Multi-Time-Scale Markov Decision Process for Joint Service Placement, Network Selection, and Computation Offloading in Aerial

Multi-Time-Scale Markov Decision Process for Joint Service Placement, Network Selection, and Computation Offloading in Aerial IoV Scenarios

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

Abstract—Vehicular Edge Computing (VEC) is considered a major enabler for multi-service vehicular 6G scenarios. However, limited computation, communication, and storage resources of terrestrial edge servers are becoming a bottleneck and hindering the performance of VEC-enabled Vehicular Networks (VNs). Aerial platforms are considered a viable solution allowing for extended coverage and expanding available resources. However, in such a dynamic scenario, it is important to perform a proper service placement based on the users' demands. Furthermore, with limited computing and communication resources, proper user-server assignments and offloading strategies need to be adopted. Considering their different time scales, a multi-time-scale optimization process is proposed here to address the joint service placement, network selection, and computation offloading problem effectively. With this scope in mind, we propose a multi-time-scale Markov Decision Process (MDP) based Reinforcement Learning (RL) to solve this problem and improve the latency and energy performance of VEC-enabled VNs. Given the complex nature of the joint optimization process, an advanced deep Q-learning method is considered. Comparison with various benchmark methods shows an overall improvement in latency and energy performance in different VN scenarios.

Index Terms—Vehicular Edge Computing, Aerial Networks, Service Placement, Network Selection, Computation Offloading, Markov Decision Processes, Multi-time Scale Optimization

1 Introduction

With the evolution of technologies such as the Internet of Things (IoT), Multi-access Edge Computing (MEC), Network Softwarization, and different communication modes, modern Vehicular Networks (VNs) are turning towards a smarter, highly reliable, and secure networking system enabled through software programmability, providing advanced intelligent services and applications to end users. These advanced services and applications often bring a tremendous amount of data to be processed and strict latency and reliability constraints [1]. With limited processing capabilities, Vehicular Terminals (VTs) are unable to process the tasks in a limited time, and often utilize Edge Computing (EC) services enabled through the nearby access points. Despite the presence of Road Side Units (RSUs), equipped with EC facilities, their limited coverage ranges, computation, communication, and storage resources require the introduction of advanced solutions. On one side the vehicular scenario is inherently dynamic, hence, dynamic service placement over the Edge Nodes (ENs) depending upon the user requests is needed. Furthermore, the limitations of computation and communication resources put restrictions on the number of VTs that each EN will serve. Therefore, proper edge-based service placement solutions along with the optimal EN selection for offloading the computational load of VTs are needed to have the benefits of Vehicular Edge Computing (VEC) systems.

Recently, various new aerial networking platforms, such as Unmanned Aerial Vehicles (UAVs), air taxis, balloons, have populated the sky. Compared to different satellite constellations, these platforms are located at reduced distances from ground users and can be further exploited for latency-critical scenarios such as VNs. With their onboard computation, communication, and storage resources, these platforms can be extremely useful to increase the capacity of traditional VEC systems that involve only terrestrial nodes. Recently, such multi-layered Edge Computing (EC) enabled distributed networking scenarios have shown promising results, especially in the case of latency-critical VNs [2], [3], [4]. However, similar to terrestrial VEC nodes, aerial platforms, with their size limitations, can hold limited resources, and proper service placement, network selection, and offloading operations are needed to boost overall performance.

In joint Terrestrial and Non-terrestrial (T/NT) VEC networks, composed of multiple layers, each VT has the option to select EC nodes from various layers. Additionally, each node on these platforms, with its storage resources, can hold a set of services. With this in mind, in such multi-user vehicular scenarios, with multiple T/NT layers, service placement, network selection, and computation offloading problems can become highly complex. Traditional optimization techniques along with heuristic and meta-heuristic approaches can only have a limited impact and may not be useful. Recently, various machine learning (ML) approaches,

- *Swapnil Sadashiv Shinde was with the Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi", University of Bologna, 40136 Bologna, Italy. He is now with CNIT—University of Bologna Research Unit, 40136 Bologna, Italy (email: swapnil.shinde@cnit.it)*
- *Daniele Tarchi is with the Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi", University of Bologna, 40136 Bologna, Italy, (email: daniele.tarchi@unibo.it)*

- *This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART").*

especially Reinforcement Learning (RL), have found ways to solve such complex networking problems [1], [5].

It is interesting to note that the service placement, network selection and offloading decisions can be performed on different time scales, based on specific demands, network topologies, and user requirements. For example, the service placement operation requires a longer time interval and can only be performed/updated at longer time scales, mainly due to centralized controller operations and longer service activation time [6]. On the other hand, network selection operation, based on the dynamicity of the VTs and nearby environments, may require a moderate amount of time to establish a proper connection between VTs and nearby reliable edge servers [7]. Finally, task offloading operations will be based on VTs task requirements, and can be performed at shorter time scales, especially in the case of latency-critical applications [8]. With these issues in mind, in the past, researchers have mainly focused on either of these three problems and proposed solutions by making some assumptions about the other two [9]. On the other hand, in some cases, joint optimization of either of these problems has been considered, while assuming the same time scales [1]. However, such solutions cannot be considered optimal since this process may require decisions made at different time scales, impacting each other's performance.

With these issues in mind, in this work, we aim to solve the joint service placement, network selection, and offloading problem over the VN environment aimed at minimizing the latency and energy costs by considering proper time scales. For this, we first model the latency and energy elements involved in this process and map them on a constrained optimization problem over a dynamic VN. Next, we exploit an RL approach to solve the problem, in particular by modeling it as a Markov Decision Process (MDP). With the involvement of multiple time scales, we consider a multi-time-scale MDP approach by modeling three different MDPs impacting each other's decisions for the considered problem. This allows us to solve the problem at different time-scales effectively.

1.1 Related Work

The EC-enabled VN, or VEC, has been extensively studied in the literature to provide latency-critical and data-intensive services to end users [1]. From a network point of view, the joint T/NT networks have gained great popularity in serving VTs with increased capacity, coverage and sustainability [2], [4]. In addition, the use of ML to solve complex vehicular problems has gained notable popularity in recent years [10]. Among others, RL-based studies are widely performed to solve complex VN problems such as resource allocation, network selection, and service placement [1], [5]. Most VEC studies can be classified according to the problems under consideration. In particular, service placement/caching, network selection, and computation offloading are the main fields of VEC research. In some cases, these problems are considered jointly. From an optimization point of view, studies are performed to optimize latency, energy, reliability, or in some cases joint costs. In the following, we discuss some of the important studies related to these three problems and some joint optimization-related solutions from the recent past. Furthermore, we examine multi-time-scale optimization by specifically focusing on the most influential papers that address this aspect.

a) Service Placement Problem: In [11], authors have studied a distributed service placement problem in a moving vehicle cluster using the knowledge of vehicle mobility patterns. A flexible and distributed service model with data-dependent tasks is considered for dynamic vehicular environments. In [12], the IoV-based dynamic service placement problem is considered with the aim of maximizing the utilization of edge resources and reducing the service delay. Deep reinforcement learning-based solutions are proposed considering dynamicity, increasing service demands, and varying request types. In [13], the authors have considered a UAV-based EC facility for vehicular service placement problem formed as a multi-objective optimization problem. In [14] the authors investigate the dynamic resource allocation problem for the placement of virtual network function (VNF) in satellite edge clouds. The aim is to jointly minimize the cost of network bandwidth and the end-to-end delay of the service. Additionally, in this case, the focus is limited to service placement. In [15] a potential game approach is proposed for the placement of VNFs in satellite edge computing, where a satellite network should provide computing services for as many user requests as possible. The VNF placement problem aims to maximize the overall network payoff, and a decentralized resource allocation algorithm based on a potential game (PGRA) is proposed to tackle the VNF placement problem by finding a Nash equilibrium. However, in these studies, the authors have focused mainly on the dynamic service placement problem without taking into account the network selection and offloading process optimizations. These service placement approaches without taking into account user demands and offloading decisions cannot guarantee optimal behavior.

b) Network Selection Problem: In [8], the authors have proposed an on-line and off-policy learning algorithm based on multi-armed bandit theory for addressing the network selection problem in VEC environments. Also in [16], the authors have proposed a multi-hop mobility-aware task offloading mechanism with optimal EN selection for autonomous driving scenarios in VEC. In [17], authors have proposed a multi-agent RL-based computation offloading strategy for vehicular scenarios. The latency performance of the offloading process is optimized with the binary computation offloading strategy. In [18], the authors focus on optimizing network selection in Satellite Aerial Ground Integrated Networks and formulate a corresponding evolutionary game. A network selection algorithm based on evolutionary games is proposed to study the autonomous decision-making process of network selection as a supplement. A network selection algorithm based on the deep deterministic policy gradient is proposed to handle continuous high-dimensional action spaces. Such network selection optimization studies and offloading process decisions are limited to node selection only, without considering the optimization of partial offloading amounts. In addition, the service placement problem is also neglected by static deployments. Offloading an imperfect amount of data to the optimal selected EN can still have limited impacts.

