Computing on Wheels: A Deep Reinforcement Learning-based Approach

S. M. Ahsan Kazmi, Tai Manh Ho, Tuong Tri Nguyen, Muhammad Fahim, Adil Khan, Md. Jalil Piran, Gaspard Baye

***Abstract*—Future generation vehicles equipped with modern technologies will impose unprecedented computational demand due to the wide adoption of compute-intensive services with stringent latency requirements. The computational capacity of the next generation vehicular networks can be enhanced by incorporating vehicular edge or fog computing paradigm. However, the growing popularity and massive adoption of novel services make the edge resources insufficient. A possible solution to overcome this challenge is to employ the onboard computation resources of close vicinity vehicles that are not resource-constrained along with the edge computing resources for enabling tasks offloading service. In this paper, we investigate the problem of task offloading in a practical vehicular environment considering the mobility of the electric vehicles (EVs). We propose a novel offloading paradigm that enables EVs to offload their resource hungry computational tasks to either a roadside unit (RSU) or the nearby mobile EVs, which have no resource restrictions. Hence, we formulate a non-linear problem (NLP) to minimize the energy consumption subject to the network resources. Then, in order to solve the problem and tackle the issue of high mobility of the EVs, we propose a deep reinforcement learning (DRL) based solution to enable task offloading in EVs by finding the best power level for communication, an optimal assisting EV for EV pairing, and the optimal amount of the computation resources required to execute the task. The proposed solution minimizes the overall energy for the system which is pinnacle for EVs while meeting the requirements posed by the offloaded task. Finally, through simulation results, we demonstrate the performance of the proposed approach, which outperforms the baselines in terms of energy per task consumption.**

***Index Terms*—Next-generation Intelligent Transport System, task offloading, vehicle-to-vehicle communication, deep reinforcement learning.**

I. Introduction

Electric Vehicles (EVs) empowered by modern technologies have revolutionized the traditional transport industry. The modern EVs equipped with high-tech electronics as well as communication, computation, and caching resources have open doors for deploying many novel services in the transport industry realizing the goal of the intelligent transport system (ITS). It is also worth mentioning that the inception of electrically charged battery-propelled vehicles instead of fossil fuel-propelled vehicles has considerably contributed to the decrease in carbon emission, footprints, and other greenhouse gases, which are endangering the environment [1], [2]. Indeed, the storage capacity of these modern EVs have significantly improved due to advances in battery technologies and fueled the adoption of EVs, energy preservation in this domain is still among one of the most important and challenging issues [1]–[3]. Furthermore, this issue becomes even more critical in the modern transport systems due to the popularity and mass adoption of novel resource and energy-hungry services in ITS. For instance, the services such as augmented reality (AR) [4], infotainment services [5], and autonomous driving [6], [7], in the transportation system have significantly increased the energy and resource requirements of EVs. Moreover, such modern services pose an additional challenge of high data communication and computational demands for processing complicated tasks. One possible solution to alleviate the aforementioned challenges is to offload the compute- and energy-intensive tasks.

Task offloading is a key feature in modern wireless networks in which a resource-constrained device offloads its task to nearby devices. For instance, compute- or energy-intensive tasks of traditional resource-constrained devices can use the cloud computing paradigm to offload their tasks [8]. Similarly, the vehicular networks to enable the connected-car technology is further extended (from both services and applications perspectives) to vehicular cloud computing and vehicular social networks. However, the cloud-based approach suffers from high latency due to long distances between the centralized cloud and the devices. To address this challenge, the edge computing paradigm can be adopted, which places computing servers close to the resource-constrained devices, thus, offering low-latency task offloading service [9], [10]. It is evident from the works in [11]–[13] that edge computing combined with approaches such as caching can significantly enhance the task offloading performance resulting in achieving significantly faster response times and better quality of service compared to the traditional cloud model. However, this approach cannot be adopted for vehicular networks due to the requirement of large geographical area coverage, which will pose a huge installation and maintenance cost for such a solution.

One promising approach for task offloading that can be

S. M. Ahsan Kazmi is with the faculty of Computer Science and creative technologies, University of the west of England, Bristol, UK. E-mail: {ahsan.kazmi@uwe.ac.uk}

Tai M Ho is with the Synchromedia Lab, École de Technologie Supérieure, Université du Québec, QC, Canada E-mail: {manh-tai.ho.1@ens.etsmtl.ca}

Tuong Tri Nguyen is with the Institute of Open Training and IT, Hue University, 05 Ha Noi, Hue City , Vietnam, E-mail: {ntuongtri@hueuni.edu.vn}

Adil Khan is with Innopolis university, Russia, E-mail: {a.khan@innopolis.ru }

Muhammad Fahim is with the school of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, UK. E-mail: {m.fahim@qub.ac.uk}

M. J. Piran is with the Department of Computer Science and Engineering, Sejong University, 05006 Seoul, South Korea. E-mail: {piran@sejong.ac.kr}.

Gaspard Baye is with the Computer and Information Science (CIS), University of Massachusetts Dartmouth, USA. E-mail: {bgaspard@umassd.edu.}

adopted in vehicular networks is to use the resources of nearby EVs. Such an approach reaps several benefits for vehicular networks. For instance, computational capacity can be enhanced, a low-latency task offloading service can be provided, and higher communication efficiency can be achieved. Furthermore, this type of offloading approach can be effective for EVs in highway scenarios, where the demands of vehicles are very dynamic due to high speeds and the density of roadside units (RSUs) in an area is typically low. Coping with the task offloading issues, it is imperative to design energy-efficient offloading solutions by taking into account the vehicular mobility, computational capacity, and latency requirements for a dynamic vehicular environment.

In a dynamic vehicular setting, the main challenge of task offloading among EVs is meeting the stringent latency requirement. This issue becomes even more challenging in a highway scenario in which EVs are moving and the environment changes dynamically. This can eventually result in longer inter-EV distances and would then require more resources and energy to perform a task resulting in longer latency or even incomplete execution if the latency requirement is not met.

Thus, an efficient task offloading approach is required to increase the overall resource availability in the network and optimize the dynamic vehicular network performance subject to the latency requirement.

A. Related Work

In recent years, task offloading in vehicular networks has gained significant attention to enhance the driving experience by enabling modern resource-hungry applications in ITS. A number of recent reseach works have shown to achieve significant performance gains by utilizing the edge computing paradigm for task offloading [5], [14], [15].

Particularly, Premsankar et al. in [14] proposed an efficient solution that could optimize the deployment cost of edge computing for mixed-integer linear programming (MLP) problem in a vehicular network. An energy-efficient heuristic approach for task offloading was presented in [15] using edge computing in vehicular networks. Similarly, the authors in [5] proposed an optimization-based solution for providing infotainment services. In this work, task offloading was done to reduce latency by jointly addressing the problem of communication, caching, and computational resources in vehicular networks. It is evident from the aforementioned works that edge computing can significantly enhance the performance of the vehicular networks by enhancing the computational resources and meeting the latency requirements. However, with the massive growth of EVs and modern services in ITS, the edge computing paradigm will eventually get strained in terms of resource capacity and would require massive edge computing server installation, eventually resulting in a huge installation costs. This issue can be addressed by an alternative approach in which strong onboard computing resources of nearby EVs can be utilized for performing task offloading. This approach would require to use the vehicle-to-vehicle (V2V) communication to perform the task offloading process, which can significantly avoid the huge installation cost of the edge servers.

A number of recent works have studied the usage of onboard vehicular resources to perform task offloading [16]–[21]. In [16], an efficient low latency scheme was presented in which vehicles collaborated to provide the task offloading mechanism. Similarly, a novel task offloading solution namely "Fog Following Me (Folo)" was proposed in [17]. The authors utilized vehicular resources to reduce the overall latency of the system. Another interesting work presented in [18] is a solution taking into account the quality of experience (QoE) for vehicles to prove the task offloading services via neighboring vehicles.

Moreover, there have been a number of incentive-based task offloading solutions for vehicular networks presented in [20]–[23]. The main objective of these works was to provide incentives to serving vehicles and motivate them in participating in the task offloading process. For instance, Huang et al. in [19] employed the resources of parked vehicles for executing computationally heavy tasks in a distributed manner. The authors in [20] presented interesting results by developing a Stackelberg game-based solution for task offloading and maximizing the benefits both for the edge server and the serving vehicles. Another Stackelberg game-based incentive scheme for task offloading was presented in which resources of the parked vehicles were utilized to maximize the overall social benefit of the vehicular network [21]. Similarly, in [22], the contract theory was employed to design an incentive mechanism that reduces latency for task offloading proposed and further extended this work by providing priority to each task in [23]. Zhou et al. in [24], integrated the contract theory and matching theories to find an interesting solution for allocating communication resources and task assigning in vehicular networks. An integration of machine learning (ML) and the contract theory has been used in [25] to enhance the performance of task offloading services in vehicular networks. In all the discussed works, the modeling of a dynamic vehicular environment is not done properly or is inappropriately modeled for realistic vehicular networks in which vehicles are moving at high speeds. In reality, vehicle mobility can significantly affect the task offloading process, especially when both the source and destination are mobile. This can result in a significant performance degradation if not handled appropriately. Therefore, a novel task offloading approach is required that can capture the dynamic nature of vehicular networks and design an appropriate task offloading scheme.

ML-based approaches are promising to find solutions for such dynamic environments. A number of ML-based approaches have been investigated and promising results have been reported. For instance, Chen et al. in [26] investigated a joint problem of resource allocation and content caching in cache-enabled networks. The authors have formulated the joint problem considering the user association, spectrum allocation, and content caching and proposed a distributed algorithm based on the liquid state machine framework. Their results showed a significant performance gain compared to the traditional Q-learning approaches.

