Collaborative Reinforcement Learning for Multi-Service Internet of Vehicles

Collaborative Reinforcement Learning for Multi-Service Internet of Vehicles

Swapnil Sadashiv Shinde, *Student Member, IEEE*, and Daniele Tarchi, *Senior Member, IEEE*

***Abstract*—Internet of Vehicles (IoV) is a recently introduced paradigm aiming at extending the Internet of Things (IoT) toward the vehicular scenario in order to cope with its specific requirements. Nowadays, there are several types of vehicles, with different characteristics, requested services, and delivered data types. In order to efficiently manage such heterogeneity, Edge Computing facilities are often deployed in the urban environment, usually co-located with the Roadside Units (RSUs), for creating what is referenced as Vehicular Edge Computing (VEC). In this paper, we consider a joint network selection and computation offloading optimization problem in multi-service VEC environments, aiming at minimizing the overall latency and the consumed energy in an IoV scenario. Two novel collaborative Q-learning based approaches are proposed, where Vehicle-to-Infrastructure (V2I) and Vehicle-to-Vehicle (V2V) communication paradigms are exploited, respectively. In the first approach, we define a collaborative Q-learning method in which, through V2I communications, several vehicles participate in the training process of a centralized Q-agent. In the second approach, by exploiting the V2V communications, each vehicle is made aware of the surrounding environment and the potential offloading neighbors, leading to better decisions in terms of network selection and offloading. In addition to the tabular method, an advanced deep learning-based approach is also used for the action value estimation, allowing to handle more complex vehicular scenarios. Simulation results show that the proposed approaches improve the network performance in terms of latency and consumed energy with respect to some benchmark solutions.**

***Index Terms*—Internet of Vehicles, Computation Offloading, Network Selection, Reinforcement Learning.**

I. Introduction

With the increasing advancements of the Internet of Things (IoT) and evolved wireless technologies, numerous new applications and services have emerged in vehicular networks in recent times. This has brought to the Internet of Vehicle (IoV) paradigm, where each node composing the urban mobility environment can be seen as a source of data, similarly to the traditional *Things* [1]. In recent times, smart vehicles with advanced features have been added to vehicular networks. Different types of vehicles are characterized by several innovative applications, like autonomous driving, image-aided navigation, road safety, and augmented reality (AR), having the potential to improve the performance of transportation systems [2]. Despite their increased on-board computational capabilities, vehicles are not able to provide the massive amount of computation resources requested by new applications. Therefore, offloading computation-intensive tasks to cloud facilities has became a necessity [3]. However, cloud facilities are located far away from the end-users, introducing a high end-to-end delay.

Multiaccess Edge Computing (MEC) is considered a viable option by enabling processing facilities closer to the end-users, attracting lots of attention from academia and industries [4]. In wireless networking scenarios, MEC has enabled latency-critical and data-intensive applications by allowing mobile devices to offload portions of their computation loads to the edge servers in the proximity [3]. Vehicular Edge Computing (VEC) is a recently introduced paradigm aiming at integrating the MEC potentialities in vehicular scenarios by deploying several edge servers co-located with the Roadside Units (RSUs) [5]. Similarly to MEC, in VEC environments computation tasks can be executed locally, at the vehicle, or offloaded to the nearby RSUs, acting as edge servers, allowing for improved Quality of Service (QoS) in terms of latency and consumed energy [6]. Due to the dynamic nature of vehicular networks, offloading a large amount of data to the RSUs without considering the vehicular mobility could seriously degrade the QoS. Indeed, with high mobility, each vehicle has a limited amount of time available for data offloading and collecting the results from the RSUs. If the vehicle passes through the RSU coverage without completing the offloading operation, it might end up paying higher latency costs due to, e.g., handover and service migration [7]. Moreover, each VEC server can offer a limited number of services to the nearby vehicular nodes (VNs), hence, selecting the proper edge server for offloading can avoid network congestion. Therefore, a proper selection of the RSU and the computation offloading amount in a VEC-empowered IoV environment is an important problem to be considered.

Recently, many new and exciting Machine Learning (ML) techniques have been introduced to solve challenging research problems in vehicular networks [2], [8]. Among others, Reinforcement Learning (RL) is a proper candidate for solving the VEC network selection and computation offloading problem, allowing to learn at run-time from the dynamic environment [9]. In addition, several new communication paradigms, including Vehicle to Vehicle (V2V), Vehicle to Infrastructures (V2I), and Vehicle to Pedestrians (V2P), have been introduced in vehicular networks [10]. Exploiting these communication paradigms for understanding the local environment in vehicular networks could lead to better solutions for network selection and computation offloading problems. As a matter of

The authors are with the Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi", University of Bologna, Italy, email: {swapnil.shinde2,daniele.tarchi}@unibo.it

fact, the possibility of exchanging information among different IoV nodes have been recently exploited in several studies to propose high-end solutions for complex vehicular problems. Some examples are computation offloading [11], collaborative computing [12] and data sharing [13].

The scenario under consideration is a multi-service multi-user VEC-enabled IoV network composed of several VNs and RSUs and one Macro Base Station (MBS) able to cover the whole area. Since VNs have limited computation and communication resources, they aim at exploiting the nearby RSUs/MBS for offloading computational tasks assuming that they can act as VEC nodes. In general, each vehicle can be covered by several RSUs, while each RSU can serve multiple VNs. Moreover, each RSU is able to provide a limited number of services. Therefore, finding a proper VN-RSU pair can improve the performance of a resource-constrained vehicular network. If a VN cannot find a suitable RSU node, it can select MBS as a possible option for data offloading. However, in such a case, VNs might need to pay additional costs in terms of latency and energy requirements due to long-distance communication. At the same time, offloading an optimal amount of data to the selected RSU/MBS can further increase the energy and latency performance.

A. Literature Review

The forthcoming intelligent IoV wonderland will generate tones of computation data, and VEC can be a viable solution for providing the computation services to the resource-constrained vehicular nodes [14]. However, for having the benefits of VEC resources over multi-user IoV networks, selecting a proper edge server and offloading amount is an important problem. In the past, several authors have tried to solve the computation offloading problem by either finding a proper edge node or the amount to be offloaded. In [15], authors have proposed the adaptive task offloading strategy in the MEC-based vehicular networks environment with a pre-allocation algorithm for vehicle tasks. Such approach, however, can lead to limited performance since offloading the incorrect amount towards an ideal edge server, or the correct amount towards the wrong edge node, can not guarantee optimal performance. In [16], the authors focus on an energy-efficient approach for computation offloading in VEC networks. Two heuristic approaches are proposed for solving the problem under different configurations. In [17], authors have studied a reliable computation offloading and task allocation problem over integrated fixed and mobile edge computing enabled vehicular users through the adaptation of software to define networking technology. In [18], the authors proposed a multi-armed bandit approach for optimally selecting the network to be used for computation offloading. In this case, both online and offline approaches are considered. In [7] the authors propose a energy-constrained approach for managing in-vehicle device offloading operations. The problem is solved through a consensus based approach. In [19], the authors propose a framework for implementing a joint federated learning and computation offloading operation where a High Altitude Platform is used as central node. The problem is formulated as an utility function and solved through a genetic algorithm.

Various works have highlighted the importance of using ML for solving the computation offloading related problems. In [20], a multi-agent DRL-based approach is developed to solve the computation offloading problem over the fog computing enabled industrial IoT system. However, the offloading performance is only measured in terms of energy efficiency, while the latency performance is neglected. In addition, tasks can either be computed locally or at the access point (i.e., binary offloading strategy). In [21], authors have proposed a multi-agent RL-based computation offloading strategy for the vehicular scenario. The latency performance of the offloading process is optimized with binary computation offloading strategy. Different from others, a two-stage meta learning-based ML model selection algorithm is proposed in [22], for minimizing the edge node resource consumption cost from a consumer's point of view. Also, in some cases, authors have focused on either the latency or energy performance of a computation offloading system. In [23], the authors have considered a task offloading problem in the vehicular scenario in which VNs can learn each other offloading delay performance. An adaptive learning-based task offloading strategy based on the multi-arm bandit theory is proposed to reduce the average offloading delay. In [24], the authors have considered an RL-based strategy and have used the Markov decision process-based model to solve it. However, they have only focused on the latency performance of a system. In another case, [25] proposed an energy-efficient workload offloading scheme over VNs. In [26], the authors propose a fast-heuristic method for solving the vehicle to RSU mapping in an energy limited scenario.

The importance of utilizing vehicular data for solving vehicular problems is highlighted by several authors. In [27], authors have studied the IoV data collection, transmission, and diffusion problem for vehicular traffic-related information. In particular, a novel architecture is proposed for the IoV aided local traffic information collection, a sink node selection scheme for the information flow, and an optimal traffic information transmission model. In another case, [28], the authors integrate the cooperative vehicular sensing capabilities in the smart city paradigm. Optimal data source selection algorithm and a novel RL-based information sharing strategy are studied with possible applications, challenges, and future directions.

In order to have a more comprehensive view of the different computational offloading strategies analyzed in this Section, we report in Table I a comparison based upon goals, target variables, offloading approaches, service models, mobility scenarios, and solutions methods. It also highlights the lack of solutions for the joint network selection and computation offloading problem over the multi-service vehicular scenario as the one proposed here.

