420 fatal crashed in 2016 [8]. These exiting features of LTE-V give birth to a lot of applications to resolve the safety problem on the road. For example, cooperative awareness message (CAM), which includes basic vehicle data such as speed, position and accelerate speed, can be exchanged among vehicles. There are some user cases for CAM including emergency vehicle warning, slow vehicle indication, intersection collision warning and so on. The other type of messages is called decentralized environmental notification message (DENM), which includes information of emergency electronic brake light and road-work zone warning, wrong-way driving warning, stationary vehicle accident, traffic condition and signal violation warning, etc. Besides those emergency-related messages, LTE-V also delivers delay-tolerant information, such as news, entertaining shows, weather forecast, etc. Apparently, there will be more and more demands for new applications emerging with the development of LTE-V technology. In the literature, there are many works focusing on solving problems in the applications presented above. From the crash warning point of view, in [9], an algorithm for the pre-crash control system was proposed to provide all-round prewarning of a potential accident. For the use case for emergency vehicles, comprehensive design of emergency vehicle warning system was proposed [10], in which vehicles not only receive warning of approaching emergency vehicle, but also are warned of detailed route information. Adaptive cruise control (ACC) is another interesting topic. Reference [11] presented design, development, implementation, and testing of a cooperative adaptive cruise control (CACC) system, which consists of two controllers for managing approaching maneuver to leading vehicles and regulating car-following, respectively. For CACC, another work [12] proposed a reinforcement learning (RL) approach for developing controllers for the secure longitudinal following of a front vehicle.

Those aforementioned works considered the common use cases in the V2X scenario and solve control problems in a relatively small scale. However, due to the development of traffic monitoring technique and internet-of-thing (IoT), more data about the traffic are available. For analyzing such a big amount of data, DL [13] is one of the promising options. DL enables the analysis of global information to obtain the overall picture. Nowadays, the development of DL makes the technique of pattern recognition more powerful. It's natural to use the power of the DL to analyze traffic and wireless communication patterns or topologies in order to extract features inside traffic data. Although, DL provides insights in data, it cannot offer decision making strategy in control-related tasks. RL [14], on the other hand, helps agent interact with environment and make decision based on the supervision of the feedback (measured as rewards) with only partial information known. DRL that combines DL and RL can build a decision making framework, which interacts with ever-changing and random environment. From the aspect of urban wireless data transmission control, a big picture is rather important since the coverage of base station will not be limited to just a few street blocks. Also, sometimes only partial information related to decision making is available, which is impossible for pure optimization. In our work, we consider a relay scenario based on V2V communication, in which a huge urban area is taken into consideration.

The main contributions of our work can be summarized as follows:

1) Based on the highlighted features of LTE-V, such as high bit rate, long range, high capacity and ubiquitous coverage, a vehicular network topology is proposed, which considers both traffic topology and wireless communication topology in the V2X scenario.
2) A bridge is built between traffic topology and delay-tolerant data allocation in V2X communications. We design a scheme which takes traffic topology and wireless communication topology as inputs and produces a data allocation strategy. This scheme use DL as a function approximator to formulate mapping between inputs and outputs. We use Python Streets4MPI toolbox [15] to simulate and generate traffic heat maps, and use Tensorflow as platform to implement deep Q-learning framework.
3) There is no explicit relationship between traffic topology and data allocation in wireless communications. Thus, a novel double deep Q-learning approach [16] is utilized to tackle the formulated problem. The input to our deep Q-learning is heterogeneous data containing traffic and wireless communication information. In order to achieve a better training result, actor and critic networks as well as experience replay are utilized to enhance the performance of traditional DRL. We use cumulative rewards to measure the performance of the deep Q-learning framework and compare it with the performance of other baselines. We also visualize the behavior of the deep Q-learning agent to obtain insight of its intelligence.

The rest of this paper is organized as follows: Section II will introduce more related works concerning the control and optimization in the vehicle relay scenario. Then, our data relay model based on LTE-V is proposed in Section III, and the problem is formulated in Section IV. Based on the formulated problem, a novel DRL approach is introduced to tackle the problem in Section V. In Section VI, we will show the effectiveness of our algorithm by simulation results. Finally, some concluding remarks are given in Section VII.

II. RELATED WORK

There exists some works focusing on the vehicular relay in V2X networks (either vehicle-to-vehicle or vehicle-to-infrastructure) [17]–[24]. In [18], a vehicular relaying technique for enhanced connectivity in densely populated urban areas was designed to investigate the performance of a transmission scheme over a Long-Term Evolution-Advanced (LTE-A) network where vehicles act as relaying cooperating terminals session between a base station and an end-user. Also in urban areas, [17] investigated the impact of the greedy and selfish individual nodes on the cooperation dynamics in vehicular ad-hoc networks. A decentralized self-organized relay selection algorithm based on game theory was proposed in vehicular ad-hoc networks [17]. In cooperative networks scenario, the cooperative secure transmissions in multiple-input signal-out (MISO) vehicular relay networks is studied [20]. Seyfi *et al.* [21] investigates cooperative diversity with relay selection over cascaded Rayleigh fading channels. In [21], the authors conducted a specific analysis on the performance of a relay selection scheme for cooperative vehicular networks with the decode-and-forward (DF) protocol.

From the data forwarding or packet delivering point of view, the assistance of relay for packet delivering dramatically enlarges the coverage area and saves a huge amount of power and energy in vehicular networks. Song and Tao [19] proposed an analytical approach based on stochastic geometry to analyze the location-aware opportunistic V2V relay scheme in terms of the transmission success probability for a target destination vehicle and the connectivity probability when the scheme is applied to inter-connect adjacent RSUs. In [25], the optimal relay station (RS) selection strategy for the vehicular subscriber station (SS) was studied. By using a highway mobility model in IEEE 802.16j MR network, Ge *et al.* formulated a nonlinear optimization problem and figured out the optimal locations of RSs. However, LTE enables a much larger coverage area, which enables a source station to communicate directly with some vehicular relays. In [26], performance evaluation of relay vehicles was addressed. Chai *et al.* formulated an optimal matching problem in a bipartite graph and solved it using the Kuhn-Munkres (K-M) algorithm. Similarly, [27] proposed a game theory approach to tackle the relay vehicle selection problem by jointly considering all the relay vehicles and source vehicles.

As we can tell from the review of previous works on vehicle relay in vehicular networks, most of the works are formulated as a joint optimization problem, for which the non-linear programming or game theory related algorithm can be implemented. However, in most cases in real traffic scenario, only partial information might be known. Meanwhile, data we obtain may only contain implicit information, which is waiting for excavation. Deep neural networks [13] is now a popular and powerful tool to extract features inside data. By traffic monitoring, 2D or even higher dimension images can be generated. Using convolutional neural networks (CNNs) [28] is one of the best choices to analyze the spatial relationship between different data samples. Since the traffic situation changes from time to time, time-correlative data also can be obtained. In this case, recurrent neural networks (RNNs) can show its prowess. In [29], a special type of RNNs, long short-term memory (LSTM) neural networks are utilized to predict long term traffic.

Although DL is powerful in pattern recognition and prediction, it cannot directly help decision making based on those patterns it observes. That is why RL [14] comes onto the stage. RL is a type of machine learning that creates agents which are capable of taking actions in an environment in order to maximize overall rewards. From the aspect of the learning method, DL can be classified into supervised, unsupervised, or semi-supervised learning. On the other hand, RL presents a form of supervision through reward without explicitly tell the agent how to perform the task. DRL [30], [31], which combines DL and RL, is now widely considered a technique that is close to artificial intelligence most. In the traffic scenario, there exist some works utilizing DRL. In [32], a safety-based control of vehicle driving is implemented by using DRL. The continuous control is achieved to let vehicles implement self-driving. In [33], a traffic light control system is achieved by using DRL in traffic simulator SUMO [34]. However, different from the solving control problems in the pure traffic scenario, our work tries to combine wireless communication with traffic in a DRL approach.

III. SYSTEM MODEL

In this section, we introduce the system model in two subsections. In the first subsection, we introduce traffic and wireless communication topology. The representation of the traffic situation and the virtual vehicle relay scheme is introduced. The combination of the traffic and wireless communication topology information is fed into the deep Q-learning framework, which makes the input data heterogeneous. In the second subsection, we go to the details of the communication model we use in our virtual vehicle relay scenario.

A. TRAFFIC AND V2V COMMUNICATION TOPOLOGY

We consider both traffic topology and V2V communication topology. We analyze the traffic topology by using the traffic density map, which also known as the traffic heat map. In our model, we simulate a series of traffic heat maps of a certain district in a city. The transportation traffic simulator we used in our work is named Streets4MPI [15], a python based software that can simulate continuous transportation traffic heat maps. The workflow of the transportation traffic simulation can be briefly summarized as follows:

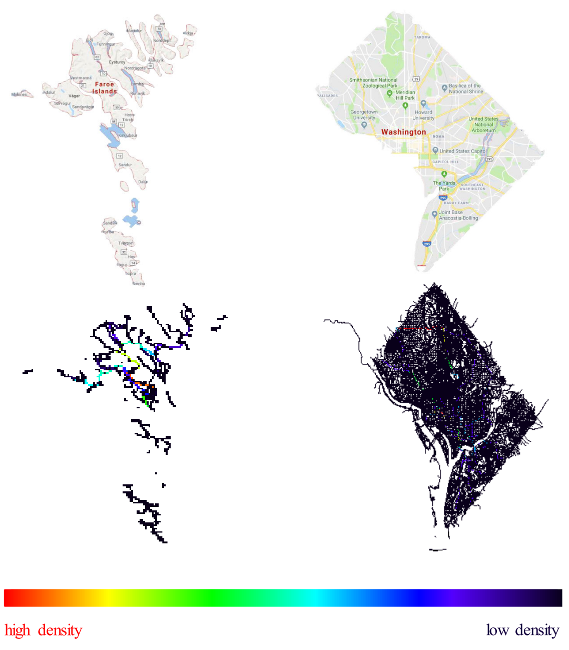


FIGURE 1. Sample traffic heat maps generated by the transportation traffic simulations at a certain timestamp on two locations as examples: the Faroe Islands and the District of Columbia. The images in the first row are generated by Google Map. The second row contains the corresponding traffic heat maps generated by our transportation traffic simulation. The density color bar only shows the relative value of density, and for each map, the scale of the density is different (i.e., Across different maps, a certain RGB value of a pixel may represent a different number of vehicles around it.)