In the domain of the vehicular networks, Ye et al. in [27] investigated the resource allocation problem for V2V setting

based on reinforcement learning (RL). Each vehicle was considered as an agent, which takes decisions independently to choose the optimal power and sub-band for enabling V2V communication by reuse of sub-bands employed for Vehicle-to-infrastructure (V2I). The results revealed the performance gain in terms of higher communication rate by employing V2V communication. Furthermore, in [28], the authors further extended their work in which they investigated the resource allocation for unicast and broadcast scenarios based on the decentralized RL. Similar to their previous work, also each agent (e.g. link or vehicle) takes the optimal decision based on the communication power and sub-band to learn how to satisfy V2V communication while reducing interference to V2I. In [29], the authors investigated the spectrum sharing based on multi-agent RL (MARL). In this work, the aim was to enable V2V communication by reusing the preoccupied V2I links without significantly affecting them. To solve the problem, the authors proposed a MARL-based solution that improves the sum capacity of V2I and V2V links. Xinran et al. in [30], investigated a joint mode selection and resource allocation problem in a vehicle-to-everything (V2X) communication architecture and formulated the problem as a Markov decision process (MDP), and applied a decentralized deep RL (DRL) algorithm that considers both V2V and V2I communication. Moreover, for the training limitation of DRL, a federated learning (FL)-based algorithm was proposed, which the authors claimed that significantly outperformed the baseline approaches. Note that in the discussed work, a significant performance gain was observed by employing learning-based approaches in a dynamic environment, however, learning was employed only to optimize the communication resources of the network.

Kun et al. in [31] employed the learning to investigate an efficient task offloading problem. Specifically, RL was used for the Internet of vehicles (IoV). In this work, the vehicles offload their data to RSUs, where RSUs were attached to MEC servers to perform the computational and analytical tasks. Then, RL was used to solve the problem and the result obtained inferred fast convergence of the proposed offloading scheme. Similarly, the authors in [32] proposed a solution for the vehicular edge computing domain using DRL. In this work, the knowledge-driven features of the IoV were used to predict the optimal action in a given state. Then, the authors formulated a long-term planning problem and showed the solution converges, and stated that the proposed solution adapts faster than the greedy offloading decision scheme for a dynamic setting. Indeed, superior performance has been shown by employing RL-based approaches for task offloading in vehicular networks due to their ability to capturing dynamic environments. However, those works did consider the energy efficiency (EE) aspects for vehicular networks, which is pivotal to employ electric vehicles. Moreover, they did consider modeling and providing solutions for task offloading among multiple vehicles.

In the EV domain, the task offloading mechanism should be able to improve the EE as the task processing consumes energy and this counts as the most important parameter in task offloading for EV networks. An interesting work based on EE and ML was presented in [33] in which the authors investigated online UAV-assisted wireless caching by jointly optimizing UAV trajectory, transmission power, and scheduling content caching and formulated the problem as infinite-horizon ergodic MDP to provide a better QoE. To solve this problem, a fluid approximation approach was used to derive a reduced-complexity optimality state and based on that an efficient RL algorithm was presented. However, this type of solution increases the cost of the network due to the installation of UAVs.

In summary, none of the aforementioned solutions considered a practically dynamic and mobile environment for task offloading with the aim of EE in EVs. Evidently, properly modeling a dynamic vehicular environment in which resources are shared among nearby mobile EVs will result in a huge resource capacity for the vehicular networks and will provide a better task offloading opportunities. This results in unlocking the full potential of novel resource-hungry services in the next generation of vehicular networks.

B. Our Contributions

Motivated by the above discussed challenges, in this work, we design an EE task offloading scheme for EV networks subject to latency constraints. Additionally, we aim to increase the computational capacity of the EV networks using computational resources of nearby EVs, which are not resource-constrained. Note that this becomes very challenging as both the source EV and the destination EV are moving, which results in a very dynamic environment that demands proper modeling of the dynamic scenario. To this end, we propose an EE task offloading solution based on DRL for the vehicular networks, which captures such a dynamic environment stemming from the mobility of EVs. The proposed approach enables designing an EE solution that maximizes the ratio of successful task offloading. Note that in this work, we assume that all EVs belong to the same operator and the operator is responsible for providing some benefits or incentive to the serving EVs. Furthermore, serving EVs cannot share their resources unconditionally and they would only share their residual resources after estimating their own requirements. Serving other EVs would require an EV to use its resources and energy, thus, resulting in overall higher resource availability in a service area especially in a highway scenario. In summary, our key contributions are summarized as follows.

- We propose a novel paradigm for EVs to perform task offloading to either the RSU or the nearby mobile EVs. Then, we formulate an optimization problem for task offloading aiming at maximizing the EE subject to the latency, communication, and computation resources of the EVs. The formulated problem turns out to be a non-convex optimization problem and obtaining solutions for such problems are non-trivial and extremely difficult for any practical size settings. Typically traditional optimization-based solutions for such problems employ approaches such as relaxing parameters and several iterations to find a sub-optimal solution.
- Then, we propose a DRL-based algorithm that captures the dynamic nature of the proposed problem and enable

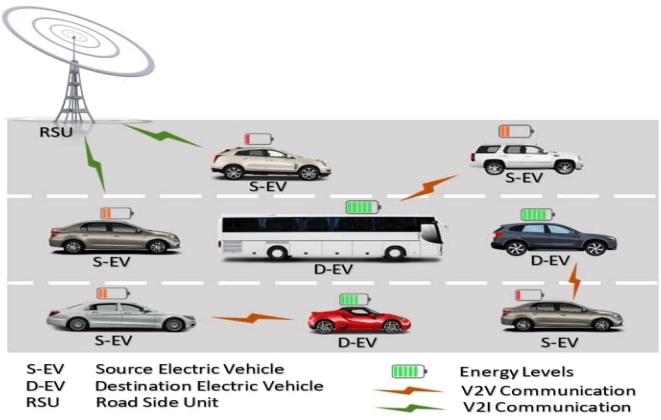


Fig. 1: An EV network in which resource-constrained EVs offload tasks to nearby EVs via V2V communication or RSU via V2I communication.

task offloading. The proposed solution finds the best power level for communication, the optimal EV that can assist the EV pairing for offloading, and the amount of the computing resources to provide for the task offloading. The proposed task offloading scheme minimizes the overall energy for the system while meeting the requirements posed by the offloaded task.

- Through extensive simulation results, we demonstrate that the proposed approach can guarantee convergence and achieve a performance gain of up to 65% in terms of energy per task completed compared to the server computation baseline even for large task sizes.

The rest of the paper is organized as follows. In Section II, we present the system model in which we model the task offloading procedure for a dynamic environment. Section III presents the DRL-based approach to solve the proposed problem. In Section IV, numerical results are presented in which we analyze and validate the performance of our proposed solution. Finally, conclusions are drawn in Section V.

II. System Model and Problem Formulation

Our system model is presented in Fig. 1. We consider a vehicular network with an RSU equipped with multi-access edge computing. Moreover, it is assumed that the RSU is responsible for managing the whole vehicular network, including the activities such as collecting information, wireless resource allocation, computation task assignment, broadcasting decisions, etc. Furthermore, we assume all EVs are distributed uniformly under the RSU service radius R.

We consider two types of EVs in our system, the first type of EV is the resource-constrained EV, which requires offloading service while the other type of EV is the one that provides its resources to assist the resource-constrained EV type. Suppose the set $\mathcal{S}$ represent the resource-constrained EVs namely the source EVs (S-EVs), which require offloading services for their tasks, and a set $\mathcal{D}$ of assisting EVs namely the destination EVs (D-EVs) that have enough resources to provide offloading services.

The RSU makes a decision about the vehicular-assisted offloading. This pairing decision will be used for assistance in task offloading based on the network states and offloading requests from EVs. Typically, the network states represent information such as distance among EVs, available onboard energy of each EV, available computational resource at each EV, etc. Then, based on such information, the RSU makes a decision for pairing S-EV and D-EV together in order to facilitate the task offloading service or providing the task offloading service to itself. Assume $s \in \mathcal{S}$ is the S-EV requesting EV and $d \in \mathcal{D}$ represent the assisting D-EV for providing service for S-EV's s task represented by tuple given by $P_s = \{rs_s, rc_s, rt_s\}$, where rs_s represents the required computation size of task (bits), rc_s is the required number of CPU cycles for processing a bit of data, and rt_s represents the required task deadline (e.g., latency threshold). We define $\alpha_{s,d} \in \{0,1\}, s \in \mathcal{S}, d \in \mathcal{D}$, as the source-destination EVs pairing variable.

$$\alpha_{(s,d)}(t) = \begin{cases} 1, \text{ if S-EV } s \text{ paired to D-EV } d, \\ 0, \text{ otherwise.} \end{cases} \tag{1}$$

Note that in this work, we consider a scenario in which the EVs drive on a highway following the same direction. Then, offloading an S-EV's s task to a destination node (either a D-EV, d, or RSU) requires the following three steps. First, the S-EV's, s, task needs to be sent to the destination node, we name this as "Uplink step (UL)" following our previous work [34]. Second, this transmitted task from S-EV s needs to be executed at the destination node, "EXE step". Finally, the result of this execution has to be sent back from the destination node to the S-EV, s. The three-step procedure is sequential and dependent on each other, thus, failing to complete any of the steps results in failure of the whole task offloading process. Note that we do not consider task offloading to multiple destinations and handovers stemming either due to lack of resources or real time network dynamics, challenges stemming from such larger and complex framework would be subject to our future work. Next, we present the EVs mobility model in the following subsection.

A. EV Mobility Model

It is assumed that both S-EV, s, and D-EV, d, are mobile. Let $x_s(t_0)$ and $x_d(t_0)$ represent the initial position at initial time instant t_0 of S-EV s and D-EV d, respectively. Similarly, $v_s(t_0)$, $v_d(t_0)$, $a_s(t_0)$ and $a_d(t_0)$ represents the S-EV s and D-EV d velocity v and acceleration a at time instant t_0, respectively. Following our previous work [34], we use the motion of relative acceleration, $a_{s,d}$, to compute the difference in acceleration between S-EV, s, and D-EV, d, at time instant, t_0, based on the kinematic equation as follows.

$$a_s(t_0) = \lim_{t \to t_0} \frac{|v_s(t) - v_s(t_0)|}{t - t_0}, \tag{2}$$

$$a_d(t_0) = \lim_{t \to t_0} \frac{|v_d(t) - v_d(t_0)|}{t - t_0}, \tag{3}$$

$$a_{s,d}(t_0) = |a_d(t_0) - a_s(t_0)|. \tag{4}$$

TABLE I: List of notations.