c) Computation Offloading Problem: In [19], the authors have proposed a federated learning-based framework for optimizing the offloading parameter values by assuming the nearest node selection strategy. In [20], the authors have studied the partial offloading problem in EC environments assisted by parked vehicles. In addition, in [21], the authors have proposed deep learning-based solutions for the Vehicle-to-Vehicle assisted partial offloading problem in vehicular fog computing environments. In [22], the authors propose a hierarchical aerial computing framework consisting of High-Altitude Platforms (HAPs) and UAVs. The framework is designed to offer MEC services for various IoT applications. The objective is to maximize the amount of IoT data processed by aerial MEC platforms, while taking into account the delay requirements of the IoT devices and the resource constraints of the UAVs and HAPs. Since an exhaustive search would be computationally expensive, the problem is addressed using a matching game theory-based algorithm to determine the offloading decisions of IoT devices to UAVs and a heuristic algorithm to determine the offloading decisions between UAVs and HAPs. However, such studies, where service placement and network selection decisions are based on a static/heuristic approach, cannot guarantee optimal performance.

d) Joint Solutions: In [1], the authors have proposed collaborative RL-based strategies to solve the problem of joint network selection and computation offloading in a multi-service VEC environment. However, the service placement is considered static. In [23], the authors have considered a dependency-aware task offloading and service-caching problem in a vehicular environment without taking into account the different time scales in a decision-making process. In [7], the authors have proposed a joint strategy for network selection and offloading in VEC environments with a single service. These joint studies are often limited to the single-time scale approach and often neglect the impact of service placement on the offloading process.

e) Multi-time scale optimization: Multi-time scale optimization has a long-standing interest, however, due to the inheritance complexity of its analysis has often been neglected or simplified to single time scale approaches. Recently, the analysis and further optimization of multi-time-scale systems have been increasingly considered. Among others, in [24], the authors explore the concept of dynamic multi-time scale user admission and resource allocation in MEC systems with a focus on semantic extraction. Since semantic extraction tasks exhibit a stochastic nature, the researchers formulate a stochastic optimization problem by representing the tasks as dynamically arriving in the temporal domain. In [25], the authors focus on the problem of event-based security control in a specific type of cyber physical systems called multi-time-scale cyber physical systems, which are vulnerable to DoS attacks. To address the challenges posed by DoS attacks and the multi-time-scale nature of these systems, a security control framework is proposed. This framework aims to design the switched controller and the event-triggered mechanism simultaneously. In [26], the authors propose a deep reinforcement learning (DRL) approach that operates on multiple time scales to optimize the use of radio resources in Non-Terrestrial Networks. The approach involves collaborative decision-making between a Low Earth Orbit (LEO) satellite and user equipment (UE), each with their own control cycles. The UE updates its policy to enhance the value functions of both the satellite and itself, whereas the LEO satellite makes decisions based on a finite-step rollout using the reference decision trajectory provided by the UE. To the best of our knowledge, no paper has considered up to now the joint service placement, network selection and computation offloading problem in a multi-time scale fashion.

1.2 Motivations and Contributions

Based on the previously commented studies, it can be seen that the authors have mainly considered solving one of the three problems discussed before, neglecting their impacts on each other. However, in some cases, joint optimization is performed without taking into account multiple time scales and their impacts. This can lead to suboptimal solutions and further analysis is required to increase the performance of VNs. Therefore, this motivates us to carry out this activity in which we aim to solve the problem of service placement, network selection, and offloading simultaneously, considering their impacts on each other. As mentioned earlier, when addressing the challenge of simultaneously performing service placement, network selection, and computation offloading, it is necessary to consider operations that occur at different time scales. This makes it impractical to use traditional solving methods that focus on a specific moment, as each subproblem requires a distinct optimization approach. This complexity is further amplified by the inclusion of terrestrial and aerial platforms, which not only enhance resilience and coverage but also introduce an additional dimension. Consequently, this paper suggests a multilayer multi-time scale approach to optimize the three operations together, using an MDP model.

The main contributions of this work can be summarized in the following points:

- **Multi-time Scale Approach:** A system model is defined with a multi-time scale approach for the service placement, network selection, and computation offloading process with dynamic VTs. Furthermore, a constrained optimization problem is formed to minimize a proper cost function, including also latency and energy terms, by performing a dynamic service placement, network selection, and offloading process.
- **MDP Solutions:** There is a lack of extensive research on multi-time scale optimization. Because these problems are highly complex, a suitable optimization approach has been devised using the MDPs. The service placement, network selection, and offloading problems are modeled as a sequential decision-making process through proper multidimensional MDP models. A multi-time scale MDP process is adapted for enabling decision making at different time scales, and optimal policy is determined through the deep Q-learning approach.
- **Performance Evaluation:** The simulation results of the proposed methods are compared with a set of benchmark methods and their effectiveness is evaluated. Finally, proper conclusions are drawn on the basis of the findings.

The structure of the paper is as follows. In Section 2, we present the system model, specifically focusing on the multilevel scenario and defining the problem as a multi-time-scale approach. In Section 3, we propose a multi-time-scale optimization method, which involves considering three MDPs, while in Section 4 the Deep Q-Learning solution for joint Service Placement, Network Selection, and Computation Offloading is described. Section 5 presents the Numerical Results, which consider various scenarios where the problem can be applied. Lastly, in Section 6, we provide the conclusions drawn from the study.

2 System Model and Problem Formulation

We focus on an IoV scenario with multiple EC layers and randomly distributed VTs in the road scenario. We consider a multi-layer ($l = 0, \dots, L$ with $L = 3$) joint air-ground network, composed of HAPs ($l = 3$), UAVs (i.e., LAP nodes) ($l = 2$), RSUs ($l = 1$), deployed along the road paths, and randomly distributed VTs ($l = 0$) traveling on a road in either direction, where $\mathcal{V} = \{v_1, \dots, v_m, \dots, v_M\}$, $\mathcal{R} = \{r_1, \dots, r_n, \dots, r_N\}$, $\mathcal{U} = \{u_1, \dots, u_p, \dots, u_P\}$, correspond to the sets denoting M VTs, N RSUs, and P UAVs, respectively. Additionally, one HAP node is denoted as H_h. In the vehicular scenario considered, VTs can request services characterized by different requirements. By assuming that $\mathcal{S} = \{S_1, \dots, S_s, \dots, S_{\bar{S}}\}$ is the set of all possible services that can be provided, due to the limited available resources, the generic jth EN from lth layer can provide only a subset of services equal to $\hat{\mathcal{S}}_j^l \subset \mathcal{S}$. In Tab. 1, the main symbols used throughout the paper are list for better comprehension of the text.

The system is modeled in a discrete time manner and the network parameters are supposed to be constant throughout each time interval τ, where τ_i identifies the ith time interval, i.e., $\tau_i = \{\forall t | t \in [i\tau, (i+1)\tau]\}$. The generic mth VT is characterized by a processing capacity equal to $c_{v,m}$ Floating Point Operations per Second (FLOPS) per CPU cycle, while its CPU frequency is $f_{v,m}$. Each VT is supposed to be able to communicate on a bandwidth $B_{v,m}^{\text{rsu}}(\tau_i)$ with each RSU, with a bandwidth $B_{v,m}^{\text{LAP}}(\tau_i)$ with each UAV and with a bandwidth $B_{v,m}^{\text{HAP}}(\tau_i)$ with the HAP. Each $v_m \in \mathcal{V}$ 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 $\left\langle D_{\rho_m}, D_{\rho_m}^r, \Omega_{\rho_m}, T_{\rho_m}, S_{\rho_m} \right\rangle$ corresponding to a task of size D_{ρ_m} Byte, expected to give in output a result with size $D_{\rho_m}^r$ Byte, requesting Ω_{ρ_m} CPU execution cycles and maximum execution latency T_{ρ_m}. Here, $S_{\rho_m} \in \mathcal{S}$ corresponds to a specific service requested by VT v_m that belongs to a set of services $\mathcal{S}$ provided by the network service provider. In addition, the average request rate for the sth service is modeled through the Zipf distribution function given by $\lambda_s(\tau_i) = 1/\kappa s^\beta$ where $\kappa = \sum_s 1/s^\beta$ with $\beta \in [0, 1]$ being the popularity skew index [27].

The nth RSU, supposed to be in a fixed position with a coverage radius $R_{r,n}$, is characterized by a processing capacity equal to $c_{r,n}$ FLOPS per CPU cycle, with CPU frequency $f_{r,n}$, CPU cores $\mathcal{L}_{r,n}$, and communication capabilities, which is supposed to be identified through a communication technology able to cover the VTs on ground with an overall bandwidth $B_{r,n}$. Each RSU can provide EC services to VTs in its coverage space. As mentioned before, with its limited resources, r_n can store only a subset of services $\hat{\mathcal{S}}_{r_n}^1 \subset \mathcal{S}$. Furthermore, it is expected that the area is covered by multiple UAVs with the pth at altitude $\bar{h}_{u,p}$ and coverage radius $R_{u,p}$. The pth UAV is supposed to move at a

TABLE 1
List of symbols used in the paper.