1) A street network of a real city in the world is imported into the canvas. The file representing the structure of the street network is an XML file named OSM XML [35].
2) A number of drivers are initialized (this number can be customized). For each driver, there is a corresponding pre-defined [origin, goal] pair, which indicates the location of the departure and arrival.
3) Each driver will drive his/her car along the shortest path between the departure and arrival. While driving, each vehicle has a speed range. Between two cars, there is a minimum distance to avoid the collision.
4) All calculated shortest paths are then traversed and the traffic load is recorded for each street.
5) Those recorded traffic load can be visualized and represented as pixel values on the traffic heat maps.
6) The simulation can be conducted circularly. The traffic heat maps vary from time to time. If one driver has finished its trip. It will start again from the departure. Thus, the total number of vehicles on the road won't change after it is set up before the simulation begins.

The simulation generates one heat map at a time. We can set the simulation interval so that a series of maps will be generated in sequence with a fixed time interval, which demonstrates the variation of the traffic flow over time. Those sample maps show the traffic in a small city. However, in the communication simulation, we will crop the map and only use a small portion.

We assume that the cache data required by the vehicular end users come from the roadside units, such as eNB, located at the suburb region of the city. The distance between the eNB and the center of our selected urban area is around 1.5 to 2 kilometers. A cache data relay scheme is considered in our model. The eNB first distributes the cache data to a vehicle on road, which is a mobile relay node. Then this vehicle, also known as the helper, disseminates the cache data to the surrounding vehicles within its capability. This kind of transmission schemes with vehicle relay has the advantage of saving unnecessary power consumption and solving traffic scalability issue, while the scheme of distributing cache data to each vehicle individually using cellular broadband access [36] is more energy-consuming. This vehicle relay scheme will be widely made used of in the future LTE-V network. The selection of the vehicle relay node relies on the topology of traffic, which is changing all the time. Since we cannot control the trace of the vehicles, we come up with a virtual relay node selection scheme. This scheme is under the assumption that every vehicle on road can be a potential helper by installing both transmitter and receiver on board, which is also going to be achieved in the near future. Under this assumption, every vehicle on the road is ready to act as a helper. If it happens to pass by the spot where the virtual relay node is placed, the real connection between it and the eNB at the suburb can be built up, and the transmission begins. When this vehicle drives off the virtual relay node, the transmission will be handed over to the next vehicle that drives in this node. Due to the handover problem, we gradually move the virtual relay node from one spot to an adjacent one instead of jumping from one spot to another faraway spot. In this paper, we have no further consideration on the impact of handover on the quality of transmission. After receiving the data from eNB, the helper will disseminate the data to other vehicular users within its opportunistic range [37], which is a term that describes a contact opportunity between the helper and the end user. The virtual relay node will be gradually moved to the spot where the overall system transmission throughput can be achieved.

Along the movement of the virtual relay node, we can draw a trace, named virtual relay trace. This trace is not explicitly related to the transportation traffic density since we cannot control the movement of vehicles. However, as transportation traffic density is one of the important factors which impact the transmission throughput (i.e., how many potential helpers and end users around), it will implicitly influence and lead the virtual relay node to a better spot, and for better transmission performance. It is not necessary for the virtual relay trace to follow the trace of roads. If the virtual relay node is moved off the roads, where there are no vehicles, we can hand over the transmission to roadside units (RSUs) or unmanned aerial vehicles (UAVs), and let them act as the helpers. If the virtual relay trace traverses a consecutive off-road area, the trace of virtual relay node works as an intended path for the UAVs.

B. COMMUNICATION MODEL

For LTE-V, there are two types of messages that are sent among vehicles, which require low latency and high reliability [38]. Recall from Section I, CAM contains some basic vehicle-related data such as speed, position and so on. On the other hand, DENM only works for some emergency situation, e.g., emergency electronic brake light and road-work zone warning.

In our model, we consider all vehicles in traffic can work as a message relay, which is capable of disseminating received data from eNB to nearby vehicles. We consider a contact model that vehicles can communicate with each other only when they move to within the transmission range [37], which is also called communication contact. The Poisson distributed contact rate has been observed in the real vehicular trace and has been implemented in a lot of works, such as [39], [40], which enables analysis for resource allocation and control problems.

Here, we consider the problem in another way that the traffic density and transmission capacity of helpers are two key factors affecting the communication range. Since our observation of the system is a static traffic map of a large region, the demonstration of traffic status is in a more macro way compared to a dynamic vehicle-oriented system using some traffic simulation software such as SUMO [34]. Since the density of every corner in the map is known in an omnipotent view, we can denote the opportunistic contact as $O_{v_e,u} = \alpha \cdot \frac{K_{v_e}}{D_{x_{v_e},y_{v_e}}}$, where K_{v_e} is the transmission capacity of helper v_e, and $D_{x_{v_e},y_{v_e}}$ denotes the traffic density at coordinate of the helper in map. α is a constant parameter satisfying $\alpha > 0$. Due to the limitation of the capacity of transmitter, we intuitively define the communication range $O_{v_e,u}$ in a way that $O_{v_e,u} \propto K_{v_e}$ and $O_{v_e,u} \propto \frac{1}{D_{x_{v_e},y_{v_e}}}$. This definition can be comprehended in this way: Under a certain power constraint, the more vehicles around the helper, the communication range will be shorter, since the helper will first serve the vehicle end users which are closer to him. The distance between helper v_e and subscriber u which connected to the helper is denoted as $d_{v_e,u}$. Then we have the constraint as $d_{v_e,u} \leq O_{v_e,u}$.

We consider eNBs e located in the suburb area, transmitting in circular coverage of star topology with radius r_e. The vehicle which receives the data transmission (also known as the helper) from the eNB is denoted as v_e. The distance between e and v_e is denoted as d_{e,v_e}. Since our work considers the location of the eNB and the helper in a pixel level, we assign the location of the eNB and the helper as coordinate (x_e, y_e) and (x_{v_e}, y_{v_e}), respectively. We assume the transmission between the eNB and the helper has line of sight without any block from building, trees or flying objects. And we also ignore the height difference between the eNB and the helper, which can be extended. Then we can denote the distance between the eNB and helper as: $d_{e,v_e} = \sqrt{(x_e - x_{v_e})^2 + (y_e - y_{v_e})^2}$. Since the eNB can only cover the helpers within its communication range, we have the constraint $d_{e,v_e} \leq r_e$.

According to the Shannon theorem and in the quasi-static scenario, the achievable rate of helper v_e when associating with transmitter e can be expressed as $r^e_{v_e} = B_{v_e}\log_2\left(1 + \frac{p_{e,v_e}|h^e_{v_e}|^2}{N_o B_{v_e}}\right)$, where B_{v_e} denotes the bandwidth allocated to helper v_e, which we set to 10 MHz. p_{e,v_e} is the transmitted power from eNB e to the receiver on v_e, which is set to 30 dBm. $h^e_{v_e}$ is the channel gain between helper v_e and transmitter eNB e, which is Rayleigh distributed with the mean 1. N_o is the noise spectral density. The relationship between the transmitted power p_{e,v_e} and received power at vehicular relay p^r_{e,v_e} is denoted as $\frac{p^r_{e,v_e}}{p_{e,v_e}} = \left[\frac{\sqrt{G_l}\lambda}{4\pi d_{e,v_e}}\right]^2$. $\sqrt{G_l}$ is the product of the transmit and receive antenna field radiation patterns in the line-of-sight direction, which is set to 15 dB. λ is the wavelength of transmitted signal. For the transmission rate between the helper and the subscriber $r^{v_e}_u$, we don't consider the path loss effect due to two reasons. First, the position of the virtual relay node changes gradually. Thus, the distance $d_{v_e,u}$ changes little from time to time. Second, the distance $d_{v_e,u}$ is within several meters. For different subscribers u, the path loss effect makes little difference. In the experiment, we set the value of signal-to-noise ratio (SNR) from the relay v_e to the subscribers u as 40 dB.

The data transmitted from eNB e to helper v_e is denoted as m_{e,v_e}, which is counted in bit. Then the time consumed during the transmission between e and v_e can be denoted as $l_{e,v_e} = \frac{m_{e,v_e}}{r^e_{v_e}}$. Similar to the latency between helper v_e and subscriber u, l_{e,v_e} also can be seen as the latency of the transmission between e and v_e, which is a factor affecting the quality of service.

IV. PROBLEM FORMULATION

Nowadays, the technology of monitoring real-time traffic is really mature. The traffic can be monitored based on moving vehicles under different weather and illumination conditions [41]. In the industry application, a lot of software provide us with convenient monitoring of real-time traffic every day. For example, when we drive on the highway, we can check Google Map on our phone to get the latest traffic status miles ahead of our scheduled route. Such information not only gives us the expectation of driving experience beforehand but also offer the potential decision for drivers to alter their route if the current one is not satisfying. The example above is a simple decision-making problem which we may encounter in daily life.

The example above gives us the inspiration to build a connection between the observation of traffic flow and control action in wireless communication. In our problem, there are several eNBs continuously transmitting data to a certain location. The location is represented by a certain pixel on our traffic topology (referring to Fig. 2). The value of the pixel

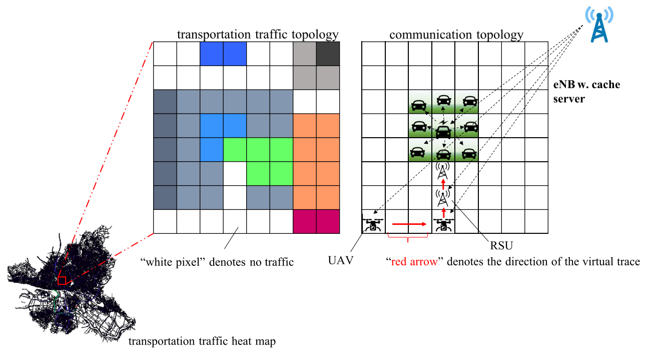

transportation traffic topology
communication topology
eNB w. cache server
RSU
"white pixel" denotes no traffic
UAV
"red arrow" denotes the direction of the virtual trace
transportation traffic heat map

FIGURE 2. Transportation topology and communication topology. The transportation topology is a matrix of RGB pixel cropped from a heat map. The RGB value (a certain color) represents a certain traffic density. The white pixel represents no traffic. For communication topology, we build the same size of the matrix as the transportation traffic topology. For each entry in the communication topology, the RGB value of the corresponding pixel in the transportation topology denotes the traffic density. All entries in the communication topology are potential virtual relay node. The eNB selects an entry for transmission and gradually hands over to the adjacent ones. If there is no traffic (white pixel), the UAVs and RSUs can serve as the alternative helpers.)

stands for the traffic density at that location. This value can be zero which means no vehicle appears in that spot at that specific timestamp. If there is no vehicle on that spot, the relay task cannot be accomplished, and no rewards received. Under the assumption of volunteerism, if there exists at least one vehicle at that spot, any vehicle there can be chosen as the helper (relay node) by receiving the signal from the eNB and disseminating it to nearby vehicles. The goal of our work is to select the vehicle relay node in order to achieve the best overall system throughput without controlling the movement of vehicles on road. The core idea is that we generate a trace of the virtual relay on which potential real vehicular relay can be selected. This trace crosses through the whole map and might not follow the direction of streets.