Notation	Definition	Notation	Definition
$\mathcal{R}$	RSU services radius	P_s	Task profile of S-EV $s \in \mathcal{S}$
$\mathcal{S}$	Set of the resource-constrained EVs	$\mathcal{D}$	Set of the assisting EVs (D-EVs)
rc_s	Required CPU cycles to compute one bit of data	rt_s	Maximum tolerable latency to compute the task P_s
rs_s	Required computation size of task [bits]	$\alpha_{(s,d)}(t)$	Pairing decision variable
UL	Uplink step	DL	Downlink step
EXE	Execution step	$x_{(\cdot)}(\cdot)$	Location/position of EV
$a_{(\cdot)}(\cdot)$	Acceleration of an EV	$a_{(\cdot,\cdot)}(\cdot)$	Relative acceleration of an EV pair
$d_{s,d}(t)$	Distance between EV s and EV d	$d_{\max}$	Maximum V2V communication range
$g_{\cdot,\cdot}(\cdot)$	Channel power gain	η	The Rayleigh channel coefficient with a complex coefficient distribution
$\psi_{s,s'}(t)$	Transition probability from state s to state s'	$\boldsymbol{\psi}_{s,d}(t)$	Channel state transition probability between s and d
$W(t_0)$	System bandwidth at time t_0	p_s	The transmit power of S-EV s
I_0	Additive White Gaussian Noise (AWGN)	$l_s(\cdot)$	The uplink transmission delay
$l_d(\cdot)$	Downlink transmission delay	rs_s^{out}	Task output size after computation
$h_d^{\max}$	Maximum computation capacity of D-EV d	$h_0^{\max}$	Maximum computation capacity available at the RSU
$l_e(t)$	Execution delay	$\boldsymbol{G}_s[t]$	Channel gains between any S-EV and destination
$\boldsymbol{H}_d[t]$	Computational resource allocated from destination	$\boldsymbol{\Psi}$	Stationary control policy
$\mathcal{X}$	Action space	$\mathcal{N}$	State space

Moreover, we can calculate the distance traveled by any EV, j, (either S-EV or D-EV) between the time instant, t_0, and time instant t, by finding its current position, $x_j(t)$, using the following kinematic equation.

$$x_j(t) = \int_{t_0}^{t} \left(v_j(t_0) + a_j(x)x\right) \partial x. \tag{5}$$

The objective is to find whether S-EV, s, can be assisted by D-EV, d, for task offloading. Note that for a successful offloading task, all the three task offloading steps are needed to be completed. This can be ensured with the help of the relative acceleration, $a_{s,d}$, between the source and the destination. Employing the notion of the relative acceleration, we can easily calculate the distance between the source and the destination at any given time instant, t. Assuming that the location of EVs s, and d at time t is $x_s(t)$, $x_d(t)$, respectively. Based on (5), we can measure the distance of $d_{s,d}(t)$ as follows.

$$\begin{aligned} d_{s,d}(t) &= x_d(t) - x_s(t) \\ &= \left(v_d(t_0) - v_s(t_0)\right) t + \frac{1}{2}\left(a_{s,d}(t)\right) t^2. \end{aligned} \tag{6}$$

Then, based on the distance at time instant, t, we can observe whether S-EV, s, and D-EV, d, can use the V2V communication if it has not violated, d_{max}, e.g., the maximum V2V communication range. Furthermore, it can be seen that the higher the relative acceleration, the greater is the inter EV's distance, and vice versa. An offloading process is completed if and only if all offloading steps are executed, however, the downlink step in which the results are needed to sent back to the S-EV, s, from the destination requires the D-EV, d, to be in the acceptable range of V2V communication. In the next subsections, we define the communication and computation model.

B. Communication and Computation Model

In our model, we use the long-term evolution (LTE)- and the fifth-generation (5G)-based technologies as opposed to the dedicated short-range communications (DSRC)/IEEE802.11p technologies due to their higher bandwidth, which are more suitable for the task offloading service [5], [24]. Communication is required to offload tasks from the S-EV, s, to the destination node, either D-EV, d, or RSU, 0. Both the uplink and downlink require communication to send the task and obtain the results back. Therefore, we need to calculate the achievable rates under both scenarios, e.g. UL and DL. Moreover, we consider that the wireless channel gain between EVs and their associated RSU is realistic time-varying channels, which reflect a dynamic environment of vehicular networks. Suppose $g_{s,k}(\cdot)$ is the channel power gain at a reference distance of $d_{s,k} = 1(m)$ (e.g., $k \in \mathcal{D} \cup 0$). Thus, the channel power gain between the S-EV, s, and destination, k, at time slot, t, can be modeled as a random variable, $g_{s,k}(t)$. Formally, we present the channel gain for each scenario as follows.

$$g_{s,d}(t) = g_{s,d}(\cdot) d_{i,j}(t)^{-\eta}, \tag{7}$$

$$g_{d,s}(t) = g_{d,s}(\cdot) d_{i,j}(t)^{-\eta}, \tag{8}$$

$$g_{s,0}(t) = g_{s,0}(\cdot) d_{i,0}(t)^{-\eta}, \tag{9}$$

$$g_{0,s}(t) = g_{0,s}(\cdot) d_{i,0}(t)^{-\eta}. \tag{10}$$

The transition probability that $g_{s,d}(t)$ moves from one network state, s, to another state, s', can be denoted as $\psi_{s,s'}(t)$ at time instant, t. Then, the wireless channel state transition probability matrix for the wireless link between the S-EV, s, and D-EV, d, is defined as follows.

$$\boldsymbol{\psi}_{s,d}(t) = [\psi_{s,s'}(t)], \tag{11}$$

where $\psi_{s,s'}(t) = \mathbf{Pr}(g_{s,d}(t+1) = s' | g_{s,d}(t) = s)$. Similarly, other scenarios pertaining to channel gain also follow the same procedure. Suppose $r_{s,d}$ is the required achievable data rate for the EV pair, $s - d$, at time, t_0, e.g., UL step communication.

$$r_{s,d}(t_0) = W(t_0) \log_2 \left(1 + \frac{p_{s,d} g_{s,d}(\cdot) d_{s,d}^{-\eta}(t_0)}{I_0}\right). \tag{12}$$

Moreover, the transmit power can be calculated using the following:

$$p_{s,d}(t_0) = \frac{I_0 d_{s,d}^{\eta}}{g_{s,d}(\cdot)} \left(2^{r_{s,d}/W(t_0)} - 1\right), \tag{13}$$

where $W(t_0)$ is the system bandwidth at time t_0, p_s represents the transmit power of S-EV, s, $d_{s,d}(t_0)$ represents the distance between S-EV, s, D-EV, d, and Rayleigh channel coefficient with a complex coefficient distribution η, I_0 represents AWGN. Similarly, if the destination node is the RSU, 0, then, the power for UL communication for an S-EV, s, is given as follows.

$$p_{s,0}(t_0) = \frac{I_0 d_{s,0}^{\eta}}{g_{s,0}(\cdot)} \left(2^{r_{s,0}/W(t_0)} - 1\right), \tag{14}$$

where $g_{s,0}$ and $d_{s,0}(t_0)$ are the channel gain and distance between S-EV s and the RSU 0, respectively. We consider that the RSU allocates orthogonal resource blocks for enabling V2V communication. Thus, we assume no inter-resource block interference and would be a subject for future studies. Similarly, in the downlink scenario, let $r_{d,s}(t)$ represent the minimum required data rate between D-EV, d, and S-EV, s, after time $t = t_0 + \triangle t$. Then, the required power for enabling this communication is as follows.

$$p_{d,s}(t) = \frac{I_0 d_{s,d}^{\eta}}{g_{d,s}(\cdot)} \left(2^{r_{d,s}/W(t)} - 1\right). \tag{15}$$

On the contrary, the required power to enable communication between the RSU, 0, and S-EV, s, is:

$$p_{0,s}(t) = \frac{I_0 d_{s,0}^{\eta}}{g_{0,s}(\cdot)} \left(2^{r_{0,s}/W(t)} - 1,\right), \tag{16}$$

where $p_{0,s}$ is the downlink transmit power of the RSU. Note that $d_{i,0}(t)$ will only reflect the S-EV mobility. Moreover, we can calculate the UL and DL transmission delay as follows.

$$l_s(t) = \alpha_{(s,d)}(t)\frac{rs_s}{r_{s,d}(t)} + (1 - \alpha_{(s,d)}(t))\frac{rs_s}{r_{s,0}(t)}, \tag{17}$$

$$l_d(t) = \alpha_{(s,d)}(t)\frac{rs_s^{out}}{r_{d,s}(t)} + (1 - \alpha_{(s,d)}(t))\frac{rs_s^{out}}{r_{0,s}(t)}, \tag{18}$$

where rs_s^{out} represents the size of the result after executing the task in the execution step. Next, we present our computation model.

Computation is required once the task is offloaded from the S-EV, s, to the destination node. In our model, we assume computation can be performed either at the D-EV or RSU. We also assume each D-EV, d, has a limited computing resource represented by h_d^{max} as opposed to the RSU's computation resource represented by h_0^{max}. Then, the objective is to identify and allocate the required computational resources for executing a task, which meets the offloaded task's deadline. Therefore, we need to compute the execution delay for that task. Given offloading task's size rs_s, we can calculate the execution delay (l_e) as follows.

$$l_e(t) = \alpha_{(s,d)}(t)\frac{rc_s rs_s}{h_{d,s}(t)} + (1 - \alpha_{(s,d)}(t))\frac{rc_s rs_s}{h_{0,s}(t)}, \tag{19}$$

where $h_{d,s}$ and $h_{0,s}$ represent the total computation resource (e.g., CPU cycles) assigned for S-EV's s task by D-EV d or RSU 0, respectively. Each task of S-EV s need to abide a maximum time threshold rt_s, which implicitly states, the uplink, execution and downlink transmission of the task need to be done within this time. Then, assume $\triangle t_s$ represents the time required to complete these steps resulting in the following constraint, which needs to be satisfied for successful task offloading.

$$\sigma_s(t) = l_s + l_d + l_e \leq \tau_s = \min\{rt_s, \triangle t_s\}, \forall s \in \mathcal{S}. \tag{20}$$

In the next subsection, we formulate our task offloading problem for a mobile vehicular network.