B. Paper Contribution

Differently from other papers, we aim at jointly optimizing the RSU selection and the computation offloading to minimize the network-wide latency and energy consumption. To this end, the problem is modeled as a cost function and solved through an RL approach. Differently from other RL-based

TABLE I
LITERATURE REVIEW COMPARISON

Reference	Goal	Target	Offloading Approach	Service Model	Mobility	Solution Method
[3]	Offloading decision and computation resource allocation	Utility (based on task processing delay, computation resource cost, additional normalization factor) maximization with delay constraint	Binary	Single service	Vehicular Mobility with varying speed	Two step optimization, Offloading decision - Game theory, resource allocation - Lagrange multiplier method
[7]	Offloading and resource allocation decisions	Energy consumption with delay constraint	Partial	Multiple tasks (generated by user equipment's inside vehicle)	Vehicular mobility with varying speed	Iterative optimization with Alternating Direction Method of Multipliers (ADMM) and nonlinear fractional programming
[9]	Offloading and resource allocation decisions	Users Quality of Experience (QoE)	Binary	Single service	Vehicular mobility data	Offloading task scheduling - matching theory, resource allocation -double DQN
[15]	Joint network selection and task offloading	Overall latency minimization	Multiple task units	Single service	Vehicular mobility with varying speed	Adaptive task offloading strategy
[16]	Offloading strategy and resource allocation	System cost (communication and computation Cost with latency constraint)	Partial	Single service	Vehicular mobility model	Mobility aware (for independent servers) and location based (for cooperative servers) offloading strategy
[17]	Offloading strategy (partial offloading, task allocation, and task reprocessing)	Reliability (based on task processing latency and application latency constraint)	Partial	Single service	Vehicular mobility model	Fault-tolerant particle swarm optimization algorithm
[18]	Offloading node selection (network and base station selection)	Latency cost	Partial	Single service	Vehicular mobility model	Multi-Armed Bandit (MAB) theory based On-line and Off-policy learning algorithms
[19]	Offloading strategy with FL performance	Joint latency and energy cost	Partial	Single service	Vehicular mobility model	Clustered and distributed optimization
[20]	Offloading strategy (Fog Access Point (FAP) selection and request forwarding)	Energy cost	Binary	Single service	Fixed devices	DRL based FAP selection
[21]	Offloading node selection	Latency cost	Binary	Single service	Vehicular mobility model	Multiagent DRL based node selection
[23]	Offloading node selection	Latency cost	Partial	Single service	Vehicular mobility model	Adaptive learning-based task offloading algorithm based on MAB
[24]	Time dependent offloading node selection	Latency cost	Partial	Single service	Vehicular mobility model	Time aware MDP-based task offloading algorithm
[25]	Energy-efficient workload offloading	Energy cost	Partial	Multiple applications	Vehicular mobility model	ADMM-based workload offloading algorithm
[29]	Mobility aware partial offloading	Outage probability and latency cost	Partial	Single service	Vehicular mobility model	D2D communication assisted partial offloading strategy
[30]	Joint computation load balancing And offloading	QoS-based utility function (based on latency cost)	Partial	Single service	Vehicular mobility model	Joint node selection, computation resource and Offloading (JSCO) algorithm
[31]	Joint offloading and resource allocation	System utility cost	Binary	Single service	Vehicular mobility model	Q learning and DRL

solutions in VEC scenarios [9], we propose here a new collaborative approach. By gaining from different vehicular communication paradigms, i.e., V2V and V2I, allowing information exchange among nodes [10], we aim at a better understanding of the local environment to be used for the development of two RL solutions.

In the first approach, a V2I collaborative Q-learning method is considered, in which VNs participate in the training process of centralized Q-agents. In the second approach, each VN is made aware of the nearby environment, and the potential offloading neighbors through the V2V links, leading to better decisions. A more advanced deep learning-based approach is also considered to estimate the action value functions for both the Q learning approaches allowing the possible extension towards high dimensional scenarios. During the learning phase, several scenarios are considered based on the VNs local environment. In the end, the numerical results show that the proposed solutions provide better latency and energy performance for end-users with respect to other benchmark methods.

The remaining parts of this paper are structured as follows. Section II introduces the system model and defines the optimization problem to be solved. Next, in Section III, the proposed solution methods are described, including two Q-learning based solutions, two Deep Q-Network based solutions and one Heuristic solution. In Section IV the numerical results obtained through computer simulations are provided and analyzed. Finally, in Section V the conclusions are drawn.

II. SYSTEM MODEL AND PROBLEM FORMULATION

The IoV scenario under consideration is composed of a set $\mathcal{V} = \{VN_1, \ldots, VN_m, \ldots, VN_M\}$ of M VNs, and a set $\mathcal{R} = \{RSU_1, \ldots, RSU_n, \ldots, RSU_N\}$ of N RSUs, creating the urban vehicular service. In addition, one MBS able to cover the whole area is supposed. Both RSUs and MBS act as edge nodes providing EC services to the VNs, enabling computation-intensive applications and services at the edge. The IoV system is modeled in a time-discrete manner, and the network parameters are constant over each

time interval τ, where τ_i identifies the ith time interval, i.e., $\tau_i = \{\forall t | t \in [i\tau, (i+1)\tau]\}$.

Each VN is equipped with communication, computing, and storage elements, where it is supposed that it can communicate with a maximum bandwidth b_m and can process with a maximum computational capability η_m. In addition, by focusing on the ith time interval, it is supposed to have a battery capacity E_m^c and a battery level $E_m(\tau_i)$. The mth VN is supposed to be located in the position $\{x_m(\tau_i), y_m(\tau_i)\}$, while it moves at a speed $\vec{v}_m(\tau_i)$ along the road paths, where:

$$\vec{v}_m(\tau_i) = \frac{(x_m(\tau_i), y_m(\tau_i)) - (x_m(\tau_{i-1}), y_m(\tau_{i-1}))}{\tau}, \ i \geq 1.$$

Each RSU can be identified through a set of parameters, where the nth RSU, located at the position $\{x_n^R, y_n^R\}$, can provide communication with a maximum bandwidth B_n, and can process with a maximum computational capability H_n. Similarly, the MBS can be identified through its position $\{x^M, y^M\}$, maximum bandwidth B^M, and maximum computational capability H^M.

We consider that RSUs and MBS compose a multi-service network able to provide multiple services to the IoV environment. By assuming that $\mathcal{S} = \{S_1, \ldots, S_s, \ldots, S_S\}$ is the set of all the possible services that can be provided, due to the limited available resources, each RSU can provide only a subset of services. Thus, for the RSU_n we can identify $\mathcal{S}_n \subseteq \mathcal{S}$ as the set of services provided by it. Since the MBS has more resources, it is supposed to offer the whole service set $\mathcal{S}$. Finally, the nth RSU is supposed to have a limited coverage range d_n, whose value depends on the communication technology and radio-propagation environment, and it is supposed to provide VEC services to the vehicles within the coverage area. Similarly, for the MBS, the coverage range d^M stands. Each $VN_m \in \mathcal{V}$ is supposed to be active in each time interval with a probability p_a within which it generates a computation task request $\rho_m(\tau_i)$ identified through the tuple $\langle D_{\rho_m}, D^r_{\rho_m}, \Omega_{\rho_m}, T_{\rho_m}, S_{\rho_m} \rangle$ corresponding to a task with size D_{ρ_m} Byte, expected to give in output a result with size $D^r_{\rho_m}$ Byte, requesting Ω_{ρ_m} CPU execution cycles, with a maximum execution latency T_{ρ_m} and requesting service S_{ρ_m}.

A. RSU Selection

We define a binary VN-RSU assignment matrix $\mathbf{A}(\tau_i) = \{a_{m,n}(\tau_i)\}_{M \times N} \in \{0,1\}$ with size $M \times N$, where $a_{m,n}(\tau_i) = 1$ if VN_m is assigned to RSU_n in the interval τ_i, and $\sum_{n=1}^{N} \sum_{m=1}^{M} a_{m,n}(\tau_i) = M$, imposing that each VN is able to offload data to only one RSU[1]. Moreove, $a_{m,n}(\tau_i) = 1 \iff S_{\rho_m} \in \mathcal{S}_n$ that is to say the assignment can occur only if the requested service has been deployed on the nth RSU.

We assume to perform partial offloading, that is to say, each task generated by the VNs can be split, and a portion remotely processed while the remaining is processed locally [29]; the portion offloaded by VN_m at τ_i is identified as $\alpha_{\rho_m}(\tau_i) \in [0,1]$. Here, $\alpha_{\rho_m}(\tau_i) = 0$ corresponds to the complete local processing of the task, while $\alpha_{\rho_m}(\tau_i) = 1$ to the complete offloading of the task to the selected VEC node. During the partial offloading process, the task processing operations, performed locally by VNs and remotely at the RSU-based edge servers, are supposed to be executed in parallel to reduce the overall processing time [30]. Each RSU is supposed to have a limited amount of computation and communication resources; hence, it can only serve a limited number of users. Therefore:

$$\begin{cases} \sum_{m=1}^{M_n(\tau_i)} \alpha_{\rho_m}(\tau_i) \cdot \Omega_{\rho_m} \leq H_n^R \cdot \tau \quad \forall i & \text{(1a)} \\ \sum_{m=1}^{M_n(\tau_i)} b_m(\alpha_{\rho_m}(\tau_i), D_{\rho_m}) \leq B_n^R \quad \forall RSU_n \in \mathcal{R}, i & \text{(1b)} \end{cases}$$

where, $M_n(\tau_i) = \sum_{m=1}^{M} a_{m,n}(\tau_i)$ is the number of vehicles assigned to the RSU n in the ith interval. While (1a) models an upper bound on the processing capacity of the RSU, (1b) introduces a transmission capacity upper bound for the VNs connected to any RSU. Here, $b_m(\alpha_{\rho_m}(\tau_i), D_{\rho_m})$ corresponds to the communication resources available for the transmission of vehicular tasks depending upon its task size and the offloading parameter. It is worth to be noticed that the capacity of each link depends on the specific communication technology and it is out of the scope of this paper.

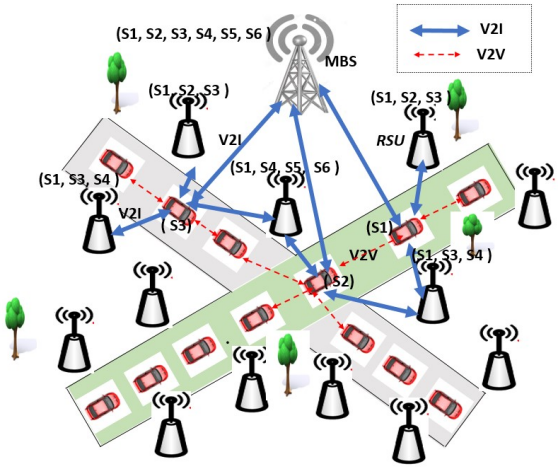

Fig. 1. The multi-service IoV system architecture.

In Fig. 1, a possible IoV scenario is depicted, where different VNs request different services, and are covered by RSUs hosting different service types. In addition, VNs are supposed to be in the coverage area of the MBS.

B. Task Processing

The processing time and energy needed for computing a task depend on the offloading policy and the selected node processing characteristics. The generic expression for the time and energy spent for the ρ_mth task computation on any device is given by [32]:

$$T_{c_l}^{\rho_m} = \frac{\Omega_{\rho_m}}{o_l f_l} \quad \text{(2a)}$$

$$E_{c_l}^{\rho_m} = k_l \frac{\Omega_{\rho_m}}{o_l} f_l^2 \quad \text{(2b)}$$

[1]Given the complex nature of the considered problem, especially over a dynamically changing vehicular environment, we have assumed that each vehicle can select only one edge node for offloading its data.

where o_l and f_l are the number of Floating-point Operation Per Second (FLOPS) per CPU-cycle and CPU-frequency, respectively, whether l is a generic index identifying one of the possible processing nodes among VNs (m), RSUs (n) and MBS. In (2b), k_l is a constant coefficient representing the chip architecture of the generic lth device.