At a certain timestamp t, we regard the current traffic heat map is one of our observations, which is denoted by the matrix $S_{traffic}$ with n rows and n columns, and each entry represents the RGB value. Thus, $S_{traffic}$ has the shape $(n, n, 3)$. After converted to greyscale, $S_{traffic}$ has a shape of $(n, n, 1)$. The other observation at timestamp t is the topology of the wireless communication, which is denoted by the matrix S_{comm} with n rows and n columns. For those spots containing the vehicular relay and connected end users, we assign the corresponding entries with 1, and otherwise with 0. S_{comm} has the shape of $(n, n, 1)$. This design helps us highlight the current location of the vehicle relay and the end-user vehicles which connect to it. We denote the observation at timestamp t as s_t, where $s_t = [S_{traffic}, S_{comm}]$. The state s_t has the shape of $(n, n, 2)$. With the observations (or states), we assume that an agent is employed by us to accomplish the virtual relay node selection task. In order to train the agent, we assume that all information regarding the wireless communication between eNBs and connected helpers, as well as helpers and connected subscribers (e.g., distance, transmission rate, the power consumed, the volume of delivered data, etc.) are known to an operator. The operator acts as a supervisor to give the feedback to the agent after any move it accomplishes. This supervision exists only during the training phase. After the agent is well-trained, we can let the agent make decisions without the instruction from the operator.

The decision of the agent relies on the observation we describe above. By feeding the observation and location of the virtual relay at last timestamp, the agent can decide an action a_t to take at time t. The action space is denoted as $\{0, 1, 2, \ldots, 8\}$, where 0 represents "staying put", and other numbers represent 8 possible directions in which the spot of vehicle relay can move. The traffic heat maps are fed continuously and current packets allocation topology is only related to the last action (Current action will result in another packets allocation topology at next timestamp). We can easily find out that s_t follows a first-order Markov chain as:

$$P(s_t|s_1, a_1, s_2, a_2, \ldots, s_{t-1}, a_{t-1}) = P(s_t|s_{t-1}, a_{t-1}). \quad (1)$$

Under state s_t, the execution of action a_t will invite an instant reward denoted as r_t, which can be expressed as:

$$r_t = \beta_{r(e,v_e)} \cdot r^e_{v_e}(t) + \beta_{r(v_e,u)} \cdot \sum_u r^{v_e}_u(t) - \beta_{l(e,v_e)} \cdot l_{e,v_e}(t) - \beta_{l(v_e,u)} \cdot l_{v_e,u}(t) - \beta_{f(v_e)} \cdot f_{v_e} - \beta_{P(e)} \cdot P_e, \quad (2)$$

where we define the instant reward at time t by taking the transmission rate $r^e_{v_e}(t)$ and $r^{v_e}_u(t)$ as revenue. The latency $l_{e,v_e}(t)$ and $l_{v_e,u}(t)$ during transmission, the fee f_{v_e} charged by the helper for providing service, and the energy P_e consumed by eNB e are counted as the cost. The parameter β represents the weight assigned to each component contributing to the reward. The larger the value of β, the more contribution it will make to the overall reward. We set a constraint on β that $\beta_{r(e,v_e)} + \beta_{r(v_e,u)} - \beta_{l(e,v_e)} - \beta_{l(e,v_e)} - \beta_{f(v_e)} - \beta_{P(e)} = 1$, where $\beta_{r(e,v_e)}, \beta_{r(v_e,u)} \geq 0$ and $\beta_{l(e,v_e)}, \beta_{l(v_e,u)}, \beta_{f(v_e)}, \beta_{P(e)} \leq 0$. Although, the reward we defined above includes heterogenous components, it makes sense if we treat the revenue or cost that every unit of component evokes the same. This reward can be regarded as the income of the service provider. The weighted components in the equation consider all factors that probably affect the reward, and at the same time enable the adaptation to different forms of rewards according to different demands. For example, if we are only concerned about the transmission rate, we can set $\beta_{l(e,v_e)}, \beta_{l(v_e,u)}, \beta_{f(v_e)}, \beta_{P(e)} = 0$. If there is no traffic at the spot where we place the virtual relay node, it will incur a zero reward, since $r^e_{v_e}(t)$ and $r^{v_e}_u(t)$ will be zero in Eq. 2. We won't choose a specific vehicle as the helper, and control its trajectory, since it would be much more expensive.

To summarize, the definition of the state, action and instant reward at timestamp t can be denoted as follows:

- State: $s_t = [S_{traffic}, S_{comm}]$, where $S_{traffic}$ and S_{comm} represent the matrices for traffic topology and wireless communication topology respectively.
- Action: $a_t = (a_{t,0}, a_{t,1}, \ldots, a_{t,8})$, where $a_{t,0}$ represents that the virtual relay node stays put. Other actions

represent the virtual relay node to be moved towards 8 directions. Every movement crosses one pixel in transportation traffic map.

- Reward: $r_t = \beta_{r(e,v_e)} \cdot r^e_{v_e}(t) + \beta_{r(v_e,u)} \cdot \sum_u r^{v_e}_u(t) - \beta_{l(e,v_e)} \cdot l_{e,v_e}(t) - \beta_{l(v_e,u)} \cdot l_{v_e,u}(t) - \beta_{f(v_e)} \cdot f_{v_e} - \beta_{P(e)} \cdot P_e$. The instant reward r_t here is a sum of weighted value of the transmission rate, latency, service fee, and power consumption.

Since the action in our model is to select a spot in the map as a relay node and virtually move the node (gradually move the spot to select another vehicle at another spot), the current action taken under the current state will impact on the future reward. This inspires us to consider both the current reward and future reward. Thus, we define the long-term system reward as

$$w_t = \sum_{\tau=t}^{\infty} \chi^{\tau-t} \cdot r_\tau, \tag{3}$$

where $\chi \in (0, 1)$ is a discounting rate, which yields a bounded objective for optimization. It reflects the fact that a current action has a weaker impact on the future reward compared with the current reward.

Notice that both current and future reward should be taken into consideration in our problem. The goal of our work is converted to designing a policy π (a mapping from state to action), denoted as $\pi : s_t \rightarrow a_t = (a_{t,0}, a_{t,1}, \ldots, a_{t,8})$, to maximize the long-term reward,

$$V^\pi(s_t) = r_t + \chi \cdot V^\pi(s_{t+1}) = r_t + w_{t+1}, \tag{4}$$

where $V^\pi(s_t)$ is the long-term reward using policy π under the state s_t.

Then, the long-term system reward maximization problem can be formulated as,

$$\max_\pi V^\pi(s_t) \quad \forall s_t, \tag{5}$$

with the recursion relation:

$$V^\pi(s_t) \leftarrow r_t + \chi \cdot V^\pi(s_{t+1}). \tag{6}$$

By taking the observation s_t as input at time t, the agent will output a policy π that indicates which action a_t should be taken. The operator will execute action a_t. After the action is taken, value $V^\pi(s_t)$ for action a_t under current state a_t is evaluated by the operator according to (2). The reward will be fed back to the agent, in order to let it "remember" the experience and react better next time when facing the same state. The traffic and wireless communication topology change from previous time t to current time $t + 1$ after the action executed. Then the agent should take a new observation s_{t+1} and output a new policy, and so on so forth. The final goal is to achieve the objective in (5).

There exist several challenges for this task listed as follows:

1) Since the goal is to optimize the overall reward by the step-by-step control, the observation is discrete and consecutive, i.e., the agent only knows the state of the environment at timestamp t. The action taken at timestamp t will affect the state of the environment at time $t + 1$. The state of the environment cannot be obtained beforehand. Therefore, a joint optimization approach considering current and future states is not feasible.
2) Only after the operator executes the action suggested by the agent that the reward of this action is known. As we explained, the operator knows all parameters about the connection between the eNBs and connected helpers, e.g., power consumption, transmission rate, latency, and channel quality indicator. A new action taken will change the connected helpers and also the value of those parameters contributing to the reward. In other words, since location information of eNB is invisible to our agent, the agent cannot know which action has the maximal reward by simply trying every action via the brute-force approach. Once the action has been taken, the operator who is responsible for collecting every information of transmitters will know all information including the volume of transmitted data, latency, the power consumption of the eNB, etc. By using the information from the operator, the expected reward can be calculated. In short, a method of dynamic trial-and-error should be adopted.
3) In our work, the virtual relay node moves gradually. Firstly, the transmission between the eNB and vehicle has a duration which continues for a few minutes or more. Secondly, the handover problem exists that it consumes some time for one vehicle switching link to another eNB or helper. Thus, in our model, the virtual relay spot should move gradually, instead of jumping from one spot to another far-off spot. In our model, the virtual relay node is controlled like a robot moving around the city. Under this assumption, the action will have a long-term effect on the reward in the future. Thus, the impact of current action on the future rewards should be considered.

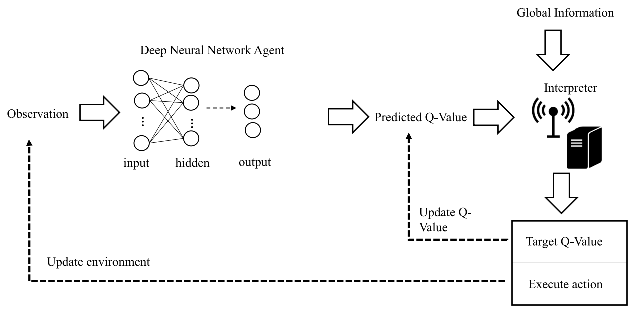


FIGURE 3. The Pipline of the solution for the proposed virtual relay selection problem.