C. Task Offloading problem formulation

We formulate an energy minimization problem for vehicular networks in which energy for task offloading services is minimized. Next, the task offloading utility for our work is modeled as the amount of energy consumed for successfully offloading and completing a task. Given the service requirement on minimum achievable data rate presented in the previous subsection:

$$E_{s,d}(\boldsymbol{p}, \boldsymbol{h}, t) = \frac{rs_s^{in}}{r_{s,d}(t)} p_{s,d}(t) + \kappa h_{d,s}^2(t) rs_s^{in} + \frac{rs_s^{out}}{r_{d,s}(t)} p_{d,s}(t), \tag{21}$$

$$E_{s,0}(\boldsymbol{p}, \boldsymbol{h}, t) = \frac{rs_s^{in}}{r_{s,0}(t)} p_{s,0}(t) + \kappa h_{0,s}^2(t) rs_s^{in} + \frac{rs_s^{out}}{r_{0,s}(t)} p_{0,s}(t), \tag{22}$$

$$E_s(\boldsymbol{\alpha}, \boldsymbol{p}, \boldsymbol{h}, t) = \alpha_{(s,d)}(t) E_{s,d} + (1 - \alpha_{(s,d)}(t)) E_{s,0}. \tag{23}$$

The objective of our optimization problem is to minimize the energy given in (21) subject to the maximum energy constraints of S-EV and D-EV. Moreover, we need to consider the task deadline threshold and abide by the maximum V2V communication range constraint that depends upon the notion of relative acceleration. Formally, we present our problem as follows.

$$\min_{\alpha,p,h} \quad \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \sum_{s=1}^{S} E_s \tag{24a}$$

s.t.

$$\sigma_s(t) \leq \tau_s, \forall s \in \mathcal{S}, \tag{24b}$$

$$\alpha_{(s,d)}(t) p_{s,d}(t) \frac{rs_s}{r_{s,d}(t)} + (1 - \alpha_{(s,d)}(t)) p_{s,0}(t) \frac{rs_s}{r_{s,0}(t)} \leq E_s^{max}, \forall s \in S, \tag{24c}$$

$$\alpha_{(s,d)}(t) p_{d,s}(t) \frac{rs_s^{out}}{r_{d,s}(t)} + \kappa h_{d,s}^2(t) rs_s \leq E_d^{max}, \forall d \in \mathcal{D}, \tag{24d}$$

$$\alpha_{(s,d)}(t) d_{s,d}(t) \leq d_{max}, \forall s \in S, d \in D. \tag{24e}$$

The constraint in (24b) ensures that the maximum delay condition is met for each S-EV s. Next, the constraints in (24c) and (24d) make sure that enough energy is available to transmit the S-EV, s, task's data to the destination and enough energy is available at the destination node (e.g., at D-EV, d) to execute the offloaded task. Finally, the last constraint (24e) refers to the V2V range between the paired source and destination and ensures that it is not violated.

Note that the problem formulated in (24) is a non-convex optimization problem with a binary decision variable. To obtain a solution for such problems via the traditional optimization approaches incurs high computational complexity requiring exponential time. Furthermore, several iterations are required at every time-slot to obtain a solution with new environmental parameters reflecting the dynamic nature of the problem. Thus, traditional optimization approaches would be intractable to solve the problem (24) for a practical vehicular network setting.

Leveraging the advantage of the DRL technique in solving the problems in a stochastic environment, in the following Section, a DRL-based framework is proposed that suits well to obtain a solution for such problems by capturing the dynamic characteristics of this EV environment. In such an approach, the agents try and learn from the real-time environment and eventually obtain a good approximate solution to such complex problems.

III. DRL-Based Task Offloading Approach

This section presents our solution approach for task offloading in which we choose the best destination node to offload ($\boldsymbol{\alpha}$), the required power level ($\boldsymbol{p}$), and the required computational resource ($\boldsymbol{h}$) to offload the task. In RL, the learning agent continuously interacts with the environment to learn an optimal policy and at each interaction, the agent updates itself by a trial and error correction method. Moreover, at each interaction with the environment, the agent takes an action from a specified action space, which results in a numerical reward and shift to a new state of the environment. The action taken by an agent is defined by a probability reflecting the agent's policy. Then, the goal would be to take such actions based on the best or optimal policy that maximizes the overall reward for the learning agent.

In this work, we have an energy minimization problem as stated in (24). The agent learns a control policy that minimizes energy consumption. In our proposed approach, the agent installed at the RSU collects all the required information from each EV and then construct the system states. In our problem, a state of an environment is the time-varying wireless channel condition between the source and the destination, available energy resources at all EVs, and the available computational resources at the destination. Based on the system states and control policy, the agent will choose the optimal actions ($\boldsymbol{\alpha}^*$, $\boldsymbol{p}^*$ $\boldsymbol{h}^*$) and inform the EVs. The goal in RL is to learn an optimal policy that can maximize the total reward for the agent. However, learning the optimal policy for the RL agent becomes challenging if the state space is very large. To overcome this challenge, DRL was introduced that combines RL with deep neural networks (DNN) [35]. In the next subsection, we review the preliminaries of the DRL for our task offloading problem and model our problem as an MDP [35].

A. RL Preliminaries

Our formulated problem in (24) can be characterized as an MDP as follows.

1) **System states**: In our model, five variables form a state at any given time instant t. The first two variables are characterized by the channel gains between any S-EV and destination ($\boldsymbol{G}_s[t] = \{g_{s,d}[t] | \forall s \in S, d \in D\}$ and $\mathbf{G}_0[t] = \{g_{s,0}[t] | \forall s \in S\}$), while the next variable represent the available energy resources at the D-EV ($e_d[t] = E_d^{max} - e'_d[t] | \forall d \in D$, where $e'_d[t]$ is the energy consumption till time instant t and the final two variables represent the computational resource allocated to the S-EV from destination ($\boldsymbol{H}_d[t] = \{h_{d,s}[t] | \forall s \in S, d \in D\}$ and $\mathbf{H}_0[t] = \{h_{0,s}[t] | \forall s \in S\}$). Then, we define the vector representing all states by $\boldsymbol{n}[t] \in \mathcal{N} = \{\mathbf{G}_s[t], \mathbf{G}_0[t], \mathbf{e}_d[t], \mathbf{H}_d[t], \mathbf{H}_0[t]\}$.
2) **Actions**: In this work, RSU follows a stationary control policy to take an action at each time step t by selecting the control variables ($\boldsymbol{\alpha}, \boldsymbol{p}, \boldsymbol{h}$) of our problem (24), formally represented as $\boldsymbol{x}[t] = (\{\alpha_{s,d}[t]\}, \{p_{s,k}[t]\}, \{h_{s,k}[t]\}) \in \mathcal{X}$, where k represents the destination node ($k \in D \cup 0$).
3) **Stationary control policy**: A stationary control policy $\boldsymbol{\Psi}$ is a mapping from states $\mathcal{N}$ to the probability of selecting an action from the action space $\mathcal{X}$. Formally, defined as $\boldsymbol{\Psi} : \mathcal{N} \rightarrow \mathcal{X}$.
4) **Rewards**: A reward in RL provides feedback to the RSU about the chosen action at each time step t. In this work, we choose reward as a value inversely proportional to $E_s(t)$ per task completed as our reward reflecting lower energy consumption, higher is the reward. Moreover, we also use the task deadline to identify the tasks that do not abide by the maximum deadline threshold. In that case, a penalty φ proportional to the deadline value will be subtracted from the reward value.

A state transition probability of a controlled Markov chain for the given stationary control policy $\boldsymbol{\Psi}$ is as follows.

$$
\begin{aligned}
&\mathbf{Pr}\{\boldsymbol{n}[t+1] | \boldsymbol{n}[t], \boldsymbol{\Psi}(\boldsymbol{n}[t])\} = \\
&\left(\prod_{s \in \mathcal{S}} \prod_{d \in D} \mathbf{Pr}\{g_{s,d}[t+1] | g_{s,d}[t], \boldsymbol{\Psi}(\boldsymbol{n}[t])\} \right) \\
&\times \left(\prod_{s \in \mathcal{S}} \mathbf{Pr}\{g_{s,0}[t+1] | g_{s,0}[t], \boldsymbol{\Psi}(\boldsymbol{n}[t])\} \right) \\
&\times \left(\prod_{d \in D} \mathbf{Pr}\{e_d[t+1] | e_d[t], \boldsymbol{\Psi}(\boldsymbol{s}[t])\} \right) \\
&\times \left(\prod_{s \in \mathcal{S}} \prod_{d \in D} \mathbf{Pr}\{h_{d,s}[t+1] | h_{d,s}[t], \boldsymbol{\Psi}(\boldsymbol{n}[t])\} \right) \\
&\times \left(\prod_{s \in \mathcal{S}} \mathbf{Pr}\{h_{0,s}[t+1] | h_{0,s}[t], \boldsymbol{\Psi}(\boldsymbol{n}[t])\} \right).
\end{aligned}
\tag{25}
$$

The aim of the learning agent (e.g., the RSU) is to find an optimal policy $\boldsymbol{\Psi}^*$ that can maximize the expected rewards from the initial state $n(1)$. In our formulated problem, $E_s(n[t], \boldsymbol{\Psi}(n[t]))$ is the expected reward in terms of energy consumption for the state n on taking an action following the policy $\boldsymbol{\Psi}(n[t])$ for S-EV s. Then, taking an expectation over the long term will provide us the expected long-term energy consumption conditioned over the initial state $n(1)$. Then, the optimal policy $\boldsymbol{\Psi}^*$ for the formulated problem would

minimize this long terms energy consumption, formally, stated as follows.

$$\boldsymbol{\Psi}^* = \arg\min_{\boldsymbol{\Psi}} \Bigg[\mathbb{E}_{\boldsymbol{\Psi}} \Bigg((1-\lambda) \sum_{t=1}^{\Delta T} (\lambda)^{t-1} E(\boldsymbol{n}[t] \; \boldsymbol{\Psi}(\boldsymbol{n}[t])) | \boldsymbol{n}(1) = \boldsymbol{n} \Bigg) \Bigg], \tag{26}$$

where λ represents the discount factor, and $E(\mathbf{n}[t], \boldsymbol{\Psi}(\mathbf{n}[t])) = \sum_{s=1}^{S} E_s(n[t], \boldsymbol{\Psi}(n[t]))$. Lets denote the right hand side of (26) as $V(n, \boldsymbol{\Psi})$, then, the optimal state value function over all states can be denoted as $V(n) = V(n, \boldsymbol{\Psi}^*), \forall n \in \mathcal{N}$. Note that the optimal state-value function can be solved using Bellman's optimality equation [36] in which an action is chosen that minimizes the action-value function. The action-value function is represented as $Q(n, x) : \mathcal{N} \times \mathcal{X} \rightarrow \mathbb{R}$. Note that, the full network statistic information is required to obtain the solution. Alternatively, a model-free RL method, Q-learning technique can be adopted, which learns the optimal control policy without depending on information pertaining to the dynamic information, transition probability of the network states [37].