Since we assume to perform a partial computation offloading, each VN transmits a portion of its task to the assigned RSU and receives back the result. In general, the transmission time and energy between VN_m and RSU_n for task ρ_m is given by:

$$T^{\rho_m}_{tx,mn} = \frac{D_{\rho_m}}{r_{mn}} \tag{3a}$$

$$E^{\rho_m}_{tx,mn} = T^{\rho_m}_{tx,mn} P t_m \tag{3b}$$

where r_{mn} is data-rate of the link between the two nodes, while $P_{tx,m}$ is the transmission power of VN_m. Similarly, the reception time and energy to receive back the processing result having size $D^r_{\rho_m}$ from RSU_n by VN_m are, respectively:

$$T^{\rho_m}_{rx,nm} = \frac{D^r_{\rho_m}}{r_{mn}} \tag{4a}$$

$$E^{\rho_m}_{rx,nm} = T^{\rho_m}_{rx,nm} P_{rx,m} \tag{4b}$$

where $P_{rx,m}$ is the power spent for receiving at the RSU_m side. A symmetric channel is considered between VN_m and RSU_n. The expression for the channel transmission rate is based on the Shannon capacity formula and can be written as:

$$r_{mn} = b_m \log_2 \left(1 + \frac{P t_m}{L(d_{mn}) N_0}\right)$$

where $P_{tx,m}$ is the transmission power of a device m, $L(d_{mn})$ is the path loss at a distance d_{mn}, and $N_0 = N_T b_m$ is the thermal noise power.

1) Task Offloading Process: If VN_m is assigned to RSU_n, then the time and energy required to offload the task to the selected RSU and to get back the result in the ith interval is:

$$T^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)) = \alpha_{\rho_m}(\tau_i) \left(T^{\rho_m}_{tx,mn} + T^{\rho_m}_{c_n} + T^{\rho_m}_{rx,nm}\right) \tag{5a}$$

$$E^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)) = \alpha_{\rho_m}(\tau_i) \left(E^{\rho_m}_{tx,mn} + E^{\rho_m}_{rx,nm}\right) \tag{5b}$$

where $T^{\rho_m}_{tx,mn}$, $T^{\rho_m}_{c_n}$, and $T^{\rho_m}_{rx,mn}$ are, respectively, the transmission time, computation time on nth RSU, and the receiving time for the task ρ_m generated by VN_m during offloading phase, and $E^{\rho_m}_{tx,mn}$ and $E^{\rho_m}_{rx,mn}$ are, respectively, the energy consumed during the task transmission and result collection phases on device. Since RSU nodes are supposed to be connected to the electrical grid, we do not consider their energy consumption in the energy analysis.

2) Local Computation: The amount of time and energy required for the local computation in the ith interval is:

$$T^{loc}_m(\alpha_{\rho_m}(\tau_i)) = (1 - \alpha_{\rho_m}(\tau_i))\, T^{\rho_m}_{c_m} \tag{6a}$$

$$E^{loc}_m(\alpha_{\rho_m}(\tau_i)) = (1 - \alpha_{\rho_m}(\tau_i))\, E^{\rho_m}_{c_m} \tag{6b}$$

where $T^{\rho_m}_{c_m}$ and $E^{\rho_m}_{c_m}$ are the time and energy spent for the whole task ρ_m local processing, while $\alpha_{\rho_m}(\tau_i)$ is the portion of the task locally processed at the time interval τ_i.

3) Partial Offloading Computation: The delay and the energy consumed during the task processing phases, when partial offloading is performed in the ith interval, can be written as:

$$T^{\rho_m}_m(\alpha_{\rho_m}(\tau_i)) = \max\left\{T^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)), T^{loc}_m(\alpha_{\rho_m}(\tau_i))\right\}$$
$$E^{\rho_m}_m(\alpha_{\rho_m}(\tau_i)) = E^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)) + E^{loc}_m(\alpha_{\rho_m}(\tau_i)).$$

Since the local computation and offloading processes are executed in parallel, the total task processing latency is the maximum of the two.

C. Vehicle Mobility and Sojourn Time

The VN mobility poses some constraints to the computation offloading decisions. Due to the VNs mobility, each offloading operation should be completed by the VN sojourn time, corresponding to the amount of time it remains under the coverage of the selected RSU [29], otherwise the system may be affected by additional latency due to, e.g., vehicle handover, service migration, additional signaling for managing vehicles and services mobility [7]. The remaining distance in which the mth VN remains in the coverage of nth RSU is:

$$D_{m,n}(\tau_i) = \sqrt{d_n^2 - (y_n - y_m(\tau_i))^2} \pm (x_n - x_m(\tau_i)) \tag{7}$$

where $\{x_m(\tau_i), y_m(\tau_i)\}$ and $\{x_n, y_n\}$ are, respectively, the position of the mth VN and the nth RSU at time interval τ_i and R_n is the coverage radius of the nth RSU. Hence, the sojourn time for the mth VN can be written as:

$$T^{soj}_{m,n}(\tau_i) = \frac{D_{m,n}(\tau_i)}{|\vec{v}_m(\tau_i)|} \quad \forall i \tag{8}$$

Each vehicle should finish the offloading process and receive the results back within the sojourn time, hence:

$$T^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)) \leq T^{soj}_{m,n}(\tau_i) \quad \forall i \tag{9}$$

D. Problem Formulation

The main aim of this work is to optimize the network-wide performance of the VEC-enabled IoV network. We aim to optimize the performance in terms of overall latency and energy consumed during the offloading process towards edge servers by selecting proper RSU nodes and offloading amounts. For this, we formulate the joint latency and energy minimization problem as:

$$\mathbf{P1}:$$
$$\min_{\mathcal{A},\mathbf{A}} \left\{ \sum_{n=1}^{N} \sum_{m=1}^{M} \left[\gamma_1 T^{\rho_m}_m\left(\alpha_{\rho_m}(\tau_i)\right) + \gamma_2 E^{\rho_m}_m\left(\alpha_{\rho_m}(\tau_i)\right)\right] \right\} \forall i \tag{10}$$

s.t.

$$\mathbf{C1}: \sum_{n=1}^{N} \sum_{m=1}^{M} a_{m,n}(\tau_i) = M \tag{11}$$

$$\mathbf{C2}: a_{m,n}(\tau_i) = 1 \iff S_{\rho_m} \in \mathcal{S}_n \quad \forall i \tag{12}$$

$$\mathbf{C3}: \text{Eqs. (1a) and (1b)} \tag{13}$$

$$\mathbf{C4}: T^{\rho_m}_m\left(\alpha_{\rho_m}(\tau_i)\right) \leq T_{\rho_m} \quad \forall VN_m \in \mathcal{V},\ \forall i \tag{14}$$

$$\mathbf{C5}: \text{Eq. (9)} \tag{15}$$

$$\mathbf{C6}: E_m^{\rho_m}(\alpha_{\rho_m}(\tau_i)) \leq E_{c_m}^{\rho_m} \tag{16}$$

$$\mathbf{C7}: 0 \leq \gamma_1, \gamma_2 \leq 1;\ \gamma_1 + \gamma_2 = 1 \tag{17}$$

where $\mathcal{A} = \{\alpha_{\rho_m}\}^M$ is the computation offloading matrix, and γ_1, γ_2 are two weighting coefficients for balancing latency and energy consumption. $\mathbf{C1}$ stands that each VN can select at most one RSU for the computation offloading. According to $\mathbf{C2}$, the selected edge node must be able to provide the service requested by the VNs. $\mathbf{C3}$ sets the bounds in terms of processing capacity and resource blocks requested by VNs towards edge nodes, while $\mathbf{C4}$ puts a limit on the maximum processing time as one of the task requirements. According to $\mathbf{C5}$, for avoiding handover phenomena and related latency, each VN should complete the offloading process before it passes through the selected RSUs coverage. According to $\mathbf{C6}$, the total energy required for the task processing during the partial computation offloading process should be bounded by the energy needed to process the complete task locally. $\mathbf{C7}$ stands that the two weighting coefficients (γ_1, γ_2) should be between 0 and 1 with their sum equal to 1.

III. Q-Learning Based Joint Network Selection and Computation Offloading

The decision on the EN to be selected for computation offloading and the amount of data to be offloaded can depend upon several factors, such as VNs position and speed, nearby VNs, the available number of RSUs for offloading, availability of the requested service, etc. If a VN is under the coverage of multiple ENs, the proper EN can be selected by sequentially testing ENs one after another. Also, VNs can make sequential decisions for finding a proper amount of data to be offloaded towards the EN. Every decision taken by VNs can alter the surrounding environment's state, and can be mapped with a reward (i.e, an increase or decrease in the task processing time and energy). Therefore, finding the proper EN and the corresponding data to be offloaded in the dynamic vehicular environment can be considered a sequential decision-making problem that can effectively be solved through the RL approach.

RL and multi-agent RL (MARL)-based methods have found applications in vehicular scenarios. For avoiding the unbearable training costs and for adequate training performance, various RL/MARL training architectures are considered in the past. A centralized MARL training process having an exponentially scalable set of actions and observation spaces, is often neglected in highly complicated vehicular environments. In the case of a fully decentralized approach, an independent set of agents tries to optimize a common reward function over its local environment. Issues like spurious rewards, mainly due to partial observability, prevent their use in the considered problem. Due to the involvement of high-speed VNs and a dynamically changing environment, a large set of training agents are required for solving highly complex problems such as the one considered here. This scales up the issues of exponential scalability and spurious rewards in these two traditional approaches.

In recent times, some other MARL architectures have been proposed, mainly for solving the previous challenges. One such example is [33], where the authors have proposed a value-decomposition network-based cooperative MARL architecture. A deep neural network-based back-propagation approach is used to decompose a common value function into a set of agent-based value functions. However, the complexity of the considered decomposition network, the limited use of state information during training, and applicability towards a reduced class of centralized value functions are bottlenecks. Also in [34], another value function-based approach is considered by estimating a joint action-value function from a set of local observation-based action individual agents values through neural networks. The issues of complexity, scalability, etc., prevent the use of this MARL-based method for solving the given problem.

With the availability of novel communication technologies such as V2V, V2I, etc, VNs can share important environmental parameters. Such information can be integrated into the training process of a collaborative RL strategy to solve the given vehicular problem effectively and with a reduced complexity with respect to a MARL approach. Therefore, below, we have proposed efficient vehicular communication-based cooperative RL strategies for solving the problem at hand. In RL, at any given time τ, an intelligent agent in a particular state $St(\tau)$ interacts with the dynamic environment En, through the selected action $ak(\tau)$, and, in return, receives observations in terms of a state change $St^*(\tau)$ and rewards R. The agent tries to maximize the future reward value over consecutive discrete time steps by taking the actions from the current state in the environment based on the received observations [35]. Therefore, state-space, action-space, and the reward function are the main elements of the RL process.