Symbol	Description
$\mathcal{V}$, $\mathcal{R}$, $\mathcal{U}$, $\mathcal{S}$	Sets of VTs, RSUs, UAVs, and services
$\hat{\mathcal{S}}_j^l$	The set of services provided by the jth EN on the lth layer
τ_i	The ith time interval
$c_{v,m}$	mth VT processing capacity in FLOPS per CPU cycle
$f_{v,m}$	mth VT CPU frequency
$B_{v,m}^{\text{rsu}}(\tau_i)$	mth VT to RSU communication bandwidth
$B_{v,m}^{\text{LAP}}(\tau_i)$	mth VT to UAV communication bandwidth
$B_{v,m}^{\text{HAP}}(\tau_i)$	mth VT to HAP communication bandwidth
$\rho_m(\tau_i)$	mth VT computational request at τ_i
D_{ρ_m}, $D_{\rho_m}^r$	Input and Output Task size for the request ρ_m
Ω_{ρ_m}	Requested CPU execution cycles for the request ρ_m
T_{ρ_m}	Target maximum execution latency for the request ρ_m
S_{ρ_m}	The service requested by mth VT
$R_{r,n}$, $B_{r,n}$	Coverage radius and communication bandwidth for the nth RSU
$c_{r,n}$, $f_{r,n}$, $\mathcal{L}_{r,n}$	nth RSU processing capacity, CPU frequency, and number of CPU cores
$\bar{h}_{u,p}$, $R_{u,p}$, $B_{u,p}$	Altitude, coverage radius, and bandwidth of pth UAV
$c_{u,p}$, $f_{u,p}$, $\mathcal{L}_{u,p}$	pth UAV processing capacity, CPU frequency, and number of CPU cores
$\bar{h}_h$, R_h, B_h	HAP node altitude, coverage radius, and bandwidth
c_h, f_h, $\mathcal{L}_h$	HAP processing capacity, CPU frequency, and number of CPU cores
	Overall communication bandwidth given by the HAP
$\vec{v}_m(\tau_i)$	mth VT speed at τ_i
$\vec{v}_{\min}$, $\vec{v}_{\max}$	Minimum and maximum VT speed
$D_{v_m,j}(\tau_i)$	Remaining path length within which mth VT remains under the coverage of the jth node at τ_i
$T_{v_m,j}^{\text{soj}}(\tau_i)$	Remaining sojourn time for the mth VT under the coverage of the jth node at τ_i
Δ^{ns}, Δ^{sp}, Δ^{off}	Network Selection, Service Placement, and Offloading interval
$\mathcal{S}^{\text{sp}}$, $\mathcal{S}^{\text{ns}}$, $\mathcal{S}^{\text{off}}$	States for Network Selection, Service Placement, and Offloading
$\mathcal{A}^{\text{sp}}$, $\mathcal{A}^{\text{ns}}$, $\mathcal{A}^{\text{off}}$	Action Space for Network Selection, Service Placement, and Offloading
$b(S_s, j, l, \tau_i^{\text{sp}})$	Binary service placement parameter
$\mathbf{A}(M, J(l), l, \tau_i^{\text{ns}})$	decision matrix for network selection
$K_{j,l}^{\max}$, $K_{j,l}(\tau_i^{\text{ns}})$	Maximum and actual number of VTs requesting services
$T_{c,j}^{\rho_m}$, $E_{c,j}^{\rho_m}$	Time and energy spent at the jth node for the computation of the ρ_mth task
$T_{tx,jk}^{\rho_j}(\tau_i)$, $T_{rx,kj}^{\rho_j}(\tau_i)$	Transmission and reception time for the ρ_mth task between nodes j and k
$E_{tx,jk}^{\rho_j}(\tau_i)$, $E_{rx,kj}^{\rho_j}(\tau_i)$	Transmission and reception energy for the ρ_mth task between nodes j and k
$T_{v_m,j}^{\text{off}}(\alpha_{\rho_m}(\tau_i^{\text{off}}))$	Time spent for task offloading
$E_{v_m,j}^{\text{off}}(\alpha_{\rho_m}(\tau_i^{\text{off}}))$	Energy spent for task offloading
$T_{v_m}^{\text{loc}}(\alpha_{\rho_m}(\tau_i^{\text{off}}))$	Time spent for local computation
$E_{v_m}^{\text{loc}}(\alpha_{\rho_m}(\tau_i^{\text{off}}))$	Energy spent for local computation
$T_{v_m,j}^{\rho_m}(\alpha_{\rho_m}(\tau_i^{\text{off}}))$	Time spent for whole task computation
$E_{v_m,j}^{\rho_m}(\alpha_{\rho_m}(\tau_i^{\text{off}}))$	Energy spent for whole task computation

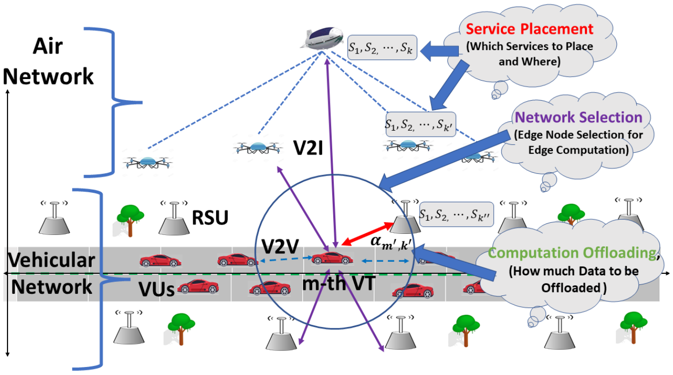
Air
Network
Vehicular
Network
$S_1, S_{2,} \cdots, S_k$
Service Placement
(Which Services to Place and Where)
$S_1, S_{2,} \cdots, S_{k'}$
Network Selection
(Edge Node Selection for Edge Computation)
V2I
RSU
V2V
$\alpha_{m',k'}$
$S_1, S_{2,} \cdots, S_{k''}$
VUs
m-th VT
Computation Offloading,
(How much Data to be Offloaded)

Fig. 1. The T/NT Integrated Scenario.

relatively slow speed compared to VTs and is characterized by a processing capability equal to $c_{u,p}$ FLOPS per CPU cycle, with CPU frequency $f_{u,p}$ and $\mathcal{L}_{u,p}$ CPU cores. In addition, its communication capabilities are supposed to be identified through a communication technology able to work on a bandwidth $B_{u,p}$. Each UAV can serve a set of VTs and RSUs in its coverage space. Here, the pth UAV, with its limited resources, can provide up to $\hat{\mathcal{S}}^2_{up} \subset \mathcal{S}$ services to the VTs. Being a centralized node with powerful computing and communication resources, we assume that the HAP node can provide the entire set of services $\mathcal{S}$ to the VTs in its coverage range of R_h meters, with a computation capacity equal to c_h FLOPS per CPU cycle, with CPU frequency f_h and $\mathcal{L}_h$ CPU cores and having bandwidth B_h. Also, the HAP node is located at altitude $\bar{h}_h$. The basic components of the system and the different communication links between them are illustrated in Figure 1.

2.1 VT Mobility Model

Compared with highly dynamic VTs, aerial network platforms move more slowly and often have negligible impacts on overall mobility parameters of VTs, i.e., relative distance, speed, location, etc. Also, these platforms can follow predefined mobility patterns based on operators' settings. Therefore, in this work, we consider that air networking nodes (i.e., UAVs and HAP) are located in a fixed position in a given interval of time, while VTs move with variable speed $\vec{v}_m(\tau_i)$. We suppose that the speed of the VTs is bounded within $\vec{v}_{\min}$ and $\vec{v}_{\max}$ while the instantaneous speed of the mth VT is modeled through a truncated normal distribution density function [3]:

$$f\left(\vec{v}_m(\tau_i)\right) = \begin{cases} \dfrac{2e^{\frac{-(\vec{v}_m(\tau_i)-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}\left(\operatorname{erf}\left(\frac{\vec{v}_{\max}-\mu}{\sigma\sqrt{2}}\right)-\operatorname{erf}\left(\frac{\vec{v}_{\min}-\mu}{\sigma\sqrt{2}}\right)\right)}, & \\ \qquad\qquad \vec{v}_{\min} \leq \vec{v}_m(\tau_i) \leq \vec{v}_{\max} & \\ 0, \quad \text{else} & \end{cases} \tag{1}$$

where μ and σ are the mean and standard deviation of the vehicles speed, and $\operatorname{erf}(x)$ is the Gauss error function over x. The path length, within which the mth VT remains under the coverage of the jth node (i.e., RSU, UAV or HAP), is given by $D_{v_m,j}(\tau_i) = \sqrt{R_j^2 - \left(y_j - y_{v_m}(\tau_i)\right)^2} \pm \left(x_j - x_{v_m}(\tau_i)\right)$ where, $\left(x_{v_m}(\tau_i), y_{v_m}(\tau_i)\right)$ is the location of the mth VT at τ_i and $\left(x_j, y_j\right)$ is the projection over the ground of a generic jth node, which can be RSU, UAV or HAP. The available sojourn time for the mth VT with respect to a generic jth node is $T^{\text{soj}}_{v_m,j}(\tau_i) = \frac{D_{v_m,j}(\tau_i)}{|\vec{v}_m(\tau_i)|}$.