To overcome the obstacles we mentioned above, we are going to propose a novel DRL method in Section V to maximize the long-term reward with implicit and partial information is provided.

V. DEEP Q-LEARNING

RL is known as a sequential decision-making technique, which solves control problems in many fields. However, traditional RL is rather unstable during training. In addition, the observation of the environment is complex for some traditional agents, such as shallow-layer neural networks. With the development of novel algorithms, computation ability of a computer, availability of big data, DL can provide a much better comprehension of the environment. Thus, DRL combines DL and RL, which makes good use of this outstanding property of DL and achieves good performance in robotics, game playing, and spoken dialogue system. Deep Q-Learning is one of the most popular types of DRL. The details of how it works will be introduced in the following several subsections. In Section V-A, we introduce how we obtain our training data and the method for data preprocessing. In Section V-B, some concepts of Q-learning is introduced. Combining DL and Q-learning, the technique of deep Q-networks is introduced in Section V-C, and actor-critic Q-networks scheme is described in Section V-D. Finally, in Section V-E, we explain the exploration policy of the deep Q-networks (DQNs) agent.

A. DATA PREPARATION AND PREPROCESSING

The traffic simulator used in our work is named Streets4MPI [15], which is a software that can simulate simple street traffic patterns based on street maps imported from OpenStreetMap [35]. It is written in Python and supports parallel computation. OpenStreetMap provides street maps from the countries all around world. We can set some parameters in the simulator to obtain different simulation results. We can load different street map by changing the parameter "osm_file". The parameter "number_of_residents" can be set to assign the total number of trips calculated in one simulation round. We also can set "max_simulation_steps" to change how many times the simulation execute. One simulation step will output one static traffic heat map for the current time. Thus, we can attain a series of traffic heat maps representing traffic topologies from time to time. After we get the traffic map, we first downsize it to make it only contain the content we need. Then, we convert the RGB traffic map to greyscale and improve the contrast of the image, which reduces the amount of computation and also improves the training speed.

We use a matrix to represent the wireless communication topology. The size of the matrix is the same as the traffic map. The spots of the helper and its surrounding connected vehicles will be annotated with 1. Other spots are annotated with 0. We first normalize the pixel value of traffic maps from 0 to 1. Then the normalized traffic topology (traffic heat maps) will be stacked with wireless communication topology (sparse matrix), which serves as the input to the first convolutional layer (shown in Fig. 4). Note that the traffic topology comes from the real traffic simulation brought by Streets4MPI. The communication topology is the result of the virtual relay selection at the previous time stamp.

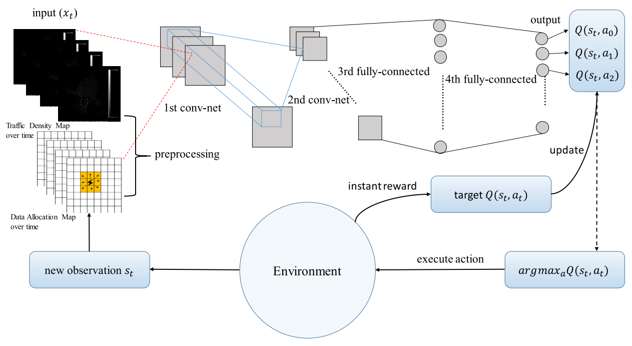


FIGURE 4. Deep Q-learning procedure for the proposed virtual relay selection problem.

B. Q-LEARNING

We utilize a category of model-free reinforcement learning algorithms named Q-Learning. Compared with the value iteration in Section IV, Q-Learning will evaluate Q-Values, which is a state-action values that explicitly tell agent how to react to the observation. $Q^*(s, a)$ is denoted as the expected sum of discounted long-term rewards when the agent takes action a under state s. The iteration function of Q-Learning can be denoted as:

$$Q_{k+1}(s, a) \leftarrow (1 - \eta) Q_k(s, a) + \eta(r + \chi \cdot \max_{a'} Q_k(s', a')), \tag{7}$$

where η is the learning rate. r is the instant reward agent can obtain when it leaves state s with action a. After s and a, the agent would continuously act optimally with discount rate χ.

There are a few techniques for solving the Q-Learning problem. If the states space and action space are limited and discrete, we can utilize a tabular method, which generates a matrix called the Q-Table. The entry of the Q-Table at row i and column j represents the Q-Value $Q(s_i, a_j)$ for taking the j^{th} action a_j under the i^{th} state s_i. At each iteration, the value for one entry of the Q-Table will be updated. After enough iterations, every Q-Value in the Q-Table will become the optimal value.

However, the tabular approach can only solve the Q-Learning problem with finite state and action space. The state space in the real world is almost infinite and table cannot represent all the states. Thus, neural networks are introduced to be used as a function approximator, which maps the states to the Q-Values corresponding to certain actions.

C. DEEP Q-NETWORKS

As we have introduced in Section V-B, the traditional Q-learning uses the table based method to update the long-term reward for action under a certain state. However, the table method cannot scale, which means it cannot handle the situation where the number of states is nearly infinitely large. To tackle this problem, deep Q-Networks (DQN) replaces the table as a new approach to obtain the approximation of Q-Values. Deep Q-Networks can take high-dimension features as input. In our problem, the input is a concatenated 2D image, which contains traffic and V2V communication topology. Both the traffic topology (traffic density map) and wireless communication topology (data allocation map) evolve over time. The traffic topology evolves independently, which obeys the rule of the physical model of the traffic. The data allocation map at the current time step is determined by the action agent takes at the previous timestamp and the traffic density around the current location of the virtual relay node. After the input layer, we stack several convolutional layers followed by several fully connected layers (totally 4 layers) as the structure of our deep Q-networks, which is demonstrated in Fig. 4. For every convolutional layer, the computation can be denoted as:

$$z_{i,j,k} = b_k + \sum_{u=1}^{f_h} \sum_{v=1}^{f_w} \sum_{k'=1}^{f_{n'}} x_{i',j',k'} \cdot w_{u,v,k',k}, \tag{8}$$

where $i' = u \cdot s_h + f_h - 1$ and $j' = v \cdot s_w + f_w - 1$. $z_{i,j,k}$ is the output of the neuron located in row i, column j in feature map k of the convolutional layer (layer l). s_h and s_w are the vertical and horizontal strides, respectively, f_h and f_w are the height and width of the field, respectively, and $f_{n'}$ is the number of feature maps in the previous layer. $x_{i',j',k'}$ is the output of the neuron located in layer $l-1$, row i', column j', feature map k' (or channel k' if the previous layer is the input layer). b_k is the bias term for feature map k (in layer l). You can think of it as a knob that tweaks the overall brightness of the feature map k. $w_{u,v,k',k}$ is the connection weight between any neuron in feature map k of the layer l and its input located at row u, column v (relative to the neuron's receptive field), and feature map k'.

From one layer to another layer, including both convolutional layer and fully-connected in our DQN model, we utilize Rectified Linear Units [42] as the activation function. In short, we denote z as the output, and x' as the input of the next layer. Then we have the relationship as $x' = max(z, 0)$.

Generally, convolutional layers will help extract features from images and reduce the dimension of the data. The output of the last convolutional layers, which is a 2D array, should be converted to a vector and serve as the input of stacked fully-connected layers. The fully-connected layers will select the features from high dimension data. The output of the last fully-connected layer is a vector, which demonstrates the predicted Q-Value for each corresponding action.

Algorithm 1 Deep Double Q-Learning Algorithm

```
1:  Initialize:
      • experience replay memory,
      • training interval and copy interval, $T_{int}$ and $T_{cop}$,
      • boolean variable $flag$ to indicate whether the game ends or not,
      • the actor network with parameter $\theta_{actor}$,
      • the critic network with parameter $\theta_{critic} = \theta_{actor}$.
2:  for training steps $n = 1, 2, \dots, N$ do
3:      Preprocess the initial observation $s_0$ to get beginning input $x_0$ for neural network.
4:      for game episode $t = 1, 2, \dots, T$ do
5:          Generate a random probability $p$.
6:          if $p \leq \varepsilon$ then
7:              randomly select an action $a_t$,
8:          else
9:              $a_t = \arg\max_a Q(x, a, \theta_{actor})$.
10:         end if
11:         Execute action $a_t$, get reward $r_t$, next observation $s_{t+1}$ and $flag$ as feedback.
12:         Preprocess $s_{t+1}$ to get next state $x_{t+1}$.
13:         Store $(x_t, a_t, r_t, x_{t+1}, flag)$ into experience replay memory.
14:         if $n$ mod $T_{int} \neq 0$ then
15:             continue.
16:         end if
17:         Get a batch size of $M_{batch}$ samples $(x^{(i)}, a^{(i)}, r^{(i)}, x^{(i+1)}, flag)$ from experience replay memory.
18:         Calculate target Q-Value, $y^{(i)} = r^{(i)} + \chi \cdot \max_{a'} Q(x'^{(i)}, a', \theta_{actor})$.
19:         Update the critic network by minimizing the cost function $J(\theta_{critic})$,
            $J(\theta_{critic}) = \frac{1}{M_{batch}} \sum_{i=1}^{M_{batch}} (y^{(i)} - Q(x^{(i)}, a^{(i)}, \theta_{critic}))^2$,
            and perform the gradient descent method on $J(\theta_{critic})$ with respect to $\theta_{critic}$.
20:         if $n$ mod $T_{cop} = 0$ then
21:             $\theta_{actor} = \theta_{critic}$.
22:         end if
23:     end for
24: end for
```

D. DOUBLE DQNs AND COST FUNCTION

We use two DQNs with the same architecture, but different trainable parameters, i.e., θ and θ', respectively. One DQN (the actor) will be used to drive the movement of the relay node, and the other DQN (the critic) will watch the actor's trials and learn from its mistakes. The critic, which is also known as the target Q-Network, will compute the loss for every action actor takes during training. The reason for using a separate DQN to generate the target Q-value is that the value of Q-Network will shift at every training step [16]. Using only one network usually incurs uncontrolled estimation of value. Based on the feedback loop structure of reinforcement learning, the network will become more and more unstable.

At regular and frequent intervals [43], the trainable variables in the critic network will be completely copied to the actor network. The updated actor network will start to take actions again, and critic will watch and learn from a new cognition level. This double Q-network structure is shown in Fig. 5.