In the considered RL-based framework, multiple agents collaboratively perform the training operations for finding the optimal policy aimed at minimizing the joint latency and energy cost with reliable network selection and computation offloading operations. We have considered a centralized training architecture assisted by vehicular communication data for improving the training performance. MBS, as a centralized entity, can perform the training process for individual training scenarios (i.e., RL agents) for finding the optimal policies. It collects the information from VNs and uses it during the learning process of the RL agents by implementing a collaborative learning process. More description about the learning scenarios is provided below in subsections III-A1 and III-A2.

The State-space, Action-space, and Reward function can be identified as follows.

1) State-Space ($\mathcal{ST}$): In a multi-service multi-user vehicular environment the available resources for the computation offloading process are changing continuously over time and are function of the offloading and network selection decisions taken by vehicles. Therefore, we have defined a state-space composed of a discrete set of states as a function of resources available for computation offloading. The state-space is a discrete set of states identifying the RSUs to be selected and their resources available for the computation offloading. Since each

VN_m can exploit a different number of RSUs for offloading, we define $\mathcal{R}_m = \{RSU_n | D_{m,n}(\tau_i) > 0 \wedge S_{\rho_m} \in \mathcal{S}_n, \forall \mathcal{R}\}$ as the set of RSUs available for the mth VN for the offloading operation; we consider a multi-dimensional state-space representation where the νth state-space corresponds to the *scenario* with ν available RSUs. For each scenario the related state-space is function of sojourn time, required latency, VN resources, resources of the available RSUs; thus, each state St_ν at time τ_i is defined as:

$$St_\nu(\tau_i) = f(\alpha_{\rho_m}(\tau_i), T^{soj}_{m,n}(\tau_i), B_n, H_n, D_{\rho_m}, D^r_{\rho_m}, \Omega_{\rho_m}, T_{\rho_m}, S_{\rho_m}).$$

We suppose to limit the multi-dimensional state space to $\bar{N}$ scenarios, hence, $\nu = 1, \ldots, \bar{N}$.

2) Action-Space ($\mathcal{AS}$): The action space defines all the possible actions available during the learning process of an RL-agent. We consider to have $\bar{N}$ agents collecting information for each possible scenario, composed by a different number of available RSUs. At each iteration, each agent explores the available RSUs, by properly setting a binary vector $\mathbb{R}_\nu(\tau_i) = \{0,1\}^\nu$ mapping the RSUs selection among the ν available in the given scenario. At the same time, the offloaded amount is selected, by either increasing, decreasing, or keeping the same amount as the previous iteration. Thus, for the τ_ith instance, we have $\alpha_{\rho_m}(\tau_i) \in \{\alpha_{\rho_m}(\tau_i - 1), \alpha_{\rho_m}(\tau_i - 1) \pm \Lambda\}$ where Λ is a step increase or decrease of the offloading amount. The generic action ak_ν for the νth scenario at time τ_i can be defined as $ak_\nu(\tau_i) = \{\mathbb{R}_\nu(\tau_i), \alpha_{\rho_m}(\tau_i)\}$ where $\mathbb{R}(\tau_i)$ is a binary vector with length ν, where 1 in the nth position corresponds to the selected RSU.

3) Reward Function (R): The reward function (R) is defined as the joint objective function of time and energy consumed for the complete task processing (10). In addition, three penalty terms are also considered modeling when the agent fails to satisfy the latency and energy constraints, as defined in (14), (9) and (16). Thus, the expression for the reward function is given by:

$$\begin{aligned} R(St_\nu(\tau_i), ak_\nu(\tau_i)) = \\ \gamma_1 T^{\rho_m}_m(\alpha_{\rho_m}(\tau_i)) + \gamma_2 E^{\rho_m}_m(\alpha_{\rho_m}(\tau_i)) \\ + \Upsilon_1 \cdot \max(0, C_1(St_\nu(\tau_i), ak_\nu(\tau_i))) \\ + \Upsilon_2 \cdot \max(0, C_2(St_\nu(\tau_i), ak_\nu(\tau_i))) \\ + \Upsilon_3 \cdot \max(0, C_3(St_\nu(\tau_i), ak_\nu(\tau_i))) \end{aligned} \quad (18)$$

where Υ_1, Υ_2 and Υ_3 are the weighting coefficients for the penalty values, and:

$$C_1(St_\nu(\tau_i), ak_\nu(\tau_i)) = T^{\rho_m}_m(\alpha_{\rho_m}(\tau_i)) - T_{\rho_m} \quad (19a)$$

$$C_2(St_\nu(\tau_i), ak_\nu(\tau_i)) = T^{off}_{m,n}(\alpha_{\rho_m}(\tau_i)) - T^{soj}_{m,n}(\tau_i) \quad (19b)$$

$$C_3(St_\nu(\tau_i), ak_\nu(\tau_i)) = E^{\rho_m}_m(\alpha_{\rho_m}(\tau_i)) - E^{\rho_m}_{c_m}. \quad (19c)$$

(19a) is the additional penalty value when VN fails to perform the task processing operation within the maximum execution latency bound, (19b) is the additional penalty when VNs fails to satisfy the sojourn time bounds during the offloading process, and (19c) is the additional cost when VN fails to follow the energy constraint in (16).

A. Collaborative Q-Learning Solutions for Joint Network Selection and Offloading

Q-learning is one of the most-known techniques used to solve RL-based problems [31]. In Q-learning, each state-action pair (St, ak) has a Q value, defined as $Q(St, ak)$, which provides the expected future reward for an agent from state St if he decides to take action a. Each agent receives the input values as set of environment state ($\mathcal{ST}$) including the possible terminal state St_{opt}, reward function R and action set ($\mathcal{AS}$). Other input parameter includes the learning rate $\gamma_{lr} \in \{0,1\}$, discount factor $\Delta \in \{0,1\}$, and a number of learning episodes E. In every episode, the agent selects the initial state St_{in}, then it takes a random action ak over the environment, receives a reward related to that action and a new state value. The widely known Epsilon-Greedy algorithm (EGA) is used for selecting the future action [35]. In EGA, each Q-agent picks the action based upon the Exploration vs Exploitation dilemma, with the exploration probability e. For every iteration, the Q-values ($Q(St_t, ak_t)$) for the state-action pair (St_t, ak_t) are updated based upon the temporal difference expression defined as:

$$\begin{aligned} Q^{new}(St^t_\nu, ak^t_\nu) \leftarrow Q(St^t_\nu, ak^t_\nu) + \gamma_{lr}\big(R_{St^t_\nu, ak^t_\nu} + \\ \Delta\left(\max_A\left\{Q(St^{t+1}_\nu, ak^{t+1}_\nu)\right\} - Q(St^t_\nu, ak^t_\nu)\right)\big) \end{aligned} \quad (20)$$

Two novel Q-learning based solutions for the joint selection of network and computation offloading problem are considered here. In the first method, a V2I collaborative Q-learning-based solution is considered, in which several randomly distributed VNs participate in the training process of a Q-agent. In the beginning, a centralized server collects data from VNs by exploiting the V2I communication links. Next, vehicles are classified based upon the local environment scenarios. In the end, every group trains the Q-agent based on its local environment. In the other approach, each VN explores the V2V communication links for collecting information about the nearby VNs. This allows to make a better decisions in terms of network selection and computation offloading.

1) V2I Assisted Collaborative Q-Learning (V2I-AC): In the V2I collaborative approach, multiple randomly located VNs participates to the training process towards a centralized set of agents through the V2I links. Since the cardinality of $\mathcal{R}_m$ depends on VN_m and RSUs positions as well the deployed services, we define multiple training scenarios, each one characterized by the number of available RSUs, i.e., $|\mathcal{R}_m|$. The scenario vector $\mathcal{SV}_{rsu}$ includes all possible available RSUs for computation offloading, i.e., $\mathcal{SV}_{rsu} = \{2, \cdots, RSU_{vmax}\}$. Here, RSU_{vmax} is the maximum number of RSU nodes available for computation offloading. In the Q-learning process, we generate $\bar{N}$ Q-tables where, $Q_\nu(\mathcal{ST}, \mathcal{AS})$ is the table of the νth scenario.

By referring to the pseudocode in Algorithm 1, in the V2I collaborative Q-learning approach, the VNs are first classified based on their scenario (lines 1-3). For each VN_m, the cardinality of available RSUs are evaluated, i.e., $|\mathcal{R}_m|$. In order to limit the number of possible scenarios to $\bar{N}$, the mth VN is managed by the νth agent, where $\nu = \min\{|\mathcal{R}_m|, \bar{N}\}$. Through this, we can classify all VNs into different groups,

Algorithm 1 V2I Assisted Collaborative Q-Learning (V2I-AC)

Input: Set of VNs $\mathcal{V}$, $\bar{N}$, $e, \gamma_{lr}, \Delta, \mathcal{ST}, \mathcal{AS}, R, \mathcal{I}$
Output: $\{Q(\mathcal{ST}, \mathcal{AS})\}$
1: **for all** $VM_m \in \mathcal{V}$ **do**
2: Find $\mathcal{R}_m$ and set $\nu = \min\{|\mathcal{R}_m|, \bar{N}\}$, $\bar{m}_\nu \leftarrow \bar{m}_\nu + 1$
3: **end for**
4: **for all** $\nu = 1, \ldots, \bar{N}$ **do**
5: **for all** VN_m such that $|\mathcal{R}_m| = \min(\nu, \bar{N})$ **do**
6: Select a random initial state $St^0_\nu \in \mathcal{ST} \wedge St^0_\nu \neq \bar{St}_\nu$
7: $St_\nu \leftarrow St^0_\nu, \quad it = 0$
8: **while** $St^t_\nu \neq \bar{St}_\nu || it = \mathcal{I}$ **do**
9: $it = it + 1$
10: Select action $ak_\nu \in \mathcal{AS}$ with probability e
11: Determine next state (St^{t+1}_ν) and reward received
12: Use Eq. (20) to find the TD and update Q table.
13: $St_\nu \leftarrow St^{t+1}_\nu$
14: **end while**
15: **end for**
16: **return** $Q_\nu(\mathcal{ST}, \mathcal{AS})$
17: **end for**

with νth scenario having $\bar{m}_\nu$ VNs for the collaborative training process, with $\bar{m}_\nu = \{|VN_m| : |\mathcal{R}_m| = \min(\nu, \bar{N}), \forall \mathcal{V}\}$. After the VNs classification, the training process for each scenario is performed (lines 4-16). The number of training episodes for each scenario is equal to the number of VNs in that particular scenario group, i.e., $\bar{m}_\nu$. In each episode, the agent selects a non-optimal random initial state St^0_ν, then it takes a random action ak_ν over the environment, receives a reward and a new state value. For every iteration, the Q-values $(Q(St^t_\nu, ak^t_\nu))$ for the state-action pair (St^t_ν, ak^t_ν) are updated following the the Temporal Difference (TD) expression (20). Once the final state $\bar{St}_\nu$ or $it = \mathcal{I}$ has been reached, with $\mathcal{I}$ being a maximum number of iterations, the agent starts the new episode of learning. This process is repeated till a predefined optimal state is reached. In the end, we receive a Q-table associated with that particular scenario $(Q_{SV_{rsu}})$.