$$Q(\boldsymbol{n}, \boldsymbol{x}) = (1-\lambda) E(\boldsymbol{n}, \boldsymbol{x}) + \lambda \sum_{\boldsymbol{n}' \in \mathcal{N}} \mathbf{Pr}\{\boldsymbol{n}' | \boldsymbol{n}, \boldsymbol{x}\} V(\boldsymbol{n}'), \forall (\boldsymbol{n}, \boldsymbol{x}). \tag{27}$$

In this method, at each time instant, the Q-function is updated based on the observations that depend upon the current network state, action taken, and the resulting network state at the next time slot. Formally, Q-function is updated as follows.

$$Q(\boldsymbol{n}[t], \boldsymbol{x}[t]) \leftarrow Q(\boldsymbol{n}[t], \boldsymbol{x}[t]) + \phi^t \Big(\beta[t] - Q(\boldsymbol{n}[t], \boldsymbol{x}[t]) \Big), \tag{28}$$

$$\beta[t] = (1-\lambda) E(\boldsymbol{n}[t], \boldsymbol{x}[t]) + \lambda \min_{\boldsymbol{x} \in \mathcal{X}} Q(\boldsymbol{n}[t+1], \boldsymbol{x}), \tag{29}$$

where $\phi^t \in [0, 1]$ is a time-varying learning rate.

Indeed, this approach resolves the requirement of complete state information, however, updating the Q-table for a practically large problem makes this approach impractical. Therefore, one solution is to use the deep Q-network (DQN)-based approach, which can provide an online estimate of the Q-function to alleviate the cost incurred by Q-learning [38]. In DQN, $Q(\boldsymbol{s}, \boldsymbol{x}) \approx Q(\boldsymbol{n}, \boldsymbol{x}; \boldsymbol{\theta^Q})$, where $\boldsymbol{\theta^Q}$ represents the parameters of the neural network that gets updated at each epoch, t. Additionally after each epoch, the RSU stores transitions in a replay memory of a finite size. A transition $\mathcal{T}$ is represented by a tuple as ($\boldsymbol{n}[t]$, $\boldsymbol{x}[t]$, $E(\boldsymbol{n}[t])$, $\boldsymbol{n}[t+1]$). Then, at each epoch t, the RSU samples an experience of K transactions forming a mini batch ($\tilde{\mathcal{O}}^{[t]}$) from the replay memory ($\mathcal{O}^{(t)}$ of size B) to train the network such that it minimizes the loss function presented in (30).

$$\mathcal{L}(\boldsymbol{\theta^Q}) = \mathbb{E}_{\tilde{\mathcal{O}}^{(t)}} \Big[\beta[t] - Q(\mathbf{n}[\mathbf{t}], \mathbf{x}[\mathbf{t}]; \boldsymbol{\theta^Q}) \Big]^2. \tag{30}$$

In order to obtain a stable solution that converges, a common solution is to employ a target network [39]. The target network is simply an earlier version of a DQN and in the next time instant, the target DQN weights are updated to the DQN.

Algorithm 1 : DRL-based EV task offloading for energy minimization

Initialize Phase:
$\mathcal{O}^{(t)}$ of size B and mini-batch $\tilde{\mathcal{O}}^{(t)}$ of size K,
$Q(\boldsymbol{n}, \mathbf{x}|\boldsymbol{\theta^Q})$ as critic network and $Q'(.)$ as target critic network with weights $\boldsymbol{\theta^Q}$ and $\boldsymbol{\theta^{Q'}} \leftarrow \boldsymbol{\theta^Q}$, respectively, actor network $\pi(\boldsymbol{n}|\boldsymbol{\theta^\pi})$ and target actor network $\pi'()$ with weights $\boldsymbol{\theta^\pi}$ and $\boldsymbol{\theta^{\pi'}} \leftarrow \boldsymbol{\theta^\pi}$,$\epsilon$-greedy probability ϵ^{Max}, ϵ^{Min} and ϵ^d decay factor.;
Learning Phase:
for each episode **do**
  Environment reset;
  $t \leftarrow 0$;
  repeat
    Observe $\boldsymbol{n}[t]$ & take action $\boldsymbol{x}[t] = \pi(\boldsymbol{n}[t]|\boldsymbol{\theta^\pi}) + \Gamma$;
    Apply $\boldsymbol{x}[t]$, obtain $\boldsymbol{n}[t+1]$, $E(\boldsymbol{n}[t], \boldsymbol{n}[t])$ and calculate reward;
    Save $(\boldsymbol{n}[t], \boldsymbol{x}[t], E(\boldsymbol{n}[t], \boldsymbol{x}[t]), \boldsymbol{n}[t+1])$ in $\mathcal{O}$;
    Randomly sample mini batch $\tilde{\mathcal{O}}[t] \subseteq \mathcal{O}$;
    Update $\boldsymbol{\theta^Q}$ and $\boldsymbol{\theta^\pi}$ via (30) and (32);
    Update $\boldsymbol{\theta^{Q'}}$ and $\boldsymbol{\theta^{\pi'}}$ as follows:
    $\boldsymbol{\theta^{Q'}} \leftarrow \rho \boldsymbol{\theta^Q} + (1-\rho) \boldsymbol{\theta^{Q'}}$;
    $\boldsymbol{\theta^{\pi'}} \leftarrow \rho \boldsymbol{\theta^\pi} + (1-\rho) \boldsymbol{\theta^{\pi'}}$;
    $t \leftarrow t+1$;
    $\Gamma = \epsilon^{\text{Min}} + (\epsilon^{\text{Max}} - \epsilon^{\text{Min}}) / \exp(-\epsilon^d t)$;
  until $\sigma_s(t) \leq \tau_s, \forall s$ or maximum iterations;
end for

Indeed, the issue about the high dimensional observation space can be resolved using a DQN-based approach, however, it is not designed to operate on continuous action space. In our work, we have two control variables including power and computation resources that are continuous-valued. One approach that can be employed is to quantize all continuous values of the network states and discretize the continuous action space. However, the performance of such a solution is highly dependent on the level of quantization and discretization and can significantly affect the stability and convergence of the network. Therefore, we choose to use the policy gradient method, which is a popular choice for such continuous domain problems. Specifically, we employ the actor-critic method [40], which can be also used with DNNs to search for the optimal control policy.

B. DRL-Based Actor-Critic Algorithm

An actor-critic approach consists of an actor-network and a critic network. This approach can be used to estimate the value function of the policy in a policy-based approach. Typically, the actor takes an action based on the policy and the critic estimates the value obtained by taking the action. In this work, we use the deep deterministic policy gradient approach in which DNN is used to approximate the parameters of the actor $\pi(\boldsymbol{n}; \boldsymbol{\theta^\pi})$ and the critic $Q(\boldsymbol{n}, \mathbf{x}; \boldsymbol{\theta^Q})$ network. Then, the aim of the actor-critic method is to take the best action based on the current state and to model the correlation among the state-action pair via the critic network. The aforementioned DQN

can be used as a critic function $Q(\boldsymbol{n}, \mathbf{x}; \boldsymbol{\theta^Q})$ and trained via the loss function presented in (30) by using the following as a target value:

$$\beta[t] = (1-\lambda)E(\boldsymbol{n}[t], \boldsymbol{x}[t]) + \lambda Q'\Big(\boldsymbol{n}[t+1], \pi'\big(\boldsymbol{n}[t+1]|\boldsymbol{\theta^{\pi'}}\big); \boldsymbol{\theta^{Q'}}\Big), \quad (31)$$

where $Q'()$ and $\pi'()$ represent the target networks with the weights $\boldsymbol{\theta^{Q'}}$ and $\boldsymbol{\theta^{\pi'}}$, respectively. Note that the target networks follow the same structure as their original counter part.

Next, we discuss the actor network. The goal of the actor-network is to choose an action based on a deterministic policy. Then, the actor parameters are directly adjusted using the gradient method such that it minimizes the objective by stepping in the direction of the gradient of the policy. In this work, the objective is represented by $G(\theta^\pi) = \mathbb{E}\big[E(n, x)\big]$ and its gradient by $\nabla_{\boldsymbol{\theta^\pi}} G$. Then, the actor network's parameters can be updated via the chain rule by the expected return from the start of distribution G [41] as follows:

$$\nabla_{\boldsymbol{\theta^\pi}} G \approx \mathbb{E}\Big[\nabla_{\boldsymbol{\theta^\pi}} Q(\boldsymbol{n}[t], \mathbf{x}[t]|\boldsymbol{\theta^Q})|_{\boldsymbol{x}[t]=\pi(\boldsymbol{n}[t]|\boldsymbol{\theta^\pi})}\Big] = \mathbb{E}\Big[\nabla_{\boldsymbol{x}} Q(\boldsymbol{n}[t], \mathbf{x}[t]|\boldsymbol{\theta^Q})|_{\boldsymbol{x}[t]=\pi(\boldsymbol{n}[t])} \nabla_{\boldsymbol{\theta^\pi}} \pi(\boldsymbol{n}[t]|\boldsymbol{\theta^\pi})\Big]. \quad (32)$$

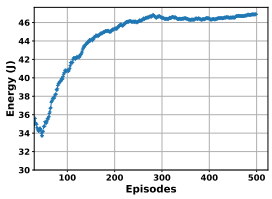
Algorithm 1 presents the DRL-based task offloading algorithm for mobile vehicular networks aiming at energy minimization. The proposed algorithm starts by random selection of weights both for the actor and critic networks represented by $\boldsymbol{\theta^\pi}$ and $\boldsymbol{\theta^Q}$, respectively. Additionally, two target networks are also created with the same structure to provide stability and the weights of these networks are copied from the original corresponding networks (Initialization phase, line 1). Next, in the learning phase, we start the training process, which is comprised of a number of episodes, each episode is completed with a variable number of steps as it depends upon the uplink, execution, and downlink transmission of the offloaded task. In each episode, the environment and all variables pertaining to the system are first reset (lines 4-5). Then, the actor-network $\pi()$ will take an action after observing the current state $\boldsymbol{n}[t]$. Note that, noise is added to the action taken by the actor-network in order to keep exploring the action space (lines 7-8). We represent it by Γ, which keeps reducing as time proceeds (line 16). Then, each entity in the system (S-EV, D-EV, and RSU) executes the taken action and move from its current state $\boldsymbol{n}[t]$ to the next state $\boldsymbol{n}[t+1]$, which results in accumulated energy consumption of all EVs presented in (21). This energy consumption is then associated with a reward value, specifically, we use an inversely proportional reward value of energy consumption per successfully offloaded tasks. Then, the agent's learning part initiates in which the agent will update the weights of both the original and target actor-critic networks (lines 11-14). Moreover, we also employ the experience replay technique through, which agents can learn from early memories stored in the memory pool. Therefore, after each transition step, the experience is stored in the memory pool and then a random mini-batch of transition is used to update both the learning networks (lines 9-10).