2.2 Multi-time Scale Approach

The service placement operations often require larger time intervals for updating over different EN mainly due to the virtual service activation latency, longer backhaul delays, etc. [28]. Here, service placement operations are performed at discrete time intervals lasting Δ^{sp}, where τ_i^{sp} identifies the ith time interval, i.e., $\tau_i^{\text{sp}} = \{\forall t | t \in [i\Delta^{\text{sp}}, (i+1)\,\Delta^{\text{sp}}]\}$. Network selection decisions can be made on a moderate time scale compared with the service placement problem. The time scale is modeled in a time-discrete manner, with a time interval Δ^{ns}, where τ_i^{ns} identifies the ith time interval, i.e., $\tau_i^{\text{ns}} = \{\forall t | t \in [i\Delta^{\text{ns}}, (i+1)\,\Delta^{\text{ns}}, 0 < (i+1)\,\Delta^{\text{ns}} \leq \Delta^{\text{sp}}]\}$. Note that the maximum number of time steps is a function of Δ^{sp}, corresponding to the time step of the service placement problem. With high dynamicity and frequent task requests, the computation offloading decisions should be performed on a shorter time scale. The time scale for the offloading process is discrete in time with a time interval Δ^{off}, where τ_i^{off} identifies the ith time interval, i.e., $\tau_i^{\text{off}} = \left\{\forall t | t \in \left[i\Delta^{\text{off}}, (i+1)\,\Delta^{\text{off}}, 0 < (i+1)\,\Delta^{\text{off}} \leq \Delta^{\text{ns}}\right]\right\}$. Note that the maximum number of offloading time steps is based on the Δ^{ns} value, which corresponds to the time step of the network selection problem.

Over time, vehicular service placement needs to be updated to serve end-users according to their demands, as well as to use appropriate resources at EC facilities. We define a binary service placement parameter $b(S_s, j, l, \tau_i^{\text{sp}})$ as,

$$b(S_s, j, l, \tau_i^{\text{sp}}) = \begin{cases} 1 & S_s \in \hat{\mathcal{S}}_j^l \\ 0 & \text{else} \end{cases}$$

with,

$$\sum_{s=1}^{\bar{S}} b(S_s, j, l, \tau_i^{\text{sp}}) \leq |\hat{\mathcal{S}}_j^l|, \quad \forall j \tag{2}$$

where $b(S_s, j, l, \tau_i) = 1$ models the network operators decision of placing the service S_s on the jth node at τ_i^{sp}. Notice that the service placement remains the same throughout the Δ^{sp} time interval, in which multiple network selection and offloading steps are performed.

On the basis of their limited coverage ranges, each VT can be covered by several RSUs, UAVs, and one HAP node. Here, we define a decision matrix $\mathbf{A}(M, J(l), l, \tau_i^{\text{ns}}) = \left\{a_{(v_m,j,l)}(\tau_i^{\text{ns}}) \in \{0,1\}\right\}$ with dimension $M \times J(l)$, where $J(l)$ is the amount of ENs in the lth layer. Here, $a_{(v_m,j,l)}$ is equal to 1 if the mth VT selects the jth EN from the layer l to offload its task, otherwise it takes the value 0. VTs can select RSU ($l = 1$), UAV ($l = 2$), or HAP ($l = 3$) to offload their data. Also, to avoid additional complexity, we consider that each VT can be assigned to only one EN which can be RSU, UAV, or HAP during the offloading process, thus,

$$\sum_{l=1}^{L} \sum_{j=1}^{J(l)} a_{(v_m,j,l)}(\tau_i^{\text{ns}}) = 1. \tag{3}$$

The number of VTs requesting services from the jth EN is given by $K_{j,l}(\tau_i^{\text{ns}}) = \sum_{m=1}^{M} a_{(v_m,j,l)}(\tau_i^{\text{ns}})$. With their limited resources, ENs can provide services to the VTs before task communication and computation costs become unbearable. We consider that $K_{j,l}^{\max}$ is the maximum number of VTs that can access the services of the jth node.

We assume that the system performs partial offloading, where tasks can be split and processed remotely while the remaining portion is processed locally [9]; the portion offloaded by the mth VT at τ_i^{off} is identified as $\alpha_{\rho_m}(\tau_i^{\text{off}}) \in [0,1]$. With multiple VTs requesting services, during the offloading process, the following constraints need to be taken into account:

$$\begin{cases} K_{j,l}(\tau_i^{\text{ns}}) \leq K_{j,l}^{\max} & \text{(4a)} \\ \displaystyle\sum_{m=1}^{K_{j,l}(\tau_i^{\text{ns}})} c_{j,l}^{\rho_m}(\tau_i^{\text{ns}}) \cdot f_{j,l}^{\rho_m}(\tau_i^{\text{ns}}) \leq (\mathcal{L}_{j,l} \cdot c_{j,l} \cdot f_{j,l}) & \text{(4b)} \\ \displaystyle\sum_{m=1}^{K_{j,l}(\tau_i^{\text{ns}})} b_{j,l}^{\rho_m}(\tau_i^{\text{ns}}) \leq B_{j,l} & \text{(4c)} \end{cases}$$

$$\forall i; j = 1, \ldots, J(l); l = 1, \ldots, L$$

where $c_{j,l}^{\rho_m}(\tau_i^{\text{ns}}) \cdot f_{j,l}^{\rho_m}(\tau_i^{\text{ns}})$ is the processing capacity of the jth EN from layer l assigned to the mth VTs task, $b_{j,l}^{\rho_m}(\tau_i^{\text{ns}})$ is the communication resource assigned to the VT for communicating with the jth EN. We consider that EN resources are shared equally among the requesting VTs. Eq. (4) models an upper bound on the number of users connected, processing capacity and the communication resources of the ENs.

2.2.1 Task Computation Model

The generic expression for the time and energy spent for the ρ_mth task computation on a jth device is given by [1]:

$$T_{c,j}^{\rho_m} = \frac{\Omega_{\rho_m}}{c_j^{\rho_m} f_j^{\rho_m}}, \quad E_{c,j}^{\rho_m} = T_{c,j}^{\rho_m} P_{c,j} \tag{5}$$

where $c_j^{\rho_m}$, $f_j^{\rho_m}$ and $P_{c,j}$ are the number of FLOPS, CPU-frequency per CPU-cycle assigned to the mth user, and computation power, respectively, whether j identifies a VT (v_m), a RSU (r_n), UAV (u_p) or a HAP (H_h).

2.2.2 Task Communication Model

For the case of a partial computation offloading, the transmission time and energy between a generic node j and a generic node k for task ρ_j is given by[1] $T_{tx,jk}^{\rho_j}(\tau_i) = \frac{D_{\rho_j}}{r_{jk}(\tau_i)}$ and $E_{tx,jk}^{\rho_j}(\tau_i) = T_{tx,jk}^{\rho_j}(\tau_i) Pt_j$, respectively, where $r_{jk}(\tau_i)$ is datarate of the link between the two nodes, while Pt_j is the transmission power of jth node. Similarly, the reception time and energy at the jth node to receive the task of size $D_{\rho_j}^r$ from kth EN are $T_{rx,kj}^{\rho_j}(\tau_i) = \frac{D_{\rho_j}^r}{r_{kj}(\tau_i)}$ and $E_{rx,kj}^{\rho_j}(\tau_i) = T_{rx,kj}^{\rho_j}(\tau_i) Pr_j$, respectively, where Pr_j is the power spent for receiving data. A symmetric channel is considered between j and k.

In this work, for modeling the characteristics of a channel between the jth and the kth node at ith interval [29], we consider that the link gain can be modeled as $h_{j,k}(\tau_i) = \beta_0 \cdot d_{j,k}^{\theta^k}(\tau_i)$, where $d_{j,k}(\tau_i)$ is the distance between node j and k at ith interval, β_0 is the channel power gain at 1 m reference distance, while θ^k is the path loss coefficient for the communication link between node j and k. The expression for the channel transmission rate is based on the Shannon capacity formula and can be written as:

$$r_{jk}(\tau_i) = b_l^{\rho_j}(\tau_i) \log_2 \left(1 + \frac{Pt_j \cdot h_{j,k}(\tau_i)}{N_0}\right) \quad \forall j, k \tag{6}$$

where Pt_j is the transmission power of a node j, $b_k^{\rho_j}(\tau_i)$ is the communication bandwidth, and $N_0 = N_T b_k^{\rho_j}(\tau_i)$ is the thermal noise power with noise power spectral density N_T.