In the V2I collaborative Q-learning method, we aim to find the joint solution for the network selection and computation offloading problem. The solution is composed of a network selection vector $\mathbb{R}_\nu$, depending on the scenario in which any given VN are classified, and the offloading amount α_{ρ_m}. In case none of the surrounding RSUs have the service requested by VN_m, i.e., $\mathcal{R}_m = \emptyset$, VN_m can offload the data towards the MBS, which contains all the services requested by VNs.

2) V2V Assisted Collaborative Q-Learning (V2V-AC): In this approach, we suppose that the VNs exploit V2V communication links for acquiring knowledge of the potential competing VNs around them before offloading for the training phase. Since the VNs offloading strategy is unknown by each VN we assume that a predefined VN-RSU selection method where each competing VN selects the nearest RSU node for complete task offloading is assumed for resource allocation purposes. Both communication and computation resources of RSU nodes are equally shared among the assigned VNs. Similarly to the V2I approach, we consider different agents acting on different scenarios, where, in this case, we extend also to the number of nearby VNs. Different learning scenarios are considered based upon the number of available RSUs for offloading and the nearby VNs. Therefore, while the number of available RSUs is determined in the same way, we extend the considered scenarios up to $\bar{N} \cdot \bar{M}$ where $\bar{M}$ is the maximum number of nearby VNs to be considered.

By defining $\mathcal{V}_m = \{RSU_{m'} | d_{m,m'}(\tau_i) \leq d_{V2V}, \forall \mathcal{R}\}$ as the set of VNs within a certain d_{V2V} coverage distance, we can set as $\mu = \min\{|\mathcal{V}_m|, \bar{M}\}$ as the scenario identifying all the vehicles with μ surrounding VNs to be exploited. Given this, the (ν, μ)th agent will exploit as training nodes all the VNs having ν available RSUs and μ competing VNs.

The number of available RSUs could be in the range of 2 to RSU_{vmax}. Similarly, the number of VNs around each test VN is ranged between 0 to VN_{vmax}. Thus, the two scenario vectors are $\mathcal{SV}_{rsu} = [2, \cdots, RSU_{vmax}]$ and $\mathcal{SV}_{vn} = [0, \cdots, VN_{vmax}]$. Therefore, each $SV(i,j) = \{\mathcal{SV}_{rsu}(i), \mathcal{SV}_{vn}(j)\}$ represents the training scenario considered.

Algorithm 2 V2V Assisted Collaborative Q-Learning (V2V-AC)

Input: Set of VNs $\mathcal{V}$, $\bar{M}, \bar{N}, e, \gamma_{lr}, \Delta, \mathcal{ST}.\mathcal{AS}, R, \mathcal{I}$
Output: $\{Q(\mathcal{ST}, \mathcal{AS})\}$
1: **for all** $VN_m \in \mathcal{V}$ **do**
2: Find $\mathcal{R}_m$ and $\mathcal{V}_m$
3: Set $\nu = \min\{|\mathcal{R}_m|, \bar{N}\}$ and $\mu = \min\{|\mathcal{V}_m|, \bar{M}\}$
4: $\bar{m}_{(\nu,\mu)} \leftarrow \bar{m}_{(\nu,\mu)} + 1$
5: **end for**
6: **for all** $\nu = 1, \ldots, \bar{N}$ **do**
7: **for all** $\mu = 1, \ldots, \bar{M}$ **do**
8: Follow steps from 5 to 14 of Algorithm 1
9: **return** $Q_{(\nu,\mu)}(\mathcal{ST}, \mathcal{AS})$
10: **end for**
11: **end for**

The Algorithm 2 describes the steps used during the training phase of each scenario. In lines 1-5, each VN scenario is determined, by estimating the number of available RSUs for data offloading $\mathcal{R}_m$ and the number of VNs around it ($\mathcal{V}_m$). Based on their training scenario, each VN will be classified into different groups. At the end of the training sessions, each scenario will have a separate Q-table. Similar to the previous approach, the solution is composed of network selection vector $\mathbb{R}_\nu$ and the offloading amount α_{ρ_m}. In case any of the surrounding RSUs have the requested service by the VN_m, i.e., $\mathcal{R}_m = \emptyset$, VN_m can offload the data towards the MBS, which contains all the services requested by VNs.

B. Deep Learning Based Solutions

The proposed collaborative learning-based methods use Q-learning with Q-table as a baseline solution method. Such techniques are widely used for solving RL problems with simple settings, however, they are not targeted with problems with the curse of high dimensionality, i.e., larger state or action spaces. In such scenarios, more advanced techniques are suitable. For the case of vehicular networks, this is often the case. Therefore a deep learning-based approach is also discussed, where a Deep Q Network (DQN) is used for approximating the Q-function, i.e., the action value function. In this case, the DQN can replace the Q-learning process (lines

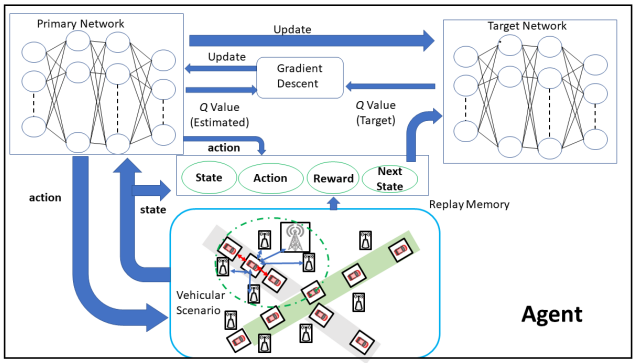

Fig. 2. DQN Architecture.

5-14 in Algorithm 1) for both V2I-AC and V2V-AC methods when estimating the Q-values.

As shown in Fig. 2, the considered DQN is based on the presence of both primary and target networks both with L_d layers with n_l ($l \in L_d$) neurons for estimating the Q-values. Considering an approach similar to [36], the primary network is used for estimating the real/primary Q-value while the target Q-values are estimated through the target network. The learning agent utilizes the backpropagation and gradient descent processes with Mean Square Error (MSE) based loss function for reducing the gap between the primary and the target Q-values. For the scenario ν, the loss function is defined as,

$$L(w,\nu) = R_{St_\nu^t, ak_\nu^t} + \Delta \max_A \left\{ Q(St_\nu^{t+1}, ak_\nu^{t+1}; w_\nu^{'}) \right\} - Q(St_\nu^t, ak_\nu^t; w_\nu) \quad (21)$$

The primary Q-value, given as $Q(St_\nu^t, ak_\nu^t; w_\nu)$, is a result of primary network with parameters w_ν, while the target Q-value $R_{St_\nu^t, ak_\nu^t} + \Delta \max_A \left\{ Q(St_\nu^{t+1}, ak_\nu^{t+1}; w_\nu^{'}) \right\}$ is based upon the results of the target network with parameters $w_\nu^{'}$. After collecting the Q-values, the EGA is used for selecting the future action. The agent's learning experiences are stored in the forms of tuple $(St_\nu^t, ak_\nu^t, R_{St_\nu^t, ak_\nu^t}, St_\nu^{t+1})$ constituted by current state, action, reward received and the next state in the replay memory of size r_b and used during the training process. Randomly sampled experiences forming a batches of size b helps to improve the training performance. The V2I-AC and V2V-AC techniques built by using the DQN-based approaches are named V2I-DAC and V2V-DAC in the following parts.

C. Limited Search-space based Heuristic Approach

With the involvement of a huge number of VTs, the problem in (10) is highly complex to be solve through the traditional heuristic approaches mainly due to large solution space $\mathcal{SP}$. For comparison purposes, a two-step limited search space-based heuristic is also considered, able to reduce the computation complexity through design parameters bounding the size of the overall search space. Algorithm 3, provides the step-by-step process to be implemented. In the first step (lines 1-13), all VTs are assigned to the available edge nodes based on the requested services and the distance measures. The ratio between the sojourn time distance and the distance between VT and edge node, i.e., $\frac{D_{m,n}(\tau_i)}{d_{m,n}(\tau_i)}$, allows selecting the edge node with the smallest communication cost and highest sojourn time (line 11). Thus, each VT greedily selects the edge node best suited for their operations without considering the presence of other competing users.

Next, a solution space $\mathcal{SP}_n\left(A^{''}(n), \Lambda_{hu}\right)$ is formed at the nth RSU node based upon the set of VTs requesting the services $(A^{''}(n))$ and step parameter Λ_{hu}. Each solution point contains a vector of offloading parameters given as,

$$\hat{\mathcal{A}}(n)_{(1\times A^{''}(n))} = \left\{ \alpha_{\rho_1}(\tau_i), \cdots, \alpha_{\rho_{A^{''}(n)}}(\tau_i) \right\}$$

The step value taken by the offloading parameter (i.e., Λ_{hu}) is considered as a design parameter limiting the size of search space. Thus, the overall solution space of the nth RSU node is based upon the total number of VTs requesting the services and the design parameter Λ_{hu}. Next for each solution point, the objective function in (10) is analyzed along with the constraint set for finding the best possible solution (lines 17-25).