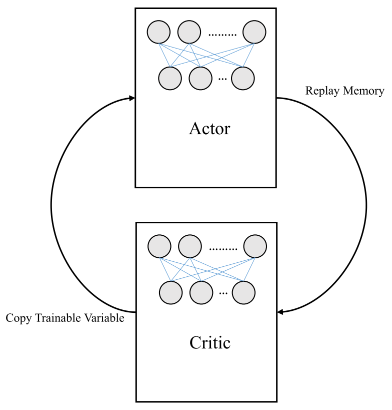


FIGURE 5. **Double Q-network structure.**

In order to let critic learn, all experience the actor obtains during playing will be stored into a container called experience replay memory. Every unit in the experience replay memory can be described as list of the current state s_t at time t, the action a_t taken under the current observation s_t, the next observation s_{t+1} and reward r_t for taking a_t under s_t. Suppose the critic will fetch several units at once from replay memory for training. The number of units fetched at one time is called the batch size, which is denoted as M_{batch}. The cost function used for training is

$$\begin{aligned} J(\theta_{critic}) &= \frac{1}{M_{batch}} \sum_{i=1}^{M_{batch}} (y^{(i)} - Q(s^{(i)}, a^{(i)}, \theta_{critic}))^2, \\ with\ \ y^{(i)} &= r^{(i)} + \chi \cdot \max_{a'} Q(s'^{(i)}, a', \theta_{actor}), \end{aligned} \tag{9}$$

where θ_{critic} and θ_{actor} are the parameters for the critic network and actor network. $s^{(i)}$, $a^{(i)}$, $r^{(i)}$ and $s'^{(i)}$ are the current state, action, reward and next state of the i^{th} memory sampled from the experience replay memory, respectively. $Q(s'^{(i)}, a', \theta_{actor})$ is the prediction of Q-Value actor expects from the next state $s'^{(i)}$ if actor chooses action a'. $Q(s^{(i)}, a^{(i)}, \theta_{critic})$ is the prediction of Q-Value of the i^{th} memory from the expectation of critic network. y^i is the target Q-Value for the i^{th} memory. $J(\theta_{critic})$ is a cost function, which is based on mean squared error. By using mean squared error, we can easily derive the gradient, which saves the computational resources. On the other hand, we don't have outliers in the data set. The RGB value of the traffic heat map is normalized (the value falls in (0,1)), and the value function is a continuous function of the input data. So there are no outliers in the data set, which is the main problem that we should consider when we use the mean squared error. As we can see in (9), how the critic is trained depends on the experience which actor learns.

The training procedure for deep double Q-learning is shown in Algorithm 1. Before the training start, we first initialize the trainable parameters for the actor and critic networks, and the intervals for copying the critic's parameters to the actor's parameters. Also, experience replay memory is initialized to store the experience from the playing of the actor. In Algorithm 1, we set two loops of iterations. The outer loop is for training iterations. The index for counting training steps will increase by one for every time training occurs. The inner loop is for game iterations. For example, if we calculate the cumulative rewards every time after the virtual relay node moves 100 steps, the 100 steps of the virtual relay node movement are one game iteration. By doing so, we can periodically record the performance of the DRL agent. In every training step, DRL agent will move the virtual vehicle relay node T times to finish one complete game. In every game iteration, the deep Q-networks will update its parameters in the way that cost function is minimized.

E. EXPLORATION POLICIES

When the actor is making movement in the environment, it faces a dilemma of balancing exploration and exploitation of the environment. Exploitation means that the actor should choose the action based on the best policy it currently knows. On the other hand, exploration means that the actor should explore more about the environment in order to gather more information. In other words, the best long-term strategy may involve short-term sacrifices. In order to gather enough information to make the best overall decisions, we intuitively expect the actor to explore the environment as thoroughly as possible. From the MDP point of view, if every state and every transition can be visited enough times, we can expect a better performance from the training, since we remember enough situations in replay memory. However, to achieve that, it will be time-consuming if we let the actor select actions by its own judgment every time since it might have preferences. It's the same as people who visit one restaurant many times, they know what are their favorite dishes and they tend to order their preferences more than dishes they are not familiar with.

It might take a long time for the actor to go through every state and transition. For simplicity, we utilize a technique called ϵ-greedy to artificially add interference. At each step, the actor will acts randomly with probability ϵ, or greedily (choose the optimal action) with probability $1 - \epsilon$. During training, we will gradually reduce the value of ϵ to let the actor explore the environment more at the beginning and less when it already knows plenty of good policies.

VI. SIMULATION RESULTS AND DISCUSSIONS

In this section, we evaluate the performance of our scheme by using Google TensorFlow platform. We implement the deep double Q-learning framework for our proposed model and implement other two schemes as baselines. One is the random action scheme and the other is the greedy action scheme. In the random action scheme, the action at every step is chosen randomly. In the greedy action scheme, the agent decides to move the virtual relay node to the one of the adjacent pixels which has the maximum traffic density (pixel value is the largest). In Section VI-A, we set up parameters in the simulation. In Section VI-B, we demonstrate some simulation results and analyze the performance of our proposed model.

TABLE 1. Hyperparameters of the DQNs.

Layer	Name	Parameters	Dimensions	Parameter Scale
Input	–	–	(2,100,100)	–
Layer 1	Convolution	Filter (5,5,5,5)	(5,100,100)	125
Layer 2	Convolution	Filter (10,3,3,2)	(10,20,20)	90
Layer 3	Convolution	Filter (10,2,2,1)	(10,10,10)	40
Layer 4	Data flatten	–	(1000,)	0
Layer 4	Fully-connected	–	(512,)	512,000
Output	–	–	(9,)	–

A. SIMULATION SETTINGS

In this simulation, we consider the traffic density map, whose size is 100×100. The original coordinate $(0, 0)$ locates at upper left of the traffic map. We set x axis along the vertical direction and y axis along the horizontal direction. The location of eNB is at coordinate $(-10, -10)$. The eNB transmits data to the helper, whose coordinate can be at any of those 10, 000 coordinates. Each pixel of the map represents a squared area with the size 10 meters $\times$ 10 meters. The whole squared training field has the size 1000 meters $\times$ 1000 meters. In the simulation, we select an initial coordinate for the helper, and let it move gradually. We set the bandwidth of the transmission as 10 MHz for eNB to the helper and the helper to end users. The transmission power for the eNB to the helper and the helper to the end users are set to 30 dBm. For the transmission from the helper to the subscribers, we set the SNR to a fixed value as 40 dB. For every unit of data received by end user, we give 100 units of revenue. For every unit of power consumed for both eNB and the helper, we give it 1 unit of cost. For every unit of latency of the service, we give it 1 unit of cost. In this way, we unify the units of different components, and let the reward function Eq. 2 make sense. Every component in the reward function shares the same value for weight β in our experiments. The final reward is calculated by the revenue minus the cost.

For the structure of the proposed DQNs, we stack 3 convolutional layers followed by 1 fully-connected layer. The detailed configuration of the DQNs structure are shown in Table 1. The actor and critic Q-network have the same structure. The training occurs after actor takes action every time, which means total training steps are equal to total action steps. The size of experience replay memory is set to 5,000. The size of the minibatch is 10, and thus 10 samples will be randomly fetched from experience replay memory for every training step. All the trainable variables in critic networks will be copied to actor networks every 10 training steps. The discount rate χ is set to 0.95. The value ϵ is set to 0.95 at the beginning of training. With the iterations of training increasing, ϵ will decay until it reaches minimum value 0.05. 100 different traffic density maps will be fed into our model iteratively. We define every 100 steps as one game.

B. SIMULATION RESULTS

In Fig. 6, we compare the cumulative rewards gained by our deep double Q-network (DDQN) with the greedy and random action scheme. As we can find out from the result,

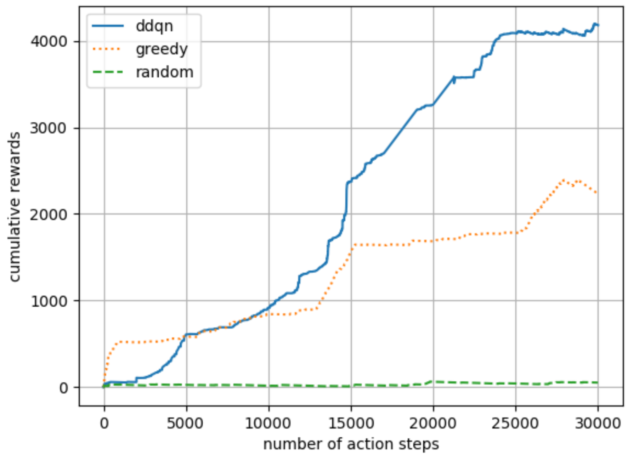


FIGURE 6. The number of action steps (training steps) versus cumulative rewards.

during the beginning 2,000 steps, the cumulative rewards gained by DDQN are almost the same with the random action approach. After that, the rewards gained by DDQN increase exponentially until it reaches 5,000 steps. From step 5,000 to step 15,000, the rewards gained by the agent take another exponential growth. After step 15,000, the rewards grow linearly until step 25,000. On the contrary, the random action approach gains little overall rewards compared to DDQN during the whole training process. For the greedy action scheme, it performs better than DDQN during the first 5,000 steps at beginning, and achieves an early advantage. From step 5,000 to step 10,000, the greedy action scheme accumulates approximately the same amount of rewards as the DDQN. After this period, DDQN has a larger growth rate than the greedy action scheme, and the greedy action scheme cannot catch up with the DDQN anymore. From this result, we demonstrate that the rewards in this scenario not only depend on the traffic density at a single pixel on the traffic density map, but also on the surrounding traffic patterns. The convolution operation in the neural networks works due to the consideration of the relationship between the adjacent pixels (potential location for virtual relay).

In Fig. 7, we evaluate the average rewards over iterations. The average rewards are calculated as the cumulative rewards divided by the current number of steps. By doing so, we burnish the cumulative rewards curves by removing variance of rewards from step to step. The average rewards curve of the random action approach is mostly flat, which indicates no growth in rewards harvesting from step to step. On the other

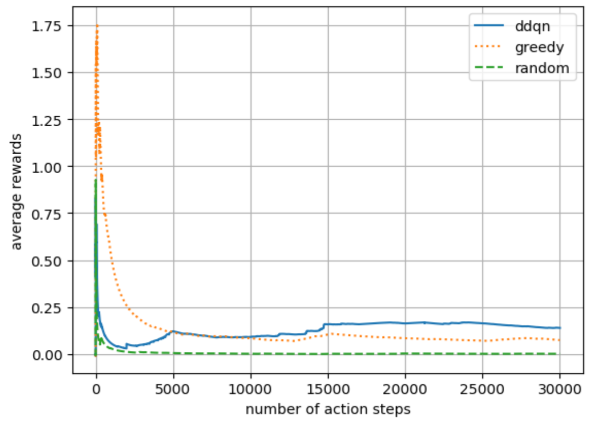


FIGURE 7. The number of action steps (training steps) versus average rewards.