2.2.3 Task Offloading Process

If mth VT is assigned to jth EN, then the time and energy required to offload the portion of task with offloading parameter α_{ρ_m} to the selected EN and to get back the result in the ith interval is given by:

$$T_{v_m,j}^{\text{off}}(\alpha_{\rho_m}(\tau_i^{\text{off}})) = \alpha_{\rho_m}(\tau_i^{\text{off}}) \left(T_{tx,v_m j}^{\rho_m}(\tau_i^{\text{off}}) + T_{c,j}^{\rho_m}(\tau_i^{\text{off}}) + T_{rx,j v_m}^{\rho_m}(\tau_i^{\text{off}})\right) \tag{7a}$$

$$E_{v_m,j}^{\text{off}}(\alpha_{\rho_m}(\tau_i^{\text{off}})) = \alpha_{\rho_m}(\tau_i^{\text{off}}) \left(E_{tx,v_m j}^{\rho_m}(\tau_i^{\text{off}}) + E_{rx,j v_m}^{\rho_m}(\tau_i^{\text{off}})\right) \tag{7b}$$

where both equations correspond to the sum of the terms related to the transmission, reception and, only for the time, local computation, introduced in Subsections 2.2.1 and 2.2.2. Similarly to other approaches, e.g., [4], [30], the analysis has been simplified by limiting to the user-side energy consumption; moreover, the energy consumption in idle state at the VT side is neglected.

2.2.4 Local Computation

From (5), the amount of time and energy required for the local computation of the remaining task in the ith interval can be written as the time and energy non-offloaded portions of task, i.e.,

$$T_{v_m}^{\text{loc}}(\alpha_{\rho_m}(\tau_i^{\text{off}})) = \left(1 - \alpha_{\rho_m}(\tau_i^{\text{off}})\right) T_{c,v_m}^{\rho_m} \tag{8a}$$

$$E_{v_m}^{\text{loc}}(\alpha_{\rho_m}(\tau_i^{\text{off}})) = \left(1 - \alpha_{\rho_m}(\tau_i^{\text{off}})\right) E_{c,v_m}^{\rho_m} \tag{8b}$$

2.2.5 Partial Computation Offloading

From (7) and (8), the delay and the energy consumed during the task processing phases when partial offloading is performed (in the ith offloading interval) can be written as:

$$T_{v_m,j}^{\rho_m}(\alpha_{\rho_m}(\tau_i^{\text{off}})) = \max\left\{T_{v_m,j}^{\text{off}}(\alpha_{\rho_m}(\tau_i^{\text{off}})), T_{v_m}^{\text{loc}}(\alpha_{\rho_m}(\tau_i^{\text{off}}))\right\}$$

$$E_{v_m,j}^{\rho_m}(\alpha_{\rho_m}(\tau_i^{\text{off}})) = E_{v_m,j}^{\text{off}}(\alpha_{\rho_m}(\tau_i^{\text{off}})) + E_{v_m}^{\text{loc}}(\alpha_{\rho_m}(\tau_i^{\text{off}}))$$

where the local and offloaded computing are supposed to be performed in parallel. Each VT should finish the offloading process and receive the results back within the sojourn time, hence:

$$T_{v_m,j}^{\text{off}}(\alpha_{\rho_m}(\tau_i^{\text{off}})) \leq T_{v_m,j}^{\text{soj}}(\tau_i^{\text{off}}) \quad \forall i \tag{9}$$

[1] In the following we identify with j and k the indexes of any generic node. Hence, j and k can have any index among v_m, r_n, u_p and H_h.

$$\mathbf{P1}: \min_{\mathcal{A},\mathbf{A},\mathcal{B}} \left\{ \frac{1}{T \cdot M} \sum_{\tau_{\bar{i}}^{\text{sp}}=0}^{T-1} \sum_{\tau_{\hat{i}}^{\text{ns}}=\tau_{\bar{i}}^{\text{sp}}}^{\tau_{\bar{i}}^{\text{sp}}+\Delta^{\text{sp}}-\Delta^{\text{ns}}} \sum_{\tau_{i}^{\text{off}}=\tau_{\hat{i}}^{\text{ns}}}^{\tau_{\hat{i}}^{\text{ns}}+\Delta^{\text{ns}}-\Delta^{\text{off}}} \left(\sum_{l=1}^{L} \sum_{j=1}^{J(l)} \sum_{m=1}^{M} \left[\gamma_1 T_{v_m,j}^{\rho_m} \left(\alpha_{\rho_m}(\tau_i^{\text{off}}) \right) + \gamma_2 E_{v_m,j}^{\rho_m} \left(\alpha_{\rho_m}(\tau_i^{\text{off}}) \right) + c(\tau_i^{\text{off}}) \right] \right) \right\} \tag{10}$$

2.2.6 Service Placement Penalty

In a multi-service VN, if the selected EN is unable to provide the requested service, VTs are impacted by additional costs due to, e.g., handover. There are several mechanisms that can be used in these scenarios. For instance, the chosen EN can transmit the VTs data to a neighboring EN that is capable of providing the requested service. Alternatively, the chosen EN can request a centralized orchestrator to provide the requested service. These operations may result in additional penalties in the overall offloading process. Hence, we propose the introduction of a generic service placement penalty during the offloading process, which is defined as:

$$c(\tau_i^{\text{off}}) = \begin{cases} \zeta & \text{if } b(S_{\rho_m}, j, l, \tau_i^{\text{sp}}) \neq 1 \wedge a_{(v_m,j,l)}(\tau_i^{\text{ns}}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

where ζ is the constant penalty when selected EN is not able to provide a requested service.

2.3 Problem Formulation

The main objective of this work is to optimize the network-wide performance of EC-enabled multiservice VN. We aim to optimize performance in terms of overall latency, energy, and service placement during the offloading process. Our main objective is to optimize the overall network cost by properly selecting ENs for service placement and computation offloading operations simultaneously. Additionally, offloading the optimal amount of data towards them further reduces overall latency and energy cost. For this, we formulate the joint latency, energy, and service placement cost minimization problem as in (10), subject to:

$$\mathbf{C1}: \text{Eq. (2)} \tag{11}$$

$$\mathbf{C2}: \text{Eq. (3)} \tag{12}$$

$$\mathbf{C3}: \text{Eqs. (4a), (4b) and (4c)} \tag{13}$$

$$\mathbf{C4}: \text{Eq. (9)} \tag{14}$$

$$\mathbf{C5}: T_{v_m,j}^{\rho_m} \left(\alpha_{\rho_m}(\tau_i^{\text{off}}) \right) \leq T_{\rho_m} \quad \forall \mathcal{V}, \forall i, j \tag{15}$$

$$\mathbf{C6}: E_{v_m,j}^{\text{off}}(\alpha_{\rho_m}(\tau_i^{\text{off}})) < E_{v_m}^{\text{loc}}(\alpha_{\rho_m}) \tag{16}$$

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

where $\mathcal{A} = \{\alpha_{\rho_m}(\tau_i^{\text{off}})\}^M$ is the computation offloading matrix, $\mathbf{A} = \{\mathbf{A}(M, J(l), l, (\tau_{\hat{i}}^{\text{ns}}))\}$ is the set of VT-EN assignment matrix, $\mathcal{B} = \{b(S_s, j, (\tau_{\bar{i}}^{\text{sp}}))\}$ is the service placement matrix, γ_1 and γ_2 are weight coefficients for balancing latency and energy consumption, and T is the whole time interval considered. **C1** puts a limit on the maximum number of services placed in each EN j. **C2** stands that each VT can select at most one EN for the computation offloading. **C3** provides the limits over the number of user requests, processing capacity and bandwidth resource blocks requested by VTs towards ENs. According to **C4**, to avoid handover phenomena and related latency, each VT should complete the offloading process before it passes through the selected EN coverage. **C5** puts a limit on the maximum processing time as one of the task requirements. In order to have a valid offloading process, according to **C6**, the weighted energy consumed on VT for processing a complete task should be lower than the total weighted energy required to compute a complete task locally. **C7** stands that the two weighting coefficients (γ_1, γ_2) should be between 0 and 1 with a sum equal to 1.

3 Multi-time Scale Optimization

In the scenario considered, the *service placement decisions* taken by the centralized operator can impact the users' network selection possibilities and the corresponding outcomes in terms of accessing services at reduced costs. On the other hand, the *network selection decisions* made by VTs can further impact the *offloading decision* and, thus, corresponding task processing costs. Therefore, these processes and the decisions associated with them can create a hierarchical structure of decisions that influence each other's performances. In addition to this, these decisions should be made at different time scales (i.e., $\Delta^{\text{sp}}, \Delta^{\text{ns}}, \Delta^{\text{off}}$). This leads to a multi-time scale optimization process involving multiple layers of decisions impacting each other. Multi-time-scale optimization has been a well-established area of research, focusing on modeling various realistic situations where multiple processes exhibit different time-scale behaviors. These situations can be effectively addressed as multi-time-scale problems. Traditionally, heuristic approaches have been employed to find solutions to such problems, but these approaches are only consistent in specific scenarios. The use of MDPs provides a suitable framework for modeling the evolving states over time. In this study, we propose an extension to the traditional MDP approach by introducing a hierarchical MDP model, where each level of the hierarchy can effectively capture a specific timing aspect. The considered multi-time-scale optimization problem can be solved effectively through sequential decision-making processes, e.g., MDP, where a multi-time scale MDP model [31] can be considered to solve the problem of joint service placements, network selection, and computation offloading effectively. In the following, we define a Multi-Time scale MDP (MDP-MT) as represented in Fig. 2 through several basic elements discussed in the following.