	Parameter	Value
Simulation Parameters	Radius of RSU	500 m
	Task data size	$1 \sim 10$ Mb
	Task deadline	$3 \sim 8$ s
	Transmission power of S-EV	23 dBm
	Communication range of Evs	100 m
	Noise Power	-174 dBm
	Path loss exponent	4
Hyper-parameters	Actor Network	256, relu, 256, relu,256, relu, sigmoid
	Critic Network	256, relu, 256, relu,256, relu, linear
	Actor LR	10^{-3}
	Critic LR	10^{-3}
	Soft Target update	10^{-2}
	Batch Size	256
	Memory size	10^6
	Discount factor	0.995

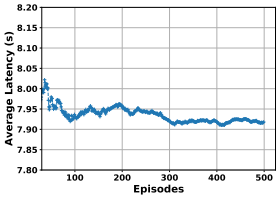
TABLE II: Parameters

Specifically, actor network's weight $\boldsymbol{\theta^\pi}$ is updated using the gradient presented in (32) while the critic network's weight $\boldsymbol{\theta^Q}$ is updated by minimizing a loss function in equation (30). Note that, the target networks' weights are also updated slowly with a controlled updating rate κ (lines 13-14). Finally, the EV task offloading algorithm terminates either once all tasks have been successfully completed or the algorithm runs for the maximum number of steps (line 17). The later condition typically states that enough resources were not available in the mobile vehicular network to offload the S-EV's task. This can stem from multiple reasons such as no D-EV available in the near vicinity, not enough resources at the RSU, higher computation, and data size requirements, high relative mobility, etc. Note that, the proposed approach can easily be adopted to support a multi-vendor setting in which computing resources can be sold, priced, and bought from different vendors opposed to a single vendor. Indeed, message passing between seller and agent is required to provide information and make decisions at the agent. Note that this can be done by incorporating incentive-based solutions tailored for vehicular network on top of the proposed DRL-based solution such as ruin theory [42], [43], auction theory [44], matching theory [45], and contract theory [13], [24].

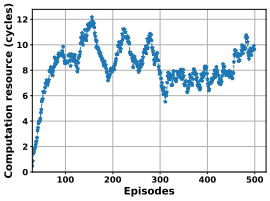
Finally, we would also like to discuss the practical implementation issues pertaining to Algorithm 1. Note that as the number of EVs in the system increase converging to a stationary control policy can be extremely slow and would require long training times. This can be challenging and can significantly incur performance degradation. One approach that can be employed to overcome this challenge is to pre-train the model in an offline manner via centralized powerful servers at the RSU with a large number of EVs in the system similar to the works in [35]. Then, the trained model can be adopted for a smaller number of EVs by setting the extra parameters related to states and actions to null. This approach reduces the complexity of the proposed approach and can also cope with scalability issues of EVs in the system.



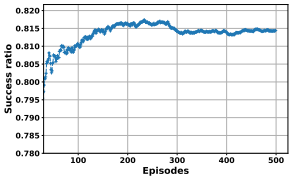
(a) Average Sum Energy



(b) Average Latency



(c) Computation resource



(d) Task success ratio

Fig. 2: System performance with respect to number of episodes.

IV. Numerical Results

This section presents our simulation results to evaluate the performance of our proposed scheme. First, we present our simulation setting followed by the performance evaluation of the proposed scheme.

A. Simulation Setup

We consider a service area of 500 meters in which an RSU is placed at the coordinate $[250, 0]$ and EVs are randomly following the homogeneous Poisson point process (HPPP) in a given service area. For this simulation, we consider 8 S-EVs and 8 D-EVs for evaluating the task offloading scheme. Moreover, both types of EVs are traveling in a uniform direction with constant acceleration. Next, for the DNN, actor-critic network, we have three fully connected layers that employ rectified linear unit (ReLU) as the activation function for its hidden layers with 256 neurons in each layer. For the output layer, we employ the Sigmoid activation function. Additionally, a replay buffer of capacity $1e6$ is also employed from which a mini-batch of 256 size is chosen. Finally, we set the discount factor to 0.995, and the learning rate to 0.001 for both networks. The main system parameters and chosen hyper-parameters setting for the DNN are listed in Table II. To implement the proposed algorithm, we employ a powerful open-source software library, PyTorch 1.4, developed by Facebook's AI Research lab (FAIR), and Tensorflow 1.9.0, developed by the Google Brain team.

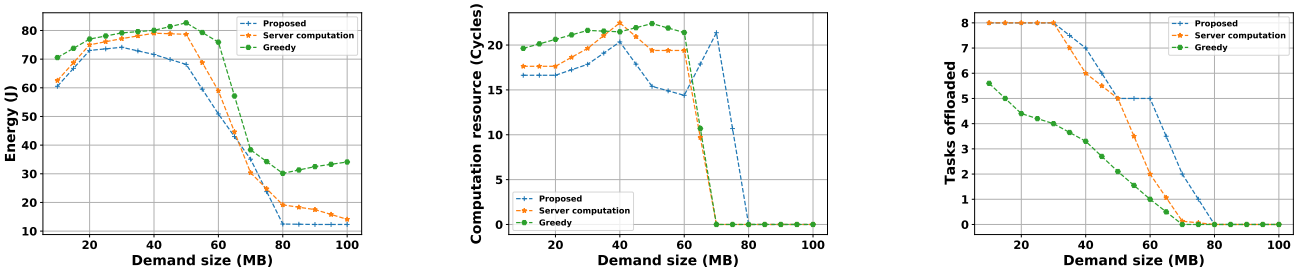
Next, for evaluating the proposed solution, we compare our results with two baselines. The first baseline is the same learning approach without the capability of performing computation at the neighboring EVs, namely "Server computation". In this approach, we use the same structure of DNN and use the same actor-critic-based learning approach to find the best power levels and amount of computing resources required at the RSU for performing the task offloading service. The second baseline is the commonly used "Greedy" approach in which all decisions are taken based on maximizing the task offloading service, a destination, which provides the best transmission rate for task offloading, best power for transmission, and maximum computation resource. This approach does not look at the long-term dynamics of the system. Note that we cannot compare our results with an optimal solution because of the scale and stochastic behavior stemming from the mobility of the proposed problem.

B. Results analysis

In this section, we present our results analysis. First, we present the results of the convergence of the proposed algorithm. In this simulation, we show the convergence of the network's sum energy, computational resource, latency, and task offloading success ratio. We run this simulation for 500 episodes for 8 S-EVs, each with task offloading requirement of size uniformly distributed in the range of $1 \sim 10$ (MB) and task deadline set in the range of $1 - 3$ seconds. Moreover, we assume the there are 8 D-EVs in the network with a compute capacity uniformly distributed in the range of $10 \sim 50$ (MB). Moreover, each EV is randomly placed in the coverage area and is traveling with a random speed uniformly distributed in the range of $10 \sim 50$ (km/h) in the same direction.

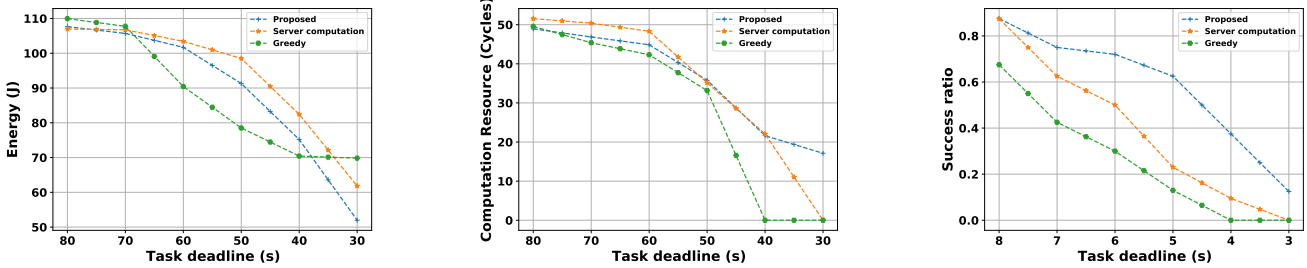
Figure 2 presents the convergence analysis. It can be seen after 250 episodes, the average sum energy, average latency values, average computation resource values, and task success rate all converge. This indicates that the proposed solution achieves a stable policy once converged. At the starting episodes from the Figure 2a, we can see that the average sum of energy is consumed less due to lack of task offloading taking place, evident from Figure 2d. However, after reaching 250 episodes, the algorithm starts to stabilize reflecting a stable policy that will converge. Thus, the performance of the proposed solution based on the policy will improve over time due to the learning mechanism in which the actor and critic networks will update their respective networks' weight as the number of episodes increases. Figure 2b and Figure 2c also follow the same trend in which as the learning progresses, more computation resources are being used because of the higher number of EVs task being offloaded resulting in more average sum latency. Moreover, we can also observe that the shape of the curve about the computation resource (as shown in Figure 2c) is not as smooth as the other results even after convergence. The main reason is the randomness stemming from the EVs' mobility, which makes this result relatively more sensitive to mobility compared to the other results.

Figure 3 represents our system performance results, when we vary the demand sizes at each S-EV. In this simulation, we keep the same parameters as discussed for the previous simulation setting and vary the demand size of each EV. Moreover, we also compare the performing with both the baselines to reflect the performance in terms of average sum energy in joules (as shown in Figure 3a), average computation resource in terms of CPU cycles required to execute the task (Figure 3b), and the average number of task successfully offloaded (as shown



(a) Average Sum Energy (b) Average Computation resource (c) Number of Task Success

Fig. 3: System performance with respect to different demand sizes.