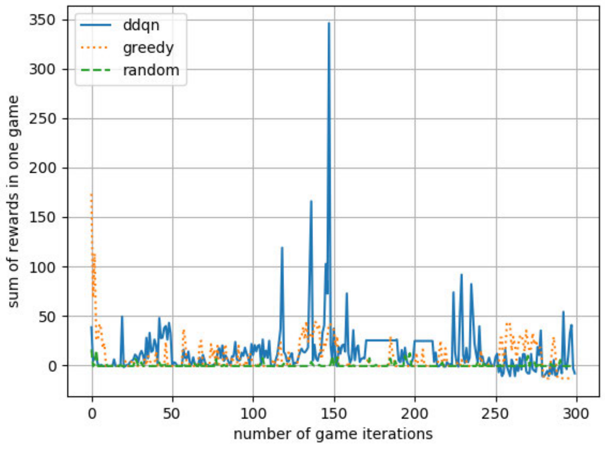


FIGURE 8. The number of game iterations versus rewards collected in one game.

hand, we observe that at the first 2,000 steps, the average reward of the greedy action scheme is big, and after that, it drops fast and converge at around 5,000 step. The curve for DDQN is flat and has a slight growth afterward. After around 15,000 steps, all curves converge and DDQN has the biggest average reward.

In Fig. 8, we collect rewards gained in every game, which contains consecutive 100 iterations. Since we implement the algorithm for 30,000 iterations, we have the rewards of 300 games counted. We find out some peaks and valleys between 100 game iterations and 250 game iterations, which corresponds to action steps (or training steps) from step 10,000 to 25,000. Compared with Fig. 6, we find out that these spikes correspond to the sudden rewards growth after a small period of saturation. We observe that the values of rewards in valleys during this period are almost the same with the rewards of the random action approach. After those valleys, some peaks can be observed, which indicates that DDQN learns how to deal with those circumstances and recover from those adversities. Compared with DDQN, the greedy action and random action scheme have smaller variance from game to game, which indicates that there is no

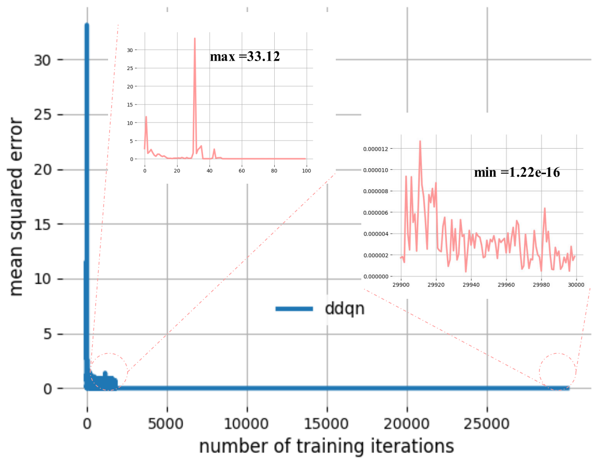


FIGURE 9. The number of training iterations versus mean square error.

such learning moments happen compared with DDQN as we have discussed.

In Fig. 9, we evaluate the mean squared error (MSE) of our DDQN model, which measures the squared value of the difference between the predicted Q-value and target Q-value. At the very beginning, the MSE is high, but drops dramatically and remains at a low level after the step 2,000. This curve demonstrates that the agent went through a studying period during the first 2,000 steps. Afterward, it handles the problem well and produces an accurate prediction of the Q-value. In order to show the result clearly, we zoom in to the first 100 steps and last 100 steps of the training, which is shown by the pink curves. The maximum and minimum mean squared error of the whole training process are 33.12 and 1.22e-16, respectively.

According to the demonstration of the figures we have discussed, there is a period at beginning of the training (the first 2,000 steps). After this period, the agent reduces errors and collects positive rewards. The behavior of the agent is represented in Fig. 10. We sample 6 traffic maps of the time stamps during that period. We can clearly observe in those traffic maps that there are some eye-catching patterns in the maps. At each pixels, the brighter the color is, the higher traffic density is. At step 5,000, the coordinate of the helper (relay point, which is a red dot in the map) is at (38, 25) when the agent stays in an area where there exists heavy traffic. From step 12,000 to 14,000, we discover that the virtual relay node hangs around the left corner of the map, which let it continuously and linearly collect rewards, which we can refer to the Fig.6. From step 22,000 to step 28,000, the virtual relay node gradually moves to the right bottom area, where the traffic density is low. Referring to Fig. 8, we also observe some negative rewards collected during this period accordingly. From Fig. 10, we demonstrate the behavior of the virtual relay node, and the relationship between the behavior and the numeric training results we have discussed and shown in previous figures.

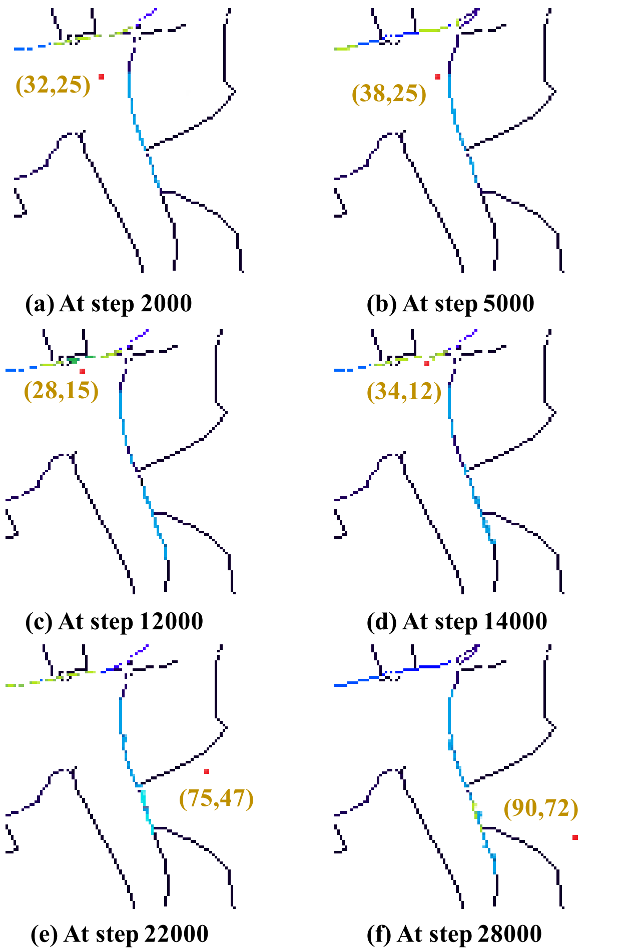


FIGURE 10. The location change of the virtual relay node through training steps. The red dot represents the vehicle relay with the brown annotation of its coordinate. The green and blue patterns in the traffic maps represent relatively high traffic density. The purple or black patterns represent relatively low traffic density.

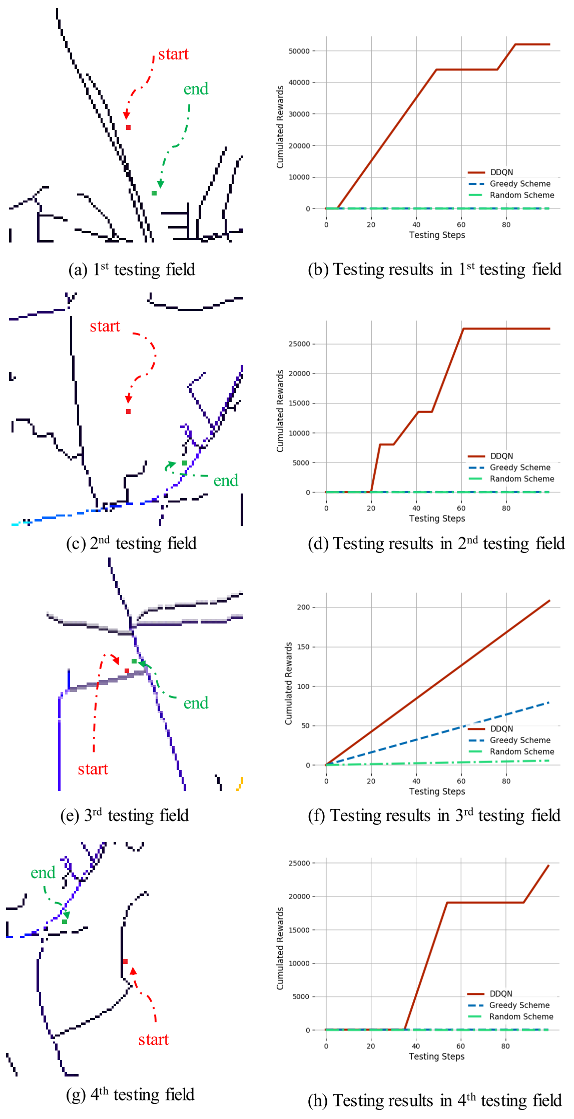


FIGURE 11. The visualization of the testing fields and corresponding cumulated rewards through testing steps. The subgraphs in the left column show the 4 different testing fields on which we implement our testing. The subgraphs in the right column show the corresponding cumulated rewards through 100 testing steps.

The above results show the training performance of the DDQN agent in a 100×100 sized field. For validation, we also let the agent select virtual relay node in a squared field with side length 100, however with 4 different traffic maps (thus different states or inputs). Considering the starting point of the virtual relay might affect the performance of the agent, we generate 4 traffic maps which are different with the training field. The testing process goes through 100 steps for each testing traffic topologies.