3.1 Service Placement MDP

The service placement MDP is a model that addresses the network service placement problem in resource-constrained EN to meet the demands of vehicular users. Decisions about service placement can be made over a longer time period, denoted as Δ^{sp}.

The discrete state space for the service placement MDP is defined as $\mathcal{S}^{\text{sp}} = \{s_1^{\text{sp}}, \cdots, s_u^{\text{sp}}, \cdots, s_U^{\text{sp}}\}$ with maximum

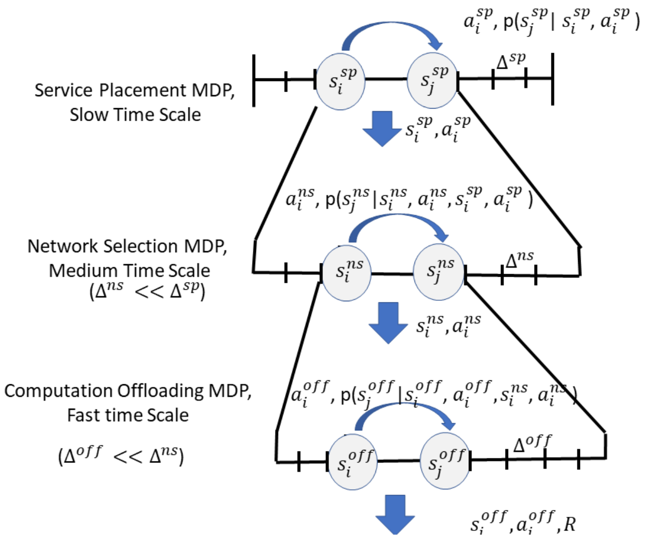
a_i^{sp}, p(s_j^{sp}| s_i^{sp}, a_i^{sp})
Δ^{sp}
Service Placement MDP, Slow Time Scale
s_i^{sp}
s_j^{sp}
s_i^{sp}, a_i^{sp}
a_i^{ns}, p(s_j^{ns}|s_i^{ns}, a_i^{ns}, s_i^{sp}, a_i^{sp})
Network Selection MDP, Medium Time Scale (Δ^{ns} << Δ^{sp})
s_i^{ns}
s_j^{ns}
Δ^{ns}
s_i^{ns}, a_i^{ns}
Computation Offloading MDP, Fast time Scale (Δ^{off} << Δ^{ns})
a_i^{off}, p(s_j^{off}|s_i^{off}, a_i^{off}, s_i^{ns}, a_i^{ns})
s_i^{off}
s_j^{off}
Δ^{off}
s_i^{off}, a_i^{off}, R

Fig. 2. Multi-scale MDP Model for the Service Placement, Network Selection, and Offloading Problem.

U states. $\mathcal{S}^{\text{sp}}$ is modeled as a function of the number of EN available and the service placement updates over time. Thus,

$$s_u^{\text{sp}}(\tau_i^{\text{sp}})=\{N(\tau_i^{\text{sp}}), P(\Delta_i^{\text{sp}}), H(\tau_i^{\text{sp}}), \hat{P}_R(\tau_i^{\text{sp}}), \hat{P_U}(\tau_i^{\text{sp}}), \hat{P}_H(\tau_i^{\text{sp}})\}$$

where $\hat{P}_R(\tau_i^{\text{sp}})_{N(\tau_i^{\text{sp}})\times S}$, $\hat{P}_U(\tau_i^{\text{sp}})_{P(\tau_i^{\text{sp}})\times S}$, and $\hat{P}_H(\tau_i^{\text{sp}})_{H(\tau_i^{\text{sp}})\times S}$ are the binary matrices modeling the change in the service placement over different edge layers at τ_i^{sp}. For example, if the selected action places the sth service on the nth RSU then $\hat{P}_R(\tau_i^{\text{sp}})(i, j) = 1$, else it takes value zero.

A discrete set of actions is defined through $\mathcal{A}^{\text{sp}} = \{a_1^{\text{sp}}, \cdots, a_{\bar{u}}^{\text{sp}}, \cdots, s_{\bar{U}}^{\text{sp}}\}$ with maximum $\bar{U}$ actions. $\mathcal{A}^{\text{sp}}$ includes all the feasible service placement options that an orchestrator can use. For limiting the complexity of the MDP we have assumed that all the ENs from the same edge layer have the same subset of services placed on them.

The performance of service placement MDP is measured through the feedback signal generated as a sum of the total reward received during the offloading and network selection process as shown in Fig. 3, i.e., the hierarchical feedback process.

3.2 Network Selection MDP

The MDP model for network selection addresses the problem of selecting a network in a scenario where decisions are made on a moderate time scale Δ^{ns}, which is different from the time scale of the service placement problem. The goal of the network selection problem is to find an appropriate EN that can provide the desired service. If the selected EN does not have the requested service available, there may be additional costs in terms of service handovers. Therefore, the decisions made by the network selection MDP are influenced by the current state-action pairs of the service placement MDP model. Furthermore, various local environment parameters, such as competing VTs, available ENs and their states, and the type of requested service, can also impact the network selection decision. It is challenging to provide a single model that can handle all dynamic situations, as it may result in reduced overall performance. In light of this, we incorporate local environment parameters obtained through V2X technology into the MDP process by representing it through various scenarios. In particular, vehicular scenarios set $\Omega = \{k_1, \cdots, k_g, \cdots, k_G\}$ based upon local vehicular density $\mathcal{D}$, the number of RSUs R_m, UAVs U_m and HAP nodes H_h covering the VT and the requested service type. Here G is the maximum number of considered scenarios. Thus the generic gth scenario, k_g, is defined as a tuple $k_g = \langle \mathcal{D}, R_m, U_m, R_m, S_{\rho_m} \rangle$ with,

$$\mathcal{D} = \begin{cases} 0 & \text{if } 1 \leq M < M_1 \\ 1 & \text{if } M_1 \leq M < M_2 \\ 2 & \text{if } M_2 \leq M \end{cases}$$

where M_1 and M_2 are parameters introduced to classify VT traffic scenarios into low, medium, and high density.

Next, we define the state-space for the network selection MDP as $\mathcal{S}^{\text{ns}} = \{s_m^{\text{ns}}(\tau_i^{\text{ns}})\}$ with $s_m^{\text{ns}}(\tau_i^{\text{ns}}) \in \{(s_{v_m}^{\text{ns}}, s_e^{\text{ns}})\}$. The individual state $s_m^{\text{ns}}(\tau_i^{\text{ns}})$ is based on the VT side state $(s_{v_m}^{\text{ns}})$, and the selected ENs state s_e^{ns} where e can be an RSU, UAV or HAP node. The VTs state $s_m^{\text{ns}}(\tau_i^{\text{ns}})$ is modeled by the requested service S_{ρ_m}, $d_{m,e}$, the distance between VT and the EN, and $D_{v_m,e}$, the distance before VT passes through the coverage area of e.

The action space for VTs in scenario g is defined as $\mathcal{A}_{\kappa_g}^{\text{sp}} = \{a_{\bar{m},j}^{\text{ns}}(\tau_i^{\text{ns}})\}$ for the network selection MDP corresponds to all possible sets of actions with individual actions, $a_{\bar{m},g}^{\text{ns}}(\tau_i^{\text{ns}}) = [\{0,1\}_{1\times R_m}, \{0,1\}_{1\times U_m}, \{0,1\}_{1\times H_m}]$, with $\sum a_{\bar{m},g}^{\text{ns}}(\tau_i^{\text{ns}}) = 1$

The performance of network selection MDP is measured through the feedback signal generated as a sum of the rewards received during the offloading process, as shown in Fig. 3.

3.3 Computation Offloading MDP

The computation offloading problem involves determining the appropriate amount of data to be offloaded to a selected EN. Given the high level of dynamism and frequent task requests, these decisions must be made quickly, within a time scale denoted Δ^{off}. The computation offloading process should be completed before the VT passes through the coverage range of the selected EN. Additionally, the entire task processing operation must be performed within a specified task latency requirement, and an optimal amount of data should be offloaded to minimize the energy costs associated with VTs' local data computation and data transmission operations. Decisions made during network selection and service placement MDPs can impact the performance of the offloading MDP. If an incorrect EN is selected during the network selection process or if services are not properly aligned between different ENs, performance during the offloading phases may be limited.