(a) Average Sum Energy (b) Average Computation resource (c) Number of Task Success

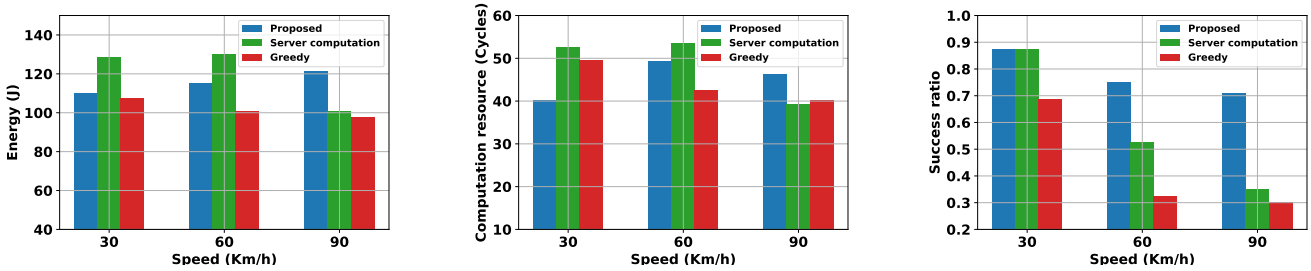
Fig. 4: System performance with respect to different maximum task deadline threshold.

in Figure 3c) for the proposed and baselines approaches. It can be seen that as the task size increases from 10 (MB) to 100 (MB), the average energy consumption reduces, which seems inappropriate for all approaches. However, we can see that the energy increases when the demand size reaches up to 50 (MB) and then it suddenly drops drastically. The main reason for this drop in energy is the fact that the vehicular resources are not enough and at 50 (MB) onwards they become saturated and no further tasks can be offloaded.

This is also evident from Figure 3b and Figure 3c. It can be seen that after this point, the computation resource is not enough to execute the demands within the required time threshold, and therefore, the task offloading service fails resulting in a low number of tasks being successfully offloaded. Moreover, we can also observe that the proposed scheme is significantly better compared to the baselines in terms of energy consumed and a number of tasks successfully offloaded while the greedy approach performs the worst among all schemes. Furthermore, the proposed approach is the most resilient to changes in task data sizes providing the best energy per task ratio making it the best choice. A performance gain of up to 13% and 33% is observed in terms of energy consumption when compared to the server computation baseline and greedy baselines respectively. Similarly, the performance gain significantly increases when we compare the energy per successful task completed in which a gain of up to 65% and 86% is observed for the demand size of 60 (MB) when compared to the server computation baseline and greedy baselines, respectively. In Figure 4, we present the performance analysis of the proposed scheme by varying the task deadline. In this simulation, each S-EV chooses a task of size randomly from the range of $50 \sim 120$ (MB) and we vary the task deadline by decreasing it from 8 seconds to 3 seconds. We present the average energy consumption (Figure 4a), average computational resources (Figure 4b), and average success ratio (Figure 4c) of the offloaded tasks for the proposed and baseline approaches. Evidently, from Figure 4a, the energy consumption reduces as the task deadline is tightened due to the fact that fewer tasks are being successfully offloaded.

Similarly, the average computation resource Figure 4b also reduces as the deadlines become more and more strict reflecting less number of task success ratio presented in Figure 4c. Indeed, the proposed approach performs the best in comparison to the baselines by maximizing the number of successful tasks offloaded in the system. Alternatively, the greedy approach performs the worst among all approaches providing the lowest task completion ratio and resulting in wastage of energy as the deadline is not met. We can observe a performance gain of up to 31% and 53% even with a loose threshold of 6 seconds in terms of energy consumed per successful task completed when compared to the server computation baseline and greedy baseline, respectively. Moreover, we can also see that the energy consumed at 6 second for the server computation baseline is almost indistinguishable and in the case of greedy baseline, it is 12% lower compared to the proposed scheme, however, these approaches are not efficient in terms of successfully completing the offloaded task.

Figure 5 presents the impact of varying the speed in the systems. In this simulation, we keep the same settings as our previous simulation presented in Figure 4 and each EV chooses the task deadline threshold randomly from a range of $5-8$ seconds. In this simulation, we increase the speed from 30



(a) Average Sum Energy (b) Average Computation resource (c) Number of Task Success

Fig. 5: System performance with respect to different maximum EVs speed.

(km/h) to 90 (km/h) to observe the impact of average energy consumption (Figure 5a), average computational resources (Figure 5b), and average success ratio (Figure 5c) of the offloaded tasks for the proposed and baseline approaches. We can see that the average sum energy from the vehicular network increase as the speed increases for the proposed scheme as shown Figure 5a. The main reason for this increase in average energy values is that more power is required if the inter S-EV and destination distance increase, which depends upon the speed.

Additionally, more computation resources are also required as the speed increases to abide by the task deadline threshold also evident from Figure 5b). Furthermore, we also see that the task success ratio decreases with speed. The main reason is that the task deadline is violated for some of the EVs at higher speed, thus, resulting in a lower success rate. On the other hand, the server computation baseline's energy also increases as the speed increases from 30 (km/h) to 60 (km/h), however, it drops at 90 (km/h). The main reason is the S-EV and RSU distance, which significantly affects the task success rate. Evidently, from Figure 5c), we can see that the success rate drops significantly at a higher speed. The greedy baseline observes the worst performance in terms of task success rate. Finally, it can be concluded that the proposed approach significantly outperforms the other baselines in terms of average energy consumption, average resource utilization, and average task success rate for the offloading service.

V. Conclusions

In this paper, a task offloading mechanism for mobile vehicular networks has been proposed using a DRL-based approach. The DRL-based proposed approach empowers us to capture the stochastic characteristics of the environment stemming from the nature of practical vehicular networks. The proposed approach effectively achieves the stationary control policy for the proposed time-varying dynamic proposed problem. Numerical results reveal that the proposed solution significantly outperforms the server computation and greedy approach baselines and enhances the area computational capacity while reducing the energy computation per task offloaded. In the future, we intend to extend this approach to multi-RSU settings where inter RSU task-offloading can play a significant role by splitting the offloaded task and achieve parallel execution of the task.

References

[1] D. Sperling, *Future drive: Electric vehicles and sustainable transportation*. Island Press, February 2013.

[2] Z. Wei, Y. Li, and L. Cai, "Electric vehicle charging scheme for a park-and-charge system considering battery degradation costs," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 3, pp. 361–373, June 2018.

[3] T. M. Ho, N. H. Tran, L. B. Le, Z. Han, S. A. Kazmi, and C. S. Hong, "Network virtualization with energy efficiency optimization for wireless heterogeneous networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 10, pp. 2386–2400, 2019.

[4] W. Zhang, B. Han, and P. Hui, "On the networking challenges of mobile augmented reality," in *Proc. of the Workshop on Virtual Reality and Augmented Reality Network*, August 2017,Los Angeles CA USA, pp. 24–29.

[5] S. M. A. Kazmi, T. N. Dang, I. Yaqoob, A. Ndikumana, E. Ahmed, R. Hussain, and C. S. Hong, "Infotainment Enabled Smart Cars: A Joint Communication, Caching, and Computation Approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8408–8420, July 2019.

[6] I. Yaqoob, L. U. Khan, S. M. Ahsan, M. Imran, N. Guizani, and C. S. Hong, "Autonomous Driving Cars in Smart Cities: Recent Advances, Requirements, and Challenges," *IEEE Network*, August 2019.

[7] R. Hussain and S. Zeadally, "Autonomous Cars: Research Results, Issues, and Future Challenges," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1275–1313, September 2018.

[8] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, June 2009.

[9] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, June 2016.

[10] L. U. Khan, I. Yaqoob, N. H. Tran, S. M. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge-computing-enabled smart cities: A comprehensive survey," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 200–10 232, April 2020.

[11] T. M. Ho, T. D. Tran, T. T. Nguyen, S. Kazmi, L. B. Le, C. S. Hong, and L. Hanzo, "Next-generation wireless solutions for the smart factory, smart vehicles, the smart grid and smart cities," *arXiv preprint arXiv:1907.10102*, 2019.

[12] T. H. Ashrafi, M. A. Hossain, S. E. Arefin, K. D. J. Das, and A. Chakrabarty, "Service based fog computing model for iot," in *2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC)*, 15-17 Oct. 2017, San Jose, CA, USA, pp. 163–172.

[13] T. N. Dang, K. Kim, L. U. Khan, S. M. A. Kazmi, Z. Han, and C. S. Hong, "On-device computational caching-enabled augmented reality for 5g and beyond: A contract-theory-based incentive mechanism," *IEEE Internet of Things Journal*, vol. 8, no. 24, pp. 17 382–17 394, May 2021.

[14] G. Premsankar, B. Ghaddar, M. Di Francesco, and R. Verago, "Efficient placement of edge computing devices for vehicular applications in smart cities," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 23-27 April 2018,Taipei, Taiwan, pp. 1–9.

[15] Z. Ning, J. Huang, X. Wang, J. J. P. C. Rodrigues, and L. Guo, "Mobile Edge Computing-Enabled Internet of Vehicles: Toward Energy-Efficient Scheduling," *IEEE Network*, vol. 33, no. 5, pp. 198–205, October 2019.

[16] G. Qiao, S. Leng, K. Zhang, and Y. He, "Collaborative Task Offloading in Vehicular Edge Multi-Access Networks," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 48–54, August 2018.

[17] C. Zhu, G. Pastor, Y. Xiao, Y. Li, and A. Ylae-Jaeaeski, "Fog Following Me: Latency and Quality Balanced Task Allocation in Vehicular Fog Computing," in *Proc. 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 11-13 June 2018, Hong Kong, China, pp. 1–9.

[18] F. Lin, X. Lü, I. You, and X. Zhou, "A Novel Utility Based Resource Management Scheme in Vehicular Social Edge Computing," *IEEE Access*, vol. 6, pp. 66 673–66 684, October 2018.