In Fig. 11, we evaluate the cumulated reward through testing steps. We select 4 different traffic topologies in order to diminish the performance difference caused by the variation of traffic condition. We should consider the agent scheme to be robust if the agent is going to make wise decision no matter what traffic topology it deals with. In the left column of the figure, we can see the 4 subgraphs showing the 4 different traffic topologies we used for testing. On each traffic graph, we annotate the starting and ending spot of the virtual relay node. In the right column of the figure, 4 subgraphs shows the corresponding cumulated rewards collected through testing steps. We can observe from the results that DDQN scheme has much better performance than the greedy and random scheme. In Fig. 11(a), (c) and (g), we can see that the position of the virtual relay node goes through a significant change. Along these movements, the cumulated rewards gain some huge leaps, which demonstrate the intelligence the agent gained. In Fig. 11(e), we can see that the position of the virtual relay node barely changes. From Fig. 11(f), we observe that the agent considers that the sweet spot is around the starting point. A relatively constant growth of cumulated rewards is obtained by all of the three schemes. However, the DDQN scheme has larger gradient than the other two schemes.

In Fig. 12, we compare the cumulative transmission rate through 100 testing steps by three different schemes. There are a few saturation found in the curves, which means no transportation traffic found at that step, or in other words, no transmission connection built. From the result,

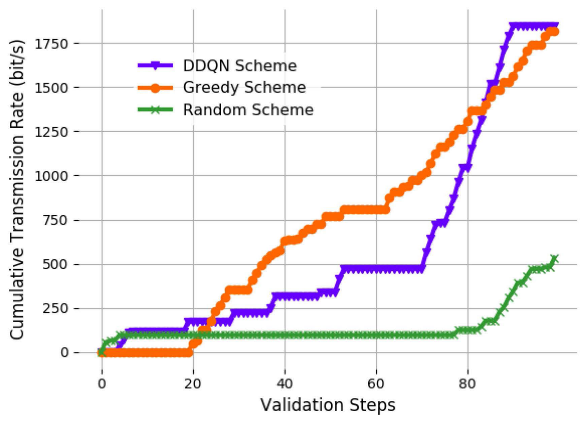


FIGURE 12. The number of validation iterations versus cumulative transmission rate from eNB to the vehicle relay.

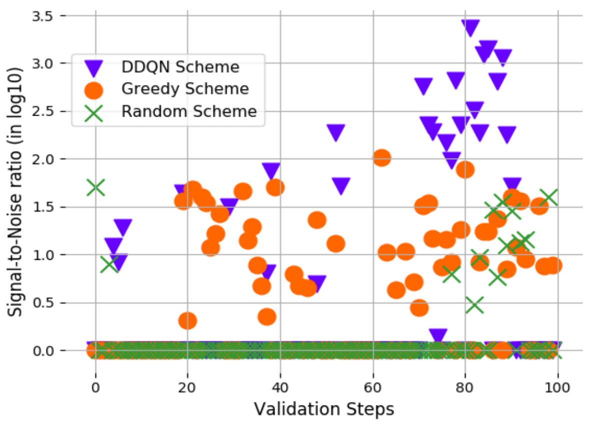


FIGURE 13. The number of validation iterations versus signal-to-noise ratio at the vehicle relay.

we observe that the greedy scheme has an edge over the other two schemes at early steps. With the validation going on, the DDQN scheme catches up and surpasses the greedy scheme, which shows that the proposed DDQN scheme is more powerful harvesting the opportunity of transmission connection in the long term.

In Fig. 13, we use the signal-to-noise ratio (SNR) in $log10$ at the vehicle relay as the channel quality indicator. There are a few zero values on the x-axis representing no signal received at the vehicle relay instead of $SNR = 1$. We compare the result of the three schemes. The random scheme has an unsteady and discontinuous transmission. It loses the connection for a long period in the middle. The greedy scheme has a steady and continuous connection with a concentrated distribution of the SNR. The DDQN scheme is able to find better spots in order to achieve better SNR with the validation going on. The average value of the SNR from DDQN scheme is also higher than the greedy and random scheme.

In Fig. 14, we use bar plot to demonstrate the relative number of the connected end users which are served by the helper. This relative value is calculated based on the opportunistic contact $O_{v_e,u}$ defined in Section III-B, which shows the transmission coverage of the helper. We have three

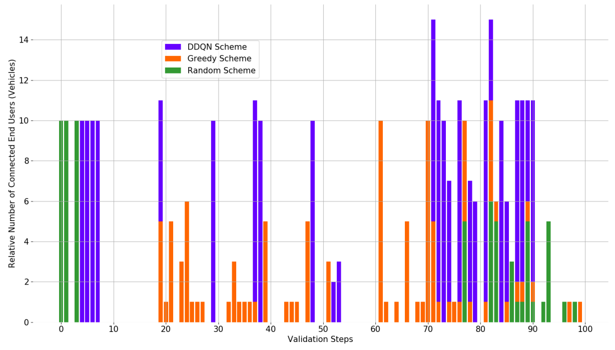


FIGURE 14. The number of validation iterations versus the relative number of the connected end users.

findings from this result: 1) Our proposed DDQN scheme and the greedy scheme have more steady transmission coverage than the random scheme. 2) By using the greedy scheme, we can ensure the transmission connection to some extent. However, the average coverage of the end users is smaller than the DDQN scheme. 3) Generally, once the DDQN scheme builds a connection, it covers more vehicles than the greedy and random scheme.

The analysis above demonstrates that the agent in our model has achieved intriguing intelligence. DRL makes it possible to let agent make wise decisions and overcome obstacles in the environment. From the aspect of vehicle relay, our proposed model helps select location for the virtual vehicle relay with only traffic information. Moreover, by utilizing our scheme, the optimized action can be chosen for setting the position of vehicle relay at the current time without knowing the traffic condition in the future. The long-term maximum utilities can be obtained and corresponding trajectory of the virtual vehicle relay node can be produced by the agent in the proposed model. In this work, we only consider one vehicle relay scenario. From MDP point of view, if we consider n relays, every relay has 9 actions, then the size of the action space becomes 9^n. The state space in our problem is generally infinite since the distribution of the traffic changes continuously. The more action taken basically means more state encountered.

VII. CONCLUSION

In this paper, we proposed a deep reinforcement learning framework in the LTE-V scenario for virtually selecting vehicle relay node. In the framework, we use deep Q-networks as our agent, which contains two types of networks. Firstly, we use deep convolutional neural networks to extract traffic patterns in the input. Then, fully-connected networks are utilized to flatten the high dimensional data and map to the Q-values as output. Q-learning is used to obtain the target Q-values by interacting with the environment. The target Q-values can be used as the labels for training deep Q-networks. Deep Q-networks agent will move the virtual vehicle relay node gradually on the traffic map. Every movement will arise rewards, the amount of which depends on the next observation. We train our agent by letting it interact with environment iteratively and learn from trials and errors. Simulation results show that agent has an excellent performance in rewards collection and error correction. Compared with the greedy and random decision-making schemes, our agent is able to improve utility performance dramatically.

REFERENCES

[1] S. Chen, J. Hu, Y. Shi, and L. Zhao, "LTE-V: A TD-LTE-based V2X solution for future vehicular network," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 997–1005, Dec. 2016.

[2] G. Araniti, C. Campolo, M. Condoluci, A. Iera, and A. Molinaro, "LTE for vehicular networking: A survey," *IEEE Commun. Mag.*, vol. 51, no. 5, pp. 148–157, May 2013.

[3] NHTSA Public Affairs. *USDOT Releases 2016 Fatal Traffic Crash Data*. Accessed: Nov. 2016. [Online]. Available: https://www.nhtsa.gov/press-releases/usdot-releases-2016-fatal-traffic-crash-data

[4] *IEEE Standard for Information Technology–Local and Metropolitan Area Networks–Specific Requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments*, IEEE Standard 802.11p-2010, Jul. 2010, pp. 1–51.

[5] C. H. Lee, K. G. Lim, B. L. Chua, R. K. Y. Chin, and K. T. K. Teo, "Progressing toward urban topology and mobility trace for vehicular ad hoc network (VANET)," in *Proc. IEEE Conf. Open Syst. (ICOS)*, Langkawi, Malaysia, Oct. 2016, pp. 120–125.

[6] M. Amadeo, C. Campolo, and A. Molinaro, "Enhancing IEEE 802.11p/WAVE to provide infotainment applications in VANETs," *Ad Hoc Netw.*, vol. 10, no. 2, pp. 253–269, Mar. 2012.

[7] Z. H. Mir and F. Filali, "LTE and IEEE 802.11p for vehicular networking: A performance evaluation," *EURASIP J. Wireless Commun. Netw.*, vol. 2014, no. 1, pp. 1–15, May 2014.

[8] National Highway Traffic Safety Administration. *2012 Motor Vehicle Crashes: Overview*. Accessed: Nov. 2013. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811856

[9] S. Zuther and K. Dietmayer, "360°-eenvironment sensing and signal processing for an automotive pre-crash application," in *Proc. IEEE Int. Conf. Veh. Electron. Saf. (ICVES)*, Pune, India, Nov. 2009, pp. 50–55.

[10] A. Buchenscheit, F. Schaub, F. Kargl, and M. Weber, "A VANET-based emergency vehicle warning system," in *Proc. IEEE Veh. Netw. Conf. (VNC)*, Tokyo, Japan, Oct. 2009, pp. 1–8.

[11] V. Milanes, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, and M. Nakamura, "Cooperative adaptive cruise control in real traffic situations," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 1, pp. 296–305, Feb. 2014.

[12] C. Desjardins and B. Chaib-draa, "Cooperative adaptive cruise control: A reinforcement learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1248–1260, Dec. 2011.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, Feb. 2015.

[14] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[15] J. Fietkau. *A Street Traffic Simulation Example Project for MPI-Based Parallel Computing in PyThon*. Accessed: May 2, 2012. [Online]. Available: https://github.com/jfietkau/Streets4MPI

[16] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," *CoRR*, vol. abs/1509.06461, pp. 2094–2100, Mar. 2015.

[17] D. Tian, J. Zhou, Z. Sheng, M. Chen, Q. Ni, and V. C. M. Leung, "Self-organized relay selection for cooperative transmission in vehicular ad-hoc networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 10, pp. 9534–9549, Oct. 2017.

[18] M. F. Feteiha and H. S. Hassanein, "Enabling cooperative relaying VANET clouds over LTE—A networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 4, pp. 1468–1479, Apr. 2015.

[19] W. Song and X. Tao, "Analysis of a location-aware probabilistic strategy for opportunistic vehicle-to-vehicle relay," in *Proc. IEEE 86th Veh. Technol. Conf. (VTC-Fall)*, Toronto, ON, Canada, Sep. 2017, pp. 1–6.