By taking into account the various performance requirements of MDP, here we introduce a discrete state space for the offloading MDP problem that is based upon following three binary functions that model the behavior of offloading MDP over time. If the mth VT is assigned to the nth EN and performs offloading operation with offloading parameter

α_{ρ_m}, the environment can be modeled through three proper binary functions, as:

$$F^1_{\rho_m,n}(\tau_i^{\text{off}}) = \begin{cases} 0 & T^{\text{off}}_{m,n}(\alpha_{\rho_m}(\tau_i^{\text{off}})) \leq T^{\text{soj}}_{m,n}(\tau_i^{\text{off}}) \\ 1 & \text{else} \end{cases}$$

$$F^2_{\rho_m,n}(\tau_i^{\text{off}}) = \begin{cases} 0 & T^{\rho_m}_{m,n}\left(\alpha_{\rho_m}(\tau_i^{\text{off}})\right) \leq T_{\rho_m} \\ 1 & \text{else} \end{cases}$$

$$F^3_{\rho_m,n}(\tau_i^{\text{off}}) = \begin{cases} 0 & E^{\text{off}}_{m,n}(\alpha_{\rho_m}(\tau_i^{\text{off}})) < w_1 E^{\rho_m}_{c,m} \\ 1 & \text{else} \end{cases}$$

where $F^1_{\rho_m,n}(\tau_i^{\text{off}})$, $F^2_{\rho_m,n}(\tau_i^{\text{off}})$ and $F^3_{\rho_m,n}(\tau_i^{\text{off}})$ are the binary functions depending upon the sojourn time constraint (9), application latency requirement (15) and the energy constraint (16), respectively.

The discrete state space for the computation offloading MDP is defined as $\mathcal{S}^{\text{off}} = \left\{s_1^{\text{off}}, \cdots, s_l^{\text{off}}, \cdots, s_L^{\text{off}}\right\}$ with maximum L states with the individual state defined as $s_l^{\text{off}}(\tau_i^{\text{off}}) = \left\{F^1_{\rho_m,n}(\tau_i^{\text{off}}), F^2_{\rho_m,n}(\tau_i^{\text{off}}), F^3_{\rho_m,n}(\tau_i^{\text{off}})\right\}$. The action space $\mathcal{A}^{\text{off}} = \{a_{\bar{l}}^{\text{off}}\}$ for the computation offloading MDP is defined as $\mathcal{A}^{\text{off}} = [0, \Lambda, 2\Lambda, \cdots, 1]$, where Λ is the step change in the value of offloading parameter α_{ρ_m}.

3.4 Reward Function

The performance of the MDP can be described through a joint reward function defined as,

$$\begin{aligned} r^{\text{off}}(s_l^{\text{off}}) &= \gamma_1 T^{\rho_m}_{v_m,j}\left(\alpha_{\rho_m}(\tau_i^{\text{off}})\right) + \gamma_2 E^{\rho_m}_{v_m,j}(\alpha_{\rho_m}(\tau_i^{\text{off}})) \\ &+ c(\tau_i^{\text{off}}) + F^1_{\rho_m,n}(\tau_i^{\text{off}}) + F^2_{\rho_m,n}(\tau_i^{\text{off}}) + F^3_{\rho_m,n}(\tau_i^{\text{off}}) \quad (18) \end{aligned}$$

The total reward received is the sum of latency, energy costs, service placement, and additional constraint failure penalties.

4 Deep Q-Learning for Service Placement, Network Selection, and Computation Offloading

The previous section introduced the various components of the MDP models. By solving these models, VTs can discover an appropriate service placement, node selection, and offloading amount that can effectively reduce total latency, energy consumption, and service placement costs.

For any instant in time τ_i, the state space $\mathcal{ST}$ is equal to $\{s(\tau_i)\}$, where $s(\tau_i) = (s_l^{\text{off}}, s_m^{\text{ns}}, s_u^{\text{sp}})$ is the instantaneous state of a multi-time-scale MDP, i.e., a combination of the three MDPs states. For the large time interval Δ^{sp}, s_u^{sp} remains unchanged while multiple transactions can occur for s_l^{off} and s_m^{ns}. The solutions can be defined as a policy function $\pi \in \Pi$:

$$\pi = \left\{\pi^{\text{off}}(s_l^{\text{off}}(\tau_i + \delta)), \pi^{\text{ns}}(s_m^{\text{ns}}(\tau_i + \delta)), \pi^{\text{sp}}(s_u^{\text{sp}}(\tau_i + \delta))\right\}$$

that maps every state $s \in \mathcal{ST}$ to action $a = \{(a_{\bar{l}}^{\text{off}}, a_{\bar{m}}^{\text{ns}}, a_{\bar{u}}^{\text{sp}})\} \in \mathcal{AS}$. Given the offloading policy π^{off}, network selection policy π^{ns}, and the lower level reward R^{off}, over a large time scale Δ^{sp}, we define a $\Delta^{\text{sp}}/\Delta^{\text{off}}$-horizon total expected reward as (19), where

$$\sigma_1(n\Delta^{\text{sp}}/\Delta^{\text{ns}} + r) = r, \quad \forall n > 0, r = 0, \cdots, \Delta^{\text{sp}}/\Delta^{\text{ns}},$$
$$\sigma_2(n\Delta^{\text{ns}}/\Delta^{\text{off}} + r) = r, \quad \forall n > 0, r = 0, \cdots, \Delta^{\text{ns}}/\Delta^{\text{off}},$$
$$0 < \alpha_1, \alpha_2 \leq 1,$$

and s_0^{ns} and s_0^{off} are the initial state values for Δ_k^{sp} and Δ_i^{ns}, respectively. Here, the total expected reward achieved by the offloading and network selection level MDPs will act as a single-step reward for a service placement MDP.

Selecting different actions can result in different policy functions, where the aim is to find an optimal policy that corresponds to the minimum delay and energy cost during vehicular task processing. For every policy π, a value function $V_\pi(s(\tau_i))$, corresponding to a state $s(\tau_i)$ can be defined for analyzing its performance. In general, $V_\pi(s(\tau_i))$ corresponds to an expected value of a discounted sum of total reward received by following the policy π from state $s(\tau_i)$, and can be defined as (19). The optimal policy π^* corresponding to the value function V is defined using the Bellman equation as in (20). Here, $P^{\text{off}}_{(s_l^{\text{off}}, s_{\hat{l}}^{\text{off}})}(\pi_l^{\text{off}}(s_l^{\text{off}}))$, $P^{\text{ns}}_{(s_m^{\text{ns}}, s_{\hat{m}}^{\text{ns}})}(\pi_m^{\text{ns}}(s_m^{\text{ns}}))$, and $P^{\text{sp}}_{(s_u^{\text{sp}}, s_{\hat{u}}^{\text{sp}})}(a_u^{\text{sp}})$ model the environment dynamics based upon the state transition probabilities. In addition, R^{off} is the mean value of the immediate reward r^{off} defined in (18).

Due to the intricate and ever-changing nature of the vehicular scenario under consideration, it is challenging to accurately determine the environmental dynamics. In light of this, we employ a model-free RL approach to solve the multi-time-scale MDP model and identify the optimal policies. Q-learning, among various other model-free strategies, has been extensively investigated for its ability to determine the optimal π^* in unfamiliar environments. The Q-learning strategy is based on a state-action function, i.e., the Q-function, defined as,

$$Q^\pi(s', a') = R(s', a') + \gamma \sum_{\hat{s} \in S} P_{s'\hat{s}}(a') V^\pi(\hat{s})$$

representing a discounted cumulative reward from state s' when action a' is taken before following the policy π. The optimal Q value can be represented as

$$Q^{\pi^*}(s', a') = R(s', a') + \gamma \sum_{\hat{s} \in S} P_{s'\hat{s}}(a') V^{\pi^*}(\hat{s})$$

where $V^{\pi^*}(\hat{s}) = \min_{\hat{a} \in A} Q^{\pi^*}(s', \hat{a})$. The Q values can be estimated through a recursive approach, where

$$Q_{t+1}(s', a') = Q_t(s', a') + \epsilon \cdot \left(r + \gamma \max_{\hat{a}} Q_t(s', \hat{a}) - Q_t(s', a')\right)$$

and ϵ is a learning rate. The Q function can be estimated using a neural network-based function approximation technique with $Q(s', a'; \theta) \approx Q(s', a')$, where θ represents the weights of the neural network. Through the training process, the values of θ can be adjusted to reduce the mean square error values.

In the Deep Q Network (DQN) based approach, two networks (i.e., primary and target Q networks) are considered for a reliable estimation of Q functions over different time scales. The primary network estimates the real/primary Q-value while the target Q-values are estimated through the target network. The RL agent uses the backpropagation and gradient descent processes with mean square error (MSE)-based loss function for reducing the gap between the

$$R^{\text{sp}}(s_u^{\text{sp}}(\Delta_k^{\text{sp}}), a_{\bar{u}}^{\text{sp}}(\Delta_k^{\text{sp}}), s_m^{\text{ns}}, \pi_m^{\text{ns}}, s_l^{\text{off}}, \pi_l^{\text{off}}) = \mathbb{E}_{s_u^{\text{sp}}, a_{\bar{u}}^{\text{sp}}}^{s_0^{\text{ns}}, s_0^{\text{off}}} \left\{ \sum_{\tau_i^{\text{ns}} = \tau_k^{\text{sp}}}^{(\tau_k^{\text{sp}} + \Delta^{\text{sp}} - \Delta^{\text{ns}})} \alpha_1^{\sigma_1(\tau_i^{\text{ns}})} \sum_{\tau_j^{\text{off}} = \tau_i^{\text{ns}}}^{(\tau_i^{\text{ns}} + \Delta^{\text{ns}} - \Delta^{\text{off}})} \alpha_2^{\sigma_2(\tau_j^{\text{off}})} R^{\text{off}} \Big(s_m^{\text{ns}}(\tau_i^{\text{ns}}), \pi^{\text{ns}}\big(s_m^{\text{ns}}(\tau_i^{\text{ns}}), s_u^{\text{sp}}(\tau_k^{\text{sp}}), a_{\bar{u}}^{\text{sp}}(\tau_k^{\text{sp}})\big), s_l^{\text{off}}(\tau_j^{\text{off}}), \right.$$