Algorithm 3 Limited Search-space based Heuristic Approach (LS-HA)

Input: Set of VNs $\mathcal{V}$, R, $\{\mathcal{R}_m\}$, Λ_{hu}
Output: $\{\mathcal{A}, \mathbf{A}\}$
1: **function** FIND($A^{'}, A^{''}$)
2: **for all** $RSU_n = 1, \ldots, R$ **do**
3: **for all** $VM_m \in \mathcal{V}$ **do**
4: $A^{'}(m,n) = 1 \iff RSU_n \in \mathcal{R}_m$
5: **end for**
6: $A^{''}(n) = \sum_{m=1}^{M} (A^{'}(m,n))$
7: **end for**
8: **end function**
9: **function** ASSIGN($\mathbf{A} = \{a(m,n)\}$)
10: **for all** $VM_m \in \mathcal{V}$ **do**
11: $a_{m,n}(\tau_i) = 1 \iff \frac{D_{m,n}(\tau_i)}{d_{m,n}(\tau_i)} = \max_{n' \in \mathcal{R}_m} \left\{ \frac{D_{m,n'}(\tau_i)}{d_{m,n'}(\tau_i)} \right\}$
12: **end for**
13: **end function**
14: **function** OFFLOAD($\mathcal{A} = \{\alpha_{\rho_m}\}^{A^{''}(n)}, \forall n$)
15: **for all** $RSU_n = 1, \ldots, R$ **do**
16: Create $\mathcal{SP}_n(A^{''}(n), \Lambda_{hu}) = \{\hat{\mathcal{A}}(n)_{(1\times A^{''}(n))}\}$ with all possible solution points to be searched in the reduced-size solution space.
17: $C_h = \infty$
18: **for all** $\hat{\mathcal{A}}(n) \in \mathcal{SP}$ **do**
19: Use (10) to evaluate cost $C(\hat{\mathcal{A}}(n), \mathbf{A})$.
20: Determine all constraint functions values.
21: **if** ($C(\hat{\mathcal{A}}(n), \mathbf{A}) \leq C_h$ and all constraints are satisfied) **then**
22: $C_h = C(\hat{\mathcal{A}}(n), \mathbf{A})$ and $\{\alpha_{\rho_m}\}^{A^{''}(n)} = \hat{\mathcal{A}}(n)$
23: **end if**
24: **end for**
25: **end for**
26: **end function**
27: **return** $\{\mathcal{A}, \mathbf{A}\}$

IV. Numerical Results

The performance evaluation of the proposed solutions has been carried out through a Python-based simulator, using ML-related libraries such as NumPy, Pandas, Matplotlib. The scenario contains a multi-service vehicular network composed of one MBS, and several randomly distributed RSUs and VNs. The main parameters used in computer simulations are in Table II. We have considered a maximum of 350 RSUs, distributed in the coverage area, and a variable number of VNs from 100 to 1800. Each RSU provides a random number of services $|\mathcal{S}|$ between 1 and 6. We consider that

TABLE II
SIMULATION PARAMETERS

BS Coverage (d_n^M)	500 m
RSU Coverage (d_n)	10 m-40 m
Task Size (D_{ρ_m})	3 MB
Task Results ($D_{\rho_m}^r$)	($D_{\rho_m}/5$) MB
Required Task Latency (T_ρ)	2 s
VN Computation Cap. (η_m)	14 GFLOPS
RSU Computation Cap. (H_n)	25 GFLOPS
Task Proc. Requirements Ω_{ρ_m}	1250 FLOPS per bit
RSU Bandwidth (B_n)	35 MHz
VN Speed ($\vec{v}_m(\tau_i)$)	6 m s^{-1}-18 m s^{-1}
No. of Services ($\mathcal{S}$)	6
VN power (Pt_m, Pr_m)	(1.6 W, 1.4 W)

VNs are not always active, while, at each time instant, they request a random service with a probability equal to 0.1, while with probability 0.9 they are inactive. The two weighting coefficients (γ_1, γ_2) in (10) have been set to 0.5.

In the Q-learning simulation, we have generated different scenarios, based on the available number of RSUs and the nearby competing VNs during offloading cases. In the training phase of each Q-agent, we have used 500 randomly distributed VNs in the MBS coverage area, considered our operational area. The maximum number of RSUs that can be exploited by each VN is $\bar{N} = 4$, while the maximum number of nearby VNs that can be exploited for the V2V approach is $\bar{M} = 15$. In the reward function, $\Upsilon_1 = 1.5$, $\Upsilon_2 = 0.8$, and $\Upsilon_3 = 0.8$ are used. The weighting coefficients determine the influence of the constraint failure penalty terms compared with the other parts (i.e., joint latency and energy costs) in the reward signal. Their values are set for generating a uniform reward signal without any bias toward cost or penalty terms. Failure of service time latency constraint can be catastrophic, resulting in service failure. However, the failure of the other two constraints (i.e., sojourn time and energy constraints) can lead to additional costs without ensuring the service failure. Given that different constraints can have different influences, the values are set accordingly. For example, given the importance of service time constraint, Υ_1 has a higher value than the other two coefficients (i.e., Υ_2 and Υ_3). These values are set empirically and can be optimized for having more precise results in the future. The learning rate (γ_{lr}), discount factor (Δ), and the epsilon value (e) for the epsilon greedy algorithm part of the Q learning simulation are 0.99, 0.7, and 0.5, respectively, while Λ is equal to 0.01 and $\mathcal{I} = 10^4$ is used. The primary and target neural networks have $L_d = 5$ fully connected layers with ReLU and linear activation functions (i.e., discrete actions). Other learning parameters include batch size $b = 32$ samples, and, a replay buffer of $r_b = 50000$ samples. For the heuristic approach $\lambda_{hu} = 0.2$ is considered.

In the following, the performance of the proposed approaches has been compared with a complete offloading procedure and with a local processing approach. Since utility minimization/maximization-based node selection strategies are often considered for computation offloading purposes [24], [30], in the following two benchmark methods are also considered, based on the communication distance between edge nodes and vehicles, and the available sojourn time. In addition a Single Agent approach has been considered for comparison.

- **Minimum Distance Based Offloading (MDBO)** In this method, each VN offloads its task to the nearest RSU [24], hence:

$$a_{m,n}(\tau_i) = 1 \Longleftrightarrow n = \underset{n' \in \mathcal{R}_m}{\operatorname{argmin}} \{d_{m,n'}(\tau_i)\}, \quad \forall m$$

  Though MDBO reduces the communication latency of the offloading process, it cannot guarantee optimal performance in terms of overall latency and energy consumption.
- **Maximum Sojourn Time Based Offloading (MSTBO)** We have considered a utility maximization-based benchmark with the sojourn time as the utility function while selecting the edge node [30]. In this method, VN selects the RSU node with maximum sojourn time for offloading it's data, hence:

$$a_{m,n}(\tau_i) = 1 \Longleftrightarrow n = \underset{n' \in \mathcal{R}_m}{\operatorname{argmax}} \{T_{m,n'}^{soj}(\tau_i)\} \quad \forall m$$

  In this method, VNs can reduce the overall handover requirements; however, RSU selection process does not consider the nearby VNs and corresponding RSU resource demands, and, as a result, optimal performance cannot be guaranteed.
- **Single Agent Based Q Learning Approaches (Q-SA and Q-DSA)** To compare the performance of the proposed Q-learning based methods, we have also considered a single agent-based Q-learning approach. Both traditional tabular Q-learning (Q-SA) and DQN-learning (Q-DSA) based methods are considered. In this case, a single RL agent without considering the surrounding environment parameters attempts to find the optimal policy for the network selection and computation offloading operations jointly. Since the agent is unaware of the total number of edge nodes able to provide the requested service, it uses the static network selection policy by selecting the nearest edge node. Also, the offloading process is performed without knowing the surrounding competing VNs.

Table III shows the computation complexity of the considered Q-learning based solutions. By following the analysis considered in [37], where the computational complexity for some basic Q-learning based algorithms is given, we extended the analysis to the proposed methods, where the computational complexity of proposed V2V and V2I-based approaches can be based on the total number of vehicular scenarios considered. For the V2I-AC approach, the complexity evaluation include the minimum number of edge node scenarios considered, i.e., $\bar{N}$, in addition to the state space $\mathcal{ST}$, action space $\mathcal{AS}$ and the maximum number of iterations during each learning round $\mathcal{I}$. In addition to this, in the V2V-AC approach, the complexity also depends upon the nearby VN scenarios, i.e., $\bar{M}$, and thus requires a higher number of computations compared with the V2I-AC approach. For the DQN-based approaches, the computation complexity is function of the number of layers (L_d), the number of neurons n_l, with $l \in L_d$, computation requirements for parameter updates, batch size, number of training episodes $\mathcal{I}$, total number of steps considered while updating the target DQN model weights $\bar{\mathcal{I}}$ [38], [39]. For

TABLE III
COMPLEXITY ANALYSIS

Solution Approach	Computational Complexity
Q-SA	$\mathcal{O}(\mathcal{ST} \cdot \mathcal{AS} \cdot \mathcal{I})$ [37]
V2I-AC	$\mathcal{O}(N \cdot \mathcal{ST} \cdot \mathcal{AS} \cdot \mathcal{I})$
V2V-AC	$\mathcal{O}(\bar{N} \cdot \bar{M} \cdot \mathcal{ST} \cdot \mathcal{AS} \cdot \mathcal{I})$
Q-DSA	$\mathcal{O}(\mathcal{I} \cdot b \cdot (n_0 n_l + \sum_{l=1}^{L_d-1} n_l n_{l+1})/\bar{\mathcal{I}})$
V2I-DAC	$\mathcal{O}(\bar{N} \cdot \mathcal{I} \cdot b \cdot (n_0 n_l + \sum_{l=1}^{L_d-1} n_l n_{l+1})/\bar{\mathcal{I}})$
V2V-DAC	$\mathcal{O}(\bar{N} \cdot \bar{M} \cdot \mathcal{I} \cdot b \cdot (n_0 n_l + \sum_{l=1}^{L_d-1} n_l n_{l+1})/\bar{\mathcal{I}})$

the Q-DSA, V2I-DAC and V2V-DAC approaches, the computation complexity expressions are given in Table III, where $n_0 = |\mathcal{ST}|$ represents the state space dimension.

1) Joint Latency and Energy Cost for task processing: In Fig. 3, we present the average cost in terms of latency and energy requirements for VNs task processing. It is possible to notice that the proposed Q-learning-based approaches have reduced costs compared with the other benchmark methods including LS-HU. Both MDBO and MSTBO approaches perform the computation offloading operation without considering the surrounding environment parameters and the resource limitations of the edge servers. By selecting the nearest RSU, MDBO can reduce communication latency and energy costs. Though the MDBO approach can keep overall cost under control mainly due to the selection of the nearest edge node (i.e., reduced communication cost), as shown in Figs. 4 and 5, it suffers from a large number of failures due to the improper node selection and offloading. Additionally, with a growing number of VNs, MDBO performance weakens compared to the Q-learning based approaches. On the other hand, though the MSTBO approach selects the edge node with the highest sojourn time, the overall cost is higher mainly due to the reduced flexibility of the overall partial offloading process. Single agent-based Q-learning approaches (i.e., Q-SA and Q-DSA) without a proper knowledge of the VNs local environment also have a limited performance with growing VN density.