[20] D. Wang, P. Ren, Q. Du, L. Sun, and Y. Wang, "Security provisioning for MISO vehicular relay networks via cooperative jamming and signal superposition," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 10732–10747, Dec. 2017.

[21] M. Seyfi, S. Muhaidat, J. Liang, and M. Uysal, "Relay selection in dual-hop vehicular networks," *IEEE Signal Process. Lett.*, vol. 18, no. 2, pp. 134–137, Feb. 2011.

[22] X. Tang, P. Ren, and Z. Han, "Hierarchical competition as equilibrium program with equilibrium constraints towards security-enhanced wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 7, pp. 1564–1578, Jul. 2018.

[23] Q. Du, H. Song, and X. Zhu, "Social-feature enabled communications among devices toward the smart IoT community," *IEEE Commun. Mag.*, vol. 57, no. 1, pp. 130–137, Jan. 2019.

[24] L. Ma, X. Huo, X. Zhao, and G. D. Zong, "Observer-based adaptive neural tracking control for output-constrained switched MIMO nonstrict-feedback nonlinear systems with unknown dead zone," *Nonlinear Dyn.*, vol. 99, no. 2, pp. 1019–1036, Jan. 2020.

[25] Y. Ge, S. Wen, Y.-H. Ang, and Y.-C. Liang, "Optimal relay selection in IEEE 802.16j multihop relay vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 5, pp. 2198–2206, Jun. 2010.

[26] R. Chai, Y. Qin, S. Peng, and Q. Chen, "Transmission performance evaluation and optimal selection of relay vehicles in VANETs," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, San Francisco, CA, USA, Mar. 2017, pp. 1–6.

[27] B. Yang, X. Sun, R. Chai, L. Cai, and X. Yang, "Game theory based relay vehicle selection for VANET," in *Proc. IEEE 24th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, London, U.K., Sep. 2013, pp. 2924–2928.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, Dec. 2012, pp. 1097–1105.

[29] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transp. Res. C, Emerg. Technol.*, vol. 54, pp. 187–197, May 2015.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: http://arxiv.org/abs/1312.5602

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[32] X. Xiong, J. Wang, F. Zhang, and K. Li, "Combining deep reinforcement learning and safety based control for autonomous driving," *CoRR*, vol. abs/1612.00147, pp. 1–10, Dec. 2016.

[33] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 13th AAAI Conf. Artif. Intell.* Phoenix, AZ, USA: AAAI Press, Feb. 2016, pp. 2094–2100.

[34] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO–simulation of urban mobility: An overview," in *Proc. 3rd Int. Conf. Adv. Syst. Simul. (SIMUL)*, Barcelona, Spain, Oct. 2011, pp. 1–6.

[35] OpenStreetMap Foundation. *OpenStreetMap*. Accessed: Aug. 9, 2004. [Online]. Available: https://www.openstreetmap.org

[36] M. Rondinone, J. Gozalvez, J. Leguay, and V. Conan, "Exploiting context information for V2X dissemination in vehicular networks," in *Proc. IEEE 14th Int. Symp. 'World Wireless, Mobile Multimedia Netw.' (WoWMoM)*, Madrid, Spain, Jun. 2013, pp. 1–9.

[37] X. Zhu, Y. Li, D. Jin, and J. Lu, "Contact-aware optimal resource allocation for mobile data offloading in opportunistic vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7384–7399, Aug. 2017.

[38] C. Ye, P. Wang, C. Wang, and F. Liu, "Mobility management for LTE-based heterogeneous vehicular network in V2X scenario," in *Proc. 2nd IEEE Int. Conf. Comput. Commun. (ICCC)*, Chengdu, China, Oct. 2016, pp. 2203–2207.

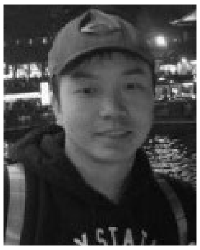
[39] K. Lee, Y. Yi, J. Jeong, H. Won, I. Rhee, and S. Chong, "Max-contribution: On optimal resource allocation in delay tolerant networks," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010, pp. 1–9.

[40] H. Zhu, L. Fu, G. Xue, Y. Zhu, M. Li, and L. M. Ni, "Recognizing exponential inter-contact time in VANETs," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010, pp. 1–5.

[41] J. Zhou, D. Gao, and D. Zhang, "Moving vehicle detection for automatic traffic monitoring," *IEEE Trans. Veh. Technol.*, vol. 56, no. 1, pp. 51–59, Jan. 2007.

[42] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, "Understanding deep neural networks with rectified linear units," in *Proc. Int. Conf. Learn. Represent.*, Vancouver, BC, Canada, Apr. 2018, pp. 1–17.

[43] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, pp. 1–14, Sep. 2015.



XUNSHENG DU (Graduate Student Member, IEEE) received the B.S. degree in management information system from the Huazhong University of Science and Technology, in 2015. He is currently pursuing the Ph.D. degree advised by Dr. Z. Han with the Department of Electrical and Computer Engineering, University of Houston. His research interests include drilling optimization, wireless communication, vehicular networks, image processing, deep learning, and reinforcement learning.

HIEN VAN NGUYEN (Member, IEEE) received the B.S. degree from the National University of Singapore, in 2007, and the Ph.D. degree from the University of Maryland at College Park, in 2013. He is a passionate researcher in machine learning, artificial intelligence, and computer vision. His research focus is on increasing the data-efficiency of learning algorithms and improving their ability to handle distributional shifts of data.

CHUNXIAO JIANG (Senior Member, IEEE) received the B.S. degree (Hons.) in information engineering from Beihang University, Beijing, in 2008, and the Ph.D. degree (Hons.) in electronic engineering from Tsinghua University, Beijing, in 2013. He is currently an Associate Professor with the School of Information Science and Technology, Tsinghua University. His research interests include application of game theory, optimization, and statistical theories to communication, networking, and resource allocation problems, in particular space networks and heterogeneous networks. He has served as a member for the Technical Program Committee. He was a recipient of the Best Paper Award from the IEEE GLOBECOM, in 2013, the Best Student Paper Award from the IEEE GlobalSIP, in 2015, the IEEE Communications Society Young Author Best Paper Award, in 2017, the Best Paper Award IWCMC, in 2017, the IEEE ComSoc TC Best Journal Paper Award of the IEEE ComSoc TC on Green Communications and Computing, in 2018, the IEEE ComSoc TC Best Journal Paper Award of the IEEE ComSoc TC on Communications Systems Integration and Modeling, in 2018, and the Best Paper Award ICC, in 2019. He has served as the symposium chair for a number of international conferences, including the IEEE ICC 2018 Symposium Co-Chair, the IWCMC 2018/2019 Symposium Chair, the WiMob 2018 Publicity Chair, the ICCC 2018 Workshop Co-Chair, and the ICC 2017 Workshop Co-Chair. He has served as an Editor for the IEEE INTERNET OF THINGS JOURNAL, the IEEE NETWORK, the IEEE COMMUNICATIONS LETTERS, and a Guest Editor for the *IEEE Communications Magazine* and the IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING.

YONG LI (Senior Member, IEEE) received the B.S. degree from the Huazhong University of Science and Technology, in 2007, and the M.S. and the Ph.D. degrees in electrical engineering from Tsinghua University, in 2009 and 2012, respectively. From 2012 to 2013, he was a Visiting Research Associate with the Telekom Innovation Laboratories and The Hong Kong University of Science and Technology. From 2013 to 2014, he was a Visiting Scientist with the University of Miami. He is currently a Faculty Member with the Department of Electronic Engineering, Tsinghua University. His articles have total citations more than 7200 Google Scholar. Among them, ten are ESI highly cited papers in computer science and five receive the conference best paper (run-up) awards. His research interests include big data, mobile computing, and wireless communications and networking. He has served as the general chair, a TPC chair, a TPC member for several international workshops and conferences. He serves on the editorial board of four international journals.

F. RICHARD YU (Fellow, IEEE) received the Ph.D. degree in electrical engineering from The University of British Columbia (UBC), in 2003. From 2002 to 2006, he was with Ericsson, Lund, Sweden, and a Start-Up, USA. He joined Carleton University, in 2007, where he is currently a Professor. He is also a Registered Professional Engineer, Canada, a Distinguished Lecturer, and the Vice President (Membership). His research interests include wireless cyber-physical systems, connected/autonomous vehicles, security, distributed ledger technology, and deep learning. He is a Fellow of the Institution of Engineering and Technology (IET). He was an Elected Member of the Board of Governors (BoG) of the IEEE Vehicular Technology Society. He received the IEEE Outstanding Service Award, in 2016, the IEEE Outstanding Leadership Award, in 2013, the Carleton Research Achievement Award, in 2012, the Ontario Early Researcher Award (Premiers Research Excellence Award), in 2011, the Excellent Contribution Award from the IEEE/IFIP TrustCom 2010, the Leadership Opportunity Fund Award from the Canada Foundation of Innovation, in 2009, and the best paper awards from the IEEE ICNC 2018, VTC 2017 Spring, ICC 2014, Globecom 2012, the IEEE/IFIP TrustCom 2009, and the International Conference on Networking, in 2005. He has served as the technical program committee (TPC) Co-Chair of numerous conferences. He serves on the editorial boards of several journals, including the Co-Editor-in-Chief of *Ad Hoc and Sensor Wireless Networks*, the Lead Series Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, and the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS.

ZHU HAN (Fellow, IEEE) received the B.S. degree in electronic engineering from Tsinghua University, in 1997, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, in 1999 and 2003, respectively. From 2000 to 2002, he was a Research and Development Engineer with JDSU, Germantown, Maryland. From 2003 to 2006, he was a Research Associate with the University of Maryland. From 2006 to 2008, he was an Assistant Professor with Boise State University, Idaho. He is currently a John and Rebecca Moores Professor with the Electrical and Computer Engineering Department and the Computer Science Department, University of Houston, Texas. He is also a Chair Professor with National Chiao Tung University, China. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grid. He received the NSF Career Award, in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society, in 2011, the Best Paper Award of *EURASIP Journal on Advances in Signal Processing*, in 2015, the IEEE Leonard G. Abraham Prize in the field of communications systems (best paper award in the IEEE JSAC), in 2016, and several best paper awards in the IEEE conferences. He has served as the IEEE Communications Society Distinguished Lecturer, from 2015 to 2018. He has been a 1% highly cited researcher, since 2017, according to *Web of Science*.