[19] X. Huang, R. Yu, J. Liu, and L. Shu, "Parked Vehicle Edge Computing: Exploiting Opportunistic Resources for Distributed Mobile Applications," *IEEE Access*, vol. 6, pp. 66 649–66 663, November 2018.

[20] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *Proc. IEEE International Conference on Communications (ICC)*, 21-25 May 2017,Paris, France, pp. 1–6.

[21] Z. Su, Q. Xu, Y. Hui, M. Wen, and S. Guo, "A Game Theoretic Approach to Parked Vehicle Assisted Content Delivery in Vehicular Ad Hoc Networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 7, pp. 6461–6474, November 2016.

[22] K. Zhang, Y. Mao, S. Leng, A. Vinel, and Y. Zhang, "Delay constrained offloading for Mobile Edge Computing in cloud-enabled vehicular networks," in *Proc. 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, 13-15 Sept. 2016, Halmstad, Sweden, pp. 288–294.

[23] K. Zhang, Y. Mao, S. Leng, S. Maharjan, A. Vinel, and Y. Zhang, "Contract-theoretic approach for delay constrained offloading in vehicular edge computing networks," *Mobile Networks and Applications*, vol. 24, no. 3, pp. 1003–1014, February 2018.

[24] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation Resource Allocation and Task Assignment Optimization in Vehicular Fog Computing: A Contract-Matching Approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3113–3125, January 2019.

[25] J. Zhao, M. Kong, Q. Li, and X. Sun, "Contract-Based Computing Resource Management via Deep Reinforcement Learning in Vehicular Fog Computing," *IEEE Access*, vol. 8, pp. 3319–3329, December 2019.

[26] M. Chen, W. Saad, and C. Yin, "Liquid state machine learning for resource allocation in a network of cache-enabled lte-u uavs," in *Proc. 2017 IEEE Global Communications Conference*. IEEE, 4-8 Dec. 2017,Singapore, pp. 1–6.

[27] H. Ye and G. Y. Li, "Deep reinforcement learning for resource allocation in V2V communications," in *Proc. IEEE International Conference on Communications (ICC)*. IEEE, 20-24 May 2018,Kansas City, MO, USA, pp. 1–6.

[28] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, 20-24 May 2018,Kansas City, MO, USA.

[29] L. Liang, H. Ye, and G. Y. Li, "Spectrum sharing in vehicular networks based on multi-agent reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2282–2292, August 2019.

[30] X. Zhang, M. Peng, S. Yan, and Y. Sun, "Deep-Reinforcement-Learning-Based Mode Selection and Resource Allocation for Cellular V2X Communications," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6380–6391, December 2019.

[31] K. Wang, X. Wang, X. Liu, and A. Jolfaei, "Task Offloading Strategy Based on Reinforcement Learning Computing in Edge Computing Architecture of Internet of Vehicles," *IEEE Access*, vol. 8, pp. 173 779–173 789, September 2020.

[32] Q. Qi and Z. Ma, "Vehicular edge computing via deep reinforcement learning," *arXiv preprint arXiv:1901.04290*, February 2020.

[33] S. Chai and V. K. Lau, "Online trajectory and radio resource optimization of cache-enabled UAV wireless networks with content and energy recharging," *IEEE Transactions on Signal Processing*, vol. 68, pp. 1286–1299, February 2020.

[34] S. M. A. Kazmi, T. N. Dang, I. Yaqoob, A. Manzoor, R. Hussain, A. Khan, C. S. Hong, and K. Salah, "A novel contract theory-based incentive mechanism for cooperative task-offloading in electrical vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–16, May 2021.

[35] T. M. Ho and K.-K. Nguyen, "Joint Server Selection, Cooperative Offloading and Handover in Multi-access Edge Computing Wireless Network: A Deep Reinforcement Learning Approach," *IEEE Transactions on Mobile Computing*, pp. 1–1, December 2020.

[36] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, November 2018.

[37] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, May 1992.

[38] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, February 2015.

[39] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, J. Pineau *et al.*, "An introduction to deep reinforcement learning," *Foundations and Trends® in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, March 2019.

[40] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour *et al.*, "Policy gradient methods for reinforcement learning with function approximation," in *NIPs*, vol. 99. Citeseer, November 1999, pp. 1057–1063.

[41] 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," *arXiv preprint arXiv:1509.02971*, July 2019.

[42] A. Manzoor, K. Kim, S. R. Pandey, S. M. A. Kazmi, N. H. Tran, W. Saad, and C. S. Hong, "Ruin theory for energy-efficient resource allocation in uav-assisted cellular networks," *IEEE Transactions on Communications*, vol. 69, no. 6, pp. 3943–3956, 2021.

[43] A. Manzoor, N. H. Tran, W. Saad, S. M. A. Kazmi, S. R. Pandey, and C. S. Hong, "Ruin theory for dynamic spectrum allocation in lte-u networks," *IEEE Communications Letters*, vol. 23, no. 2, pp. 366–369, December 2018.

[44] D. H. Kim, S. M. A. Kazmi, A. Ndikumana, A. Manzoor, W. Saad, and C. S. Hong, "Distributed radio slice allocation in wireless network virtualization: Matching theory meets auctions," *IEEE Access*, vol. 8, pp. 73 494–73 507, 2020.

[45] S. M. A, N. H. Tran, W. Saad, Z. Han, T. M. Ho, T. Z. Oo, and C. S. Hong, "Mode selection and resource allocation in device-to-device communications: A matching game approach," *IEEE Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3126–3141, November 2017.

S. M. Ahsan Kazmi is currently a Senior Lecturer in the Department of Computer Science and Creative Technologies, University of the West of England, Bristol. Prior to this, he worked as an Assistant Professor with the Institute of Information Security and Cyber-Physical System, Innopolis University, Russia for three years. He has also worked as a Postdoctoral fellow, from 2017 to 2018 at the Department of Computer Science and Engineering, Kyung Hee University, South Korea, from where he completed his Ph.D. degree, in 2017. His research interests include applying analytical techniques of optimization, machine learning, and game theory to radio resource management for future networks. He received the best KHU Thesis Award in engineering in 2017 and several best paper awards from prestigious conferences.

Tai Manh Ho received the B.Eng. and M.S. degrees from the Hanoi University of Science and Technology, Vietnam, in 2006 and 2008, respectively, and the Ph.D. degree from Kyung Hee University, South Korea, in 2018, all in computer engineering. Since 2019, he has been with the Synchromedia Lab, École de technologie supérieure, Université du Québec, Montréal, QC, Canada where he is currently a postdoctoral fellow. His current research interests include radio resource management and enabling technologies for 5G wireless systems.

Tuong Tri Nguyen received the B.S. degree in Physics-Informatics from Hue University in 1997. He received M.S. in Computer Science from Hue University in 2004 and Ph.D. degrees in Computer Engineering from the School of Computer Engineering, Yeungnam University, Korea in 2016. Currently, he is a professor in Computer Engineering at Hue University. His main research interest includes social network services, big data analytics, data mining, and machine learning. He has served as a member of the program committee of International Conferences such as ACIIDS, ICCCI, KSE.

Muhammad Fahim received his B.S. with distinction from Gomal University, Pakistan in 2007. He got M.S. from the National University of Computer and Emerging Sciences (NUCES), Pakistan in 2009. He got his Ph.D. degree from Kyung Hee University, South Korea in February 2014. He worked as Post-Doctoral Fellow at the Department of Computer Engineering, Kyung Hee University, South Korea. He served as an Assistant Professor in the Department of Computer and Software Engineering, Faculty of Engineering and Natural Sciences, Istanbul Sabahattin Zaim University, Istanbul, Turkey for 3 years. He also led the Machine Learning research laboratory in Istanbul Sabahattin Zaim University. He worked as an Assistant Professor at the Institute of Information Systems, Innopolis University, Innopolis, Russia for four years. Currently, he is a lecturer in School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, UK. His research interests include machine learning models, wearable computing, digital signal processing, and behavior recognition in intelligent environments.

Adil Khan received his B.S. degree in Information Technology from National University of Sciences and Technology (NUST), Pakistan in 2005. He completed his M.Sc. and Ph.D. degrees in Computer Engineering from Kyung Hee University, South Korea in 2011. He is currently a Full Professor at the Institute of Artificial Intelligence and Data Science, Innopolis University, Russia. He has 16+ years of research experience in the field of Artificial Intelligence and Machine Learning. He is also the Head of Machine Learning & Knowledge Representation Lab at Innopolis. He has lead and delivered several industry funded research projects from SamSung, LG, and SK Telecom. He has published more than 80 research articles in prestigious conferences and journals.

Md. Jalil Piran received his Ph.D. degree in Electronics and Information Engineering from Kyung Hee University, South Korea, in 2016.Then, he continued his research carrier as a Post-Doctoral Fellow in Information and Communication Engineering with the Networking Laboratory, KyungHee University. Dr. Jalil Piran is currently an Assistant Professor with the Department of Computer Science and Engineering, Sejong University, Seoul, South Korea. Prof. Piran published a substantial number of technical papers in well-known international journals and conferences in the area of Intelligent Information and Communication Technology (IICT), specifically in the fields of: Wireless Communications and Networking; 5G/6G, Internet of Things (IoT), Data Science, Machine Learning, Security. In the worldwide communities, he is a Senior Member of IEEE, an Active Delegate from South Korea in the Moving Picture Experts Group (MPEG) since 2013, and an Active Member of the International Association of Advanced Materials (IAAM) since 2017. Prof. Piran received the IAAM Scientist Medal of the year 2017 for notable and outstanding research in new age technology and innovation, Stockholm, Sweden. He has been recognized as the Outstanding Emerging Researcher by the Iranian Ministry of Science, Technology, and Research in 2017.Also, his Ph.D. dissertation has been selected as the "Dissertation of the Year 2016" by the Iranian Academic Center for Education, Culture, and Research in the Engineering Group.

Baye Gaspard received his BSc with honors from the Information and Communication Technology (ICT) university, Yaoundé, in 2016. He was awarded the best overall graduating student in ICT university with more than seven awards, one from the president. He is currently a Ph.D. student and Research Assistant at the University of Massachusetts Cybersecurity lab, A NSA/DHS designated Center of Academic Excellence in Cyber Defense Research(CAER). His research focuses on applying Machine Learning techniques to improve user privacy, optimize cellular networks, secure databases, and application security by speeding up vulnerability detections and mitigations in modern systems.