With better knowledge of the surrounding environment, Q-learning based approaches can jointly select the proper edge node and the amount to be offloaded. Though the V2I-AC approach is able to adapt the Q-learning policies (and perform better compared with benchmark methods) according to the varying number of edge nodes, without having the proper knowledge of the surrounding competing VTs, its performance is slightly worse compared with the V2V-AC approach. By considering both the number of edge nodes and the surrounding competing VNs information, the V2V-AC approach is able to perform the computation offloading operation with superior performance. In particular, for the case of 1400 VNs, a 30.3% performance gain in terms of reduced cost can be observed for the V2V-AC approach compared with the MSTBO method. The DQN approaches, especially V2V-DAC, allows to achieve a performance gain and is able to handle more complex scenarios. Therefore, in the considered multi-service based vehicular scenario, the proposed schemes can serve VNs with innovative services having better latency and energy performance; it is in particular important to stress that the collaborative approach by itself is able to increase

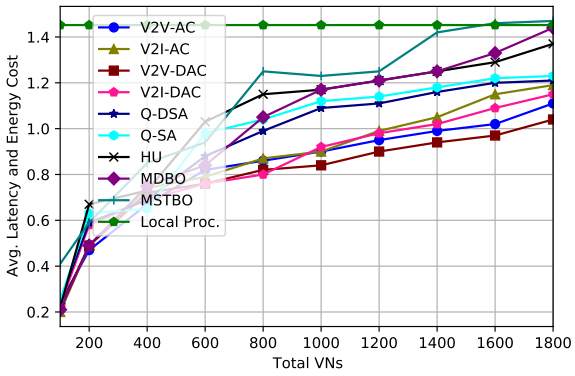

Fig. 3. Average Joint Latency and Energy Cost for variable number of VNs.

the performance, even when implemented with a tabular Q-learning approach.

2) RSU handover required during computation offloading: In a VEC environment, each VN should perform the offloading process before it crosses through the coverage range of the RSU node. The offloading process is composed by the computation data offloading towards the RSU server and the reception back of the results. In case a VN is not able to finish the offloading process before going out the coverage of the selected RSU an additional cost in terms of handover latency should be considered.

In Fig. 4, the amount of handover requests is considered, mapping the amount of times each VN is not able to complete an offloading request by the sojourn time. It is possible to notice that they increase with the increased number of VNs in the service area, and that the performance with respect to the benchmark is better. Though both MDBO and MSTBO approaches select the edge nodes to maximize the particular utility (performance in terms of communication latency/available sojourn time), they fail to adapt according to the varying VNs demands. Without properly distributing the VNs requests and improper offloading amounts they fail to adapt the sojourn time bounds of RSU nodes resulting in higher failures. The LS-HU approach with a more flexible node selection and the offloading process can have a better performance compared to the other benchmark approaches. On the other hand, by properly utilizing the parameters of the surrounding environment through V2I and V2V, proposed Q learning-based approaches are able to reduce the overall number of failures. It is also possible to see that the Q-SA approach has a higher number of failures compared with both V2I-AC and V2V-AC approaches highlighting the importance of the proposed vehicular communication-based learning framework. Moreover, it is clear the advantage of the V2V approaches with respect to the V2I information sharing, while the DQN solutions allows to slightly increase the performance with respect to the tabular Q-learning approaches.

3) Number of VNs failing to satisfy the service time constraint: Each VN should complete the task processing operation within a requested service latency bound failure which can reduce the QoS. In Fig. 5, we present the average

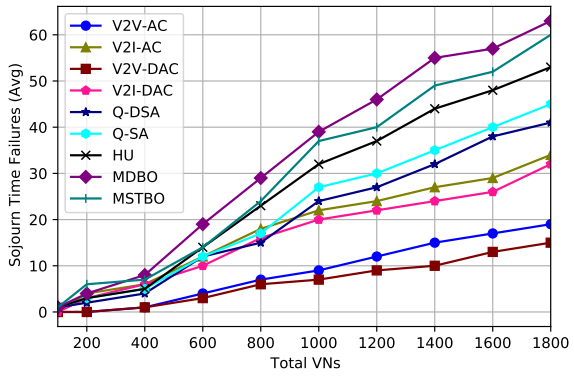

Fig. 4. Average Number of RSUs Handover Requests for variable number of VNs.

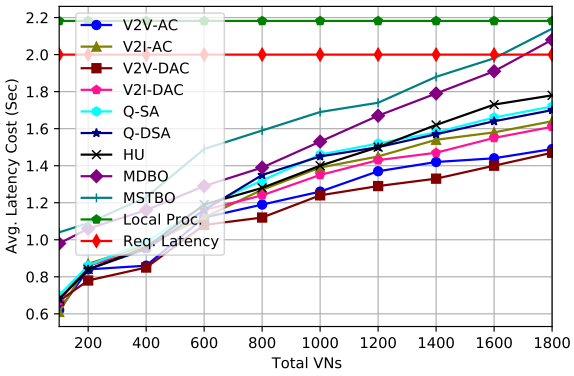

Fig. 6. Average Task Completion Latency for variable number of VNs.

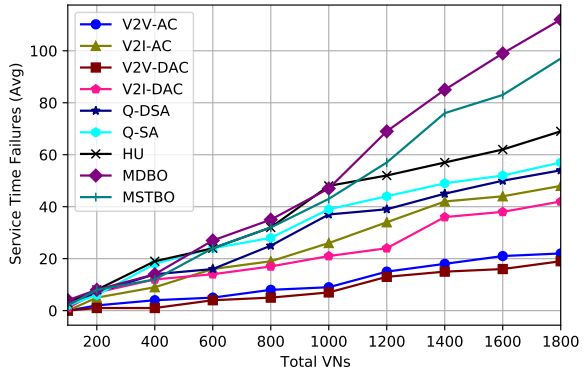

Fig. 5. Average number of Service Time Failures for variable number of VNs.

number of VNs failing to satisfy the service latency bound. Both MDBO and MSTBO approaches are failing to perform the task processing within the service time requirement. This shows that performing computation offloading operation without considering the surrounding environment results in higher failures. On the other hand, the proposed Q-learning-based approaches, especially the V2V-AC method, exploiting various environmental parameters, are able to select the proper edge node and amount to be offloaded jointly, resulting in superior performance compared with the other benchmark schemes. The DQN approaches allows to slightly increase the performance with respect to the tabular Q-learning approaches. Additionally, the proposed VNs scenario-based Q-learning approaches outperform the traditional single agent based approaches i.e., Q-SA and Q-DSA, in terms of a number of failures. Thus, the proposed Q-learning and DQN approaches can be useful for enabling latency-critical services over VNs.

4) Task Completion Latency: Total task completion latency is a function of computation offloading and the local device computation times. Here, we compare the average task completion latency of different schemes when changing the vehicles density in the service area. Fig. Fig:LC provides the average task computation latency required by each solution method. In a given service area, the total latency required for processing the tasks with Q-learning approaches is much smaller than the benchmarks. Both Q-learning and DQN approaches are able to select proper edge nodes and the amount of data to be offloaded towards them that results in a better performance in terms of overall latency requirements. By analyzing the vehicular environment data through the V2V and V2I technologies, the Q-learning and DQN approaches are able to reduce the overall task processing latency. The traditional single agent-based approaches, i.e., Q-SA and Q-DSA, require higher latency costs, mainly due to the improper node selections and offloading process. On the other hand, the latency performance of the benchmark methods increases rapidly, which can be disastrous for latency-critical services. Though the MDBO approach is using the nearest edge node for offloading its data, without properly assessing the resource availability, the number of edge nodes, and the competing VNs, it fails to adapt to the increasing VNs demands. Similarly, the MSTBO approach also fails to adapt its network selection and computation offloading policies according to the vehicular users' demands resulting in higher latency costs. The LS-HU approach can reduce the latency cost by selecting the nodes with proper distance measures and offloading adequate data compare to the other benchmark methods. This result shows the high potential of proposed schemes for enabling latency-critical applications and services.

5) Average Energy Consumption: Fig. 7 shows the average amount of energy consumed by VNs for the complete task processing. The total energy consumed includes the energy required for the local device computation, transmission, and reception of tasks towards and from RSUs. It has to be noticed that the total energy required for locally processing the vehicular tasks is relatively higher than the offloading process energy. Therefore, with a complete offloading process, the benchmark methods have relatively better energy performance compared with the proposed Q-learning and DQN approaches. On the other hand, Q-learning methods require higher energy mainly because of the local processing energy part, where the DQN approaches consume slightly more than the tabular Q-learning approach. However, the overall performance in terms of joint latency and energy requirements (as shown in Fig.

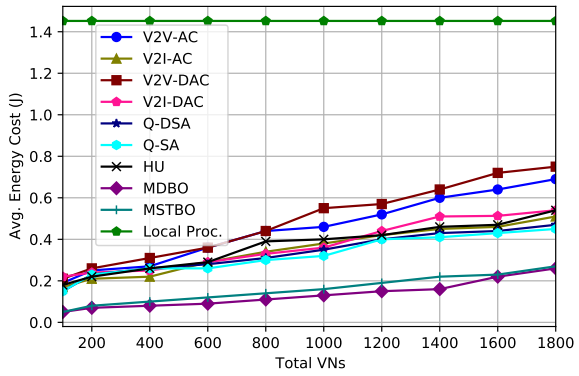

Fig. 7. Average Energy Consumption for Task Completion for variable number of VNs.

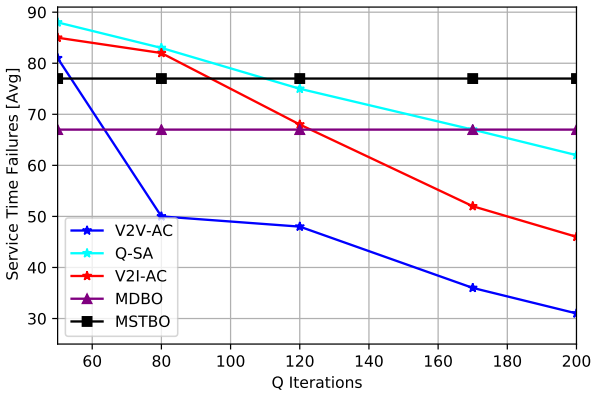

Fig. 8. Q-Learning Training Performance in Terms of a Service Time Failures for variable number of iterations.

TABLE IV
AVERAGE PERCENTAGE OF DATA OFFLOADING.

# VNs	100	200	400	600	800	1000	1200	1400	1600	1800
V2I-AC	.79	.80	.82	.64	.65	.57	.58	.59	.60	.60
V2V-AC	.80	.78	.77	.60	.59	.50	.48	.47	.45	.45
V2I-DAC	.79	.80	.80	.62	.66	.58	.59	.57	.56	.56
V2V-DAC	.82	.75	.75	.58	.54	.50	.49	.45	.43	.42

3), considering also the handover and latency failures is much better than the benchmarks.

6) Average Computation Offloading: We analyze then the impact of nearby VNs on the computation offloading amount. In Table IV, the average percentage of data offloaded by VNs towards RSUs is shown. Since the available resources at each RSU are limited, if the number of vehicles increases, the V2V assisted collaborative Q-learning approach offloads the lowest amount of data towards RSU servers for avoiding handovers. On the other hand, the V2I approach does not take into account the other VNs, and, as a result, a higher offloading percentage and more handover requests are set.

7) Training Iterations in Terms of Service Failures: For analyzing the training performance of the proposed Q learning-based solutions in Fig. 8, we show the number of VNs failing to perform the task processing in a limited time. For a given set of VNs with reduced training iterations, Q-learning approaches have a higher number of failures, however, with increasing training iterations Q-learning approaches, especially the V2V-AC approach outperforms the other solutions. It has to be clarified that DQN-based training iterations can last for longer time compared to the tabular approach, despite a reduced number of overall iterations is generally needed. Hence, a comparison with the tabular method is not fair, preventing their representation in figure.

V. CONCLUSION

In this paper, we have considered a joint RSU selection and computation offloading problem in a multi-service multi-user vehicular network. We have modeled it as a RL-based problem and solved it by using Q-learning methods. Two collaborative learning-based approaches where the Q-agents learn the optimal policy exploiting the V2I and V2V communication paradigms are considered. Along with the traditional tabular method, a DQN approach is also considered for estimating the Q values, allowing to handle more complex learning scenarios. Compared with other benchmark methods, the proposed schemes provide better network-wide performance in terms of latency and consumed energy. Thus, proposed schemes can have a great potential for enabling latency-critical applications and services over VNs. This work highlights the importance of enabling collaborative RL strategies, exploiting vehicular communication modes, for solving the joint network selection and the offloading problem over a multi-service vehicular network; this is also strengthen by the fact that the collaboration among vehicular nodes has an impact on the performance higher than the implementation of DQN with respect to the tabular Q-learning.

As a future direction, the offloading process costs can further be reduced by allowing VNs to select more than one edge node for computation offloading. However, such an approach can add extra dimensions of complexity, and more rigorous solution methods. It can also be interesting to use a multi-task learning-based approach (i.e. multi task RL/FL) with additional vehicular service parameters. By considering the distributed nature of the VNs and corresponding data, other decentralized learning methods such as FL can also be a potential solution method.

REFERENCES

[1] B. Ji, X. Zhang, S. Mumtaz, C. Han, C. Li, H. Wen, and D. Wang, "Survey on the Internet of Vehicles: Network architectures and applications," *IEEE Commun. Std. Mag.*, vol. 4, no. 1, pp. 34–41, Mar. 2020.

[2] M. N. Ahangar, Q. Z. Ahmed, F. A. Khan, and M. Hafeez, "A survey of autonomous vehicles: Enabling communication technologies and challenges," *Sensors*, vol. 21, no. 3, pp. 1–33, 2021, Art. No.706.

[3] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5G," ETSI, White Paper 11, Sep. 2015. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf

[5] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile Networks and Applications*, vol. 26, no. 3, pp. 1145–1168, 2021.

[6] A. B. De Souza, P. A. Rego, T. Carneiro, J. D. C. Rodrigues, P. P. Rebouças Filho, J. N. De Souza, V. Chamola, V. H. C. De Albuquerque, and B. Sikdar, "Computation offloading for vehicular environments: A survey," *IEEE Access*, vol. 8, pp. 198 214–198 243, 2020.

[7] Z. Zhou, P. Liu, Z. Chang, C. Xu, and Y. Zhang, "Energy-efficient workload offloading and power control in vehicular edge computing," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Barcelona, Spain, Apr. 2018, pp. 191–196.

[8] L. Liang, H. Ye, and G. Y. Li, "Toward intelligent vehicular networks: A machine learning framework," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 124–135, Feb. 2019.

[9] Z. Ning, P. Dong, X. Wang, J. J. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 6, pp. 1–24, Dec. 2019, art. no. 60.

[10] H. Zhou, W. Xu, J. Chen, and W. Wang, "Evolutionary V2X technologies toward the internet of vehicles: Challenges and opportunities," *Proc. IEEE*, vol. 108, no. 2, pp. 308–323, Feb. 2020.

[11] C. Chen, L. Chen, L. Liu, S. He, X. Yuan, D. Lan, and Z. Chen, "Delay-optimized v2v-based computation offloading in urban vehicular edge computing and networks," *IEEE Access*, vol. 8, pp. 18 863–18 873, 2020.

[12] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Collaborative data scheduling for vehicular edge computing via deep reinforcement learning," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9637–9650, 2020.

[13] J. Cui, F. Ouyang, Z. Ying, L. Wei, and H. Zhong, "Secure and efficient data sharing among vehicles based on consortium blockchain," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8857–8867, 2022.

[14] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the Internet of Vehicles," *Proc. IEEE*, vol. 108, no. 2, pp. 246–261, Feb. 2020.

[15] L. Tang, B. Tang, L. Zhang, F. Guo, and H. He, "Joint optimization of network selection and task offloading for vehicular edge computing," *Journal of Cloud Computing*, vol. 10, pp. 1–13, 2021, art. no. 23.

[16] C. Yang, Y. Liu, X. Chen, W. Zhong, and S. Xie, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, vol. 7, pp. 26 652–26 664, 2019.

[17] X. Hou, Z. Ren, J. Wang, W. Cheng, Y. Ren, K.-C. Chen, and H. Zhang, "Reliable computation offloading for edge-computing-enabled software-defined iov," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7097–7111, 2020.

[18] A. Bozorgchenani, S. Maghsudi, D. Tarchi, and E. Hossain, "Computation offloading in heterogeneous vehicular edge networks: On-line and off-policy bandit solutions," *IEEE Trans. Mobile Comput.*, 2021, early access, doi: 10.1109/TMC.2021.3082927.

[19] S. S. Shinde, A. Bozorgchenani, D. Tarchi, and Q. Ni, "On the design of federated learning in latency and energy constrained computation offloading operations in vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, 2021, early access, doi: 10.1109/TVT.2021.3135332.

[20] Y. Ren, Y. Sun, and M. Peng, "Deep reinforcement learning based computation offloading in fog enabled industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4978–4987, 2021.

[21] X. Zhu, Y. Luo, A. Liu, M. Z. A. Bhuiyan, and S. Zhang, "Multiagent deep reinforcement learning for vehicular computation offloading in IoT," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9763–9773, 2021.

[22] D. Chen, Y.-C. Liu, B. Kim, J. Xie, C. S. Hong, and Z. Han, "Edge computing resources reservation in vehicular networks: A meta-learning approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5634–5646, 2020.

[23] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3061–3074, 2019.

[24] X. Zhang, J. Zhang, Z. Liu, Q. Cui, X. Tao, and S. Wang, "MDP-based task offloading for vehicular edge computing under certain and uncertain transition probabilities," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3296–3309, 2020.

[25] Z. Zhou, J. Feng, Z. Chang, and X. Shen, "Energy-efficient edge computing service provisioning for vehicular networks: A consensus ADMM approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5087–5099, 2019.

[26] L. Cesarano, A. Croce, L. D. C. Martins, D. Tarchi, and A. A. Juan, "A real-time energy-saving mechanism in internet of vehicles systems," *IEEE Access*, vol. 9, pp. 157 842–157 858, 2021.

[27] J. Wang, C. Jiang, Z. Han, Y. Ren, and L. Hanzo, "Internet of vehicles: Sensing-aided transportation information collection and diffusion," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 3813–3825, 2018.

[28] J. Wang, C. Jiang, K. Zhang, T. Q. S. Quek, Y. Ren, and L. Hanzo, "Vehicular sensing networks in a smart city: Principles, technologies and applications," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 122–132, 2018.

[29] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "Mobile edge computing partial offloading techniques for mobile urban scenarios," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, United Arab Emirates, Dec. 2018.

[30] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377–4387, 2018.

[31] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11 158–11 168, Nov. 2019.

[32] X. Mo and J. Xu, "Energy-efficient federated edge learning with joint communication and computation design," *Journal of Communications and Information Networks*, vol. 6, no. 2, pp. 110–124, Jun. 2021.

[33] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls *et al.*, "Value-decomposition networks for cooperative multi-agent learning," *arXiv:1706.05296*, 2017, doi:10.48550/arXiv.1706.05296.

[34] T. Rashid, M. Samvelyan, C. Schroeder de Witt, G. Farquhar, J. N. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 21, pp. 1–51, 2020, art. no. 178.

[35] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2nd ed. Cambridge, MA, USA: MIT press, 2018.

[36] H. Guo, J. Liu, J. Ren, and Y. Zhang, "Intelligent task offloading in vehicular edge computing networks," *IEEE Wireless Communications*, vol. 27, no. 4, pp. 126–132, 2020.

[37] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan, "Is q-learning provably efficient?" *Advances in neural information processing systems*, vol. 31, 2018.

[38] H. Yang, Z. Xiong, J. Zhao, D. Niyato, L. Xiao, and Q. Wu, "Deep reinforcement learning-based intelligent reflecting surface for secure wireless communications," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 375–388, 2021.

[39] Z. Ning, P. Dong, X. Wang, L. Guo, J. J. P. C. Rodrigues, X. Kong, J. Huang, and R. Y. K. Kwok, "Deep reinforcement learning for intelligent internet of vehicles: An energy-efficient computational offloading scheme," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1060–1072, 2019.

Swapnil Sadashiv Shinde (Student Member, IEEE) is a Ph.D. student at the University of Bologna, Italy. He received the the MS degree in Telecommunication Engineering from the University of Bologna, Italy, in 2020. From 2015 to 2017, he worked as a Project Engineer in the Indian Institute of Technology, Kanpur, India. His main focus is on the Connected Vehicles for Beyond 5G Scenarios.

Daniele Tarchi (Senior Member, IEEE) is an Associate Professor at the University of Bologna, Italy. He holds a Ph.D. degree in Informatics and Telecommunications Engineering from the University of Florence, Florence, Italy, in 2004. He is the author of more than 130 published articles in international journals and conference proceedings. His research interests are mainly on Wireless Communications and Networks, Satellite Communications and Networks, Edge Computing, Fog Computing, Smart Cities, and Optimization Techniques. Prof. Tarchi is an IEEE Senior Member since 2012.