$$\left. \pi^{\text{off}}\Big(s_l^{\text{off}}(\tau_j^{\text{off}}), s_m^{\text{ns}}(\tau_i^{\text{ns}}), \pi^{\text{ns}}\big(s_m^{\text{ns}}(\tau_i^{\text{ns}}), s_u^{\text{sp}}(\tau_k^{\text{sp}}), a_{\bar{u}}^{\text{sp}}(\tau_k^{\text{sp}})\big), s_u^{\text{sp}}(\tau_k^{\text{sp}}), a_{\bar{u}}^{\text{sp}}(\tau_k^{\text{sp}})\Big)\Big) \right\} \quad (19)$$

$$V(s_l^{\text{off}}, s_m^{\text{ns}}, s_u^{\text{sp}})^{\pi^*} = \min_{a_u^{\text{sp}} \in \mathcal{A}^{\text{sp}}} \left\{ \min_{\pi_m^{\text{ns}} \in \Pi^{\text{ns}}} \left\{ \min_{\pi_l^{\text{off}} \in \Pi^{\text{off}}} \left\{ R^{\text{sp}}\left(s_u^{\text{sp}}, a_u^{\text{sp}}, s_m^{\text{ns}}, \pi_m^{\text{ns}}, s_l^{\text{off}}, \pi_l^{\text{off}}\right) \right.\right.\right.$$

$$\left.\left.\left. + \gamma \sum_{\forall s_{\hat{l}}^{\text{off}}} \sum_{\forall s_{\hat{m}}^{\text{ns}}} \sum_{\forall s_{\hat{u}}^{\text{sp}}} P_{(s_l^{\text{off}}, s_{\hat{l}}^{\text{off}})}^{\text{off}}(\pi_l^{\text{off}}(s_l^{\text{off}})) P_{(s_m^{\text{ns}}, s_{\hat{m}}^{\text{ns}})}^{\text{ns}}(\pi_m^{\text{ns}}) P_{(s_u^{\text{sp}}, s_{\hat{u}}^{\text{sp}})}^{\text{sp}}(a_u^{\text{sp}}) V^*(s_{\hat{l}}^{\text{off}}, s_{\hat{m}}^{\text{ns}}, s_{\hat{u}}^{\text{sp}}) \right\}\right\}\right\} \quad (20)$$

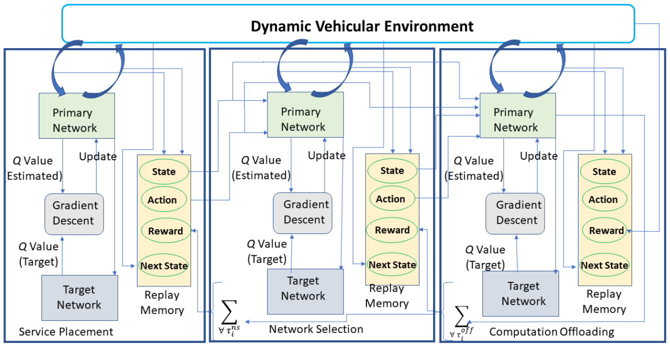

Fig. 3. Proposed Deep Q-Learning Solution

primary and the target Q-values where the loss function is defined as:

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{\hat{a}} Q_t(s', \hat{a}, \theta') - Q(x, a, \theta)\right)^2\right] \quad (21)$$

where the primary values $Q(x, a, \theta)$ are based on primary network parameters θ, and $r + \gamma \max_{\hat{a}} Q_t(s', \hat{a}, \theta')$ is the target Q value based upon the target network parameters θ'.

The Deep Q-learning method for solving the multi-time-scale MDP model to determine the optimal service placements, network selection, and offloading policies is illustrated in Fig. 3. The problem is tackled using three different DQN architectures, each designed for a specific time scale. In the case of service placement, the DQN architectures consist of a primary network and a target network, each having $\bar{K}^{\text{sp}}$ layers with $\bar{n}_{\bar{k}}^{\text{sp}}$ neurons. Here, $\bar{k}$ ranges from 1 to $\bar{K}^{\text{sp}}$. Similarly, for each scenario k_g, we consider network selection and computation offloading DQN with $\bar{K}_g^{\text{ns}}$ and $\bar{K}_g^{\text{off}}$ layers, and neurons $\bar{n}_{g,\bar{k}}^{\text{ns}}$ and $\bar{n}_{g,\bar{k}}^{\text{off}}$. Here, $\bar{k}$ ranges from 1 to $\bar{K}_g^{\text{ns}}$ for network selection and from 1 to $\bar{K}_g^{\text{off}}$ for computation offloading. Additionally, we consider replay memories with sizes $\mathcal{D}^{\text{sp}}$, $\mathcal{D}_g^{\text{ns}}$, and $\mathcal{D}_g^{\text{off}}$ to store the agents' past experiences.

At first, the primary network associated with the service placement problem detects the current state of the vehicular environment. Decisions about service placement are updated periodically, with a time scale denoted τ_i^{sp}. These updates are based on a proper action determined by the Epsilon Greedy Policy (EGP), which is controlled by the parameter e^{sp}. The state-action pair $(s_u^{\text{sp}}(\tau_i^{\text{sp}}), a_{\bar{u}}^{\text{sp}}(\tau_i^{\text{sp}}))$ is then transmitted to the other two DQNs. The primary network of a second DQN detects the current state of the environment and uses EGP with parameter e_g^{ns} on each Δ^{ns} interval to update the network selection decisions. The state-action pairs $(s_u^{\text{ns}}(\tau_i^{\text{ns}}), a_{\bar{u}}^{\text{ns}}(\tau_i^{\text{ns}}))$ are then sent to the offloading DQN for further processing. The decisions regarding offloading are determined by the current dynamics of the environment, which are influenced by $a_{\bar{u}}^{\text{sp}}(\tau_i^{\text{sp}})$, $a_{\bar{u}}^{\text{ns}}(\tau_i^{\text{ns}})$, and the EGP strategies with parameter $e^{\text{off},g}$ over Δ^{off} intervals. Additionally, the system receives an instantaneous reward, which, along with other entities such as the current state, action, and next state, is stored in the replay buffer. The cumulative reward of the offloading process is then back-propagated to the network selection and service placement DQNs, considering multiple offloading decisions.

Algorithm 1 details the DQN process for the multi-time-scale service placement, network selection, and computation offloading problem. The process begins with the definition and initialization of primary and target networks for the tasks under consideration (Line 1-2). After that, the training process iterates over $\bar{N}$ training iterations, where in each iteration DQN models are updated (Line 3-33). In each training iteration, an initially random set of states is selected (Line 5). Next, throughout several epochs, denoted as I_{SP}, the model undergoes training using a gradient descent method. This involves adjusting the weight values to minimize the loss function, which is dependent on the target and primary Q values. During each training epoch, the action a^{ns} is chosen using the EGP method with a probability of e. This action is then applied to the service placement network to generate the next state (Line 13-14). Since the reward is dependent on the performance of moderate and small time scale MDPs, the MDP models for network selection and offloading processes corresponding to different scenarios are trained in a similar manner over the I_{ns} and I_{off} epochs, respectively.

Algorithm 1 Multi-time Scale Deep Q-Learning

```
Input: $\Delta^{\text{sp}}, \Delta^{\text{ns}}, \Delta^{\text{off}}, \mathcal{S}^{\text{sp}}, \mathcal{S}^{\text{ns}}, \mathcal{S}^{\text{off}}, \mathcal{A}^{\text{sp}}, \mathcal{A}^{\text{ns}}, \mathcal{A}^{\text{off}}, \mathcal{S}, \mathcal{V}, \mathcal{R}, \mathcal{U}, h, \bar{N},$
       $\mathcal{I}_{\text{sp}}, \mathcal{I}_{\text{ns}}, \mathcal{I}_{\text{off}}, \bar{i}, G, e^{\text{sp}}, e_g^{\text{ns}}, e_g^{\text{off}}, \mathcal{D}^{\text{sp}}, \mathcal{D}_g^{\text{ns}}, \mathcal{D}_g^{\text{off}},$
       $\epsilon^{\text{sp}}, \epsilon_g^{\text{ns}}, \epsilon_g^{\text{off}}, \epsilon^{\text{sp}}, \gamma_g^{\text{ns}}, \epsilon_g^{\text{off}}$
Output: $w^{p,\text{sp}}, \{w_g^{p,\text{ns}}, w_g^{p,\text{off}}, \forall g \in K\}$
Initialize $w^{p,\text{sp}}, \{w_g^{p,\text{ns}}, \forall g \in K\}, \{w_g^{p,\text{off}}, \forall g \in K\}$
Duplicate policy networks to Target Networks,
        i.e., $w^{T,\text{sp}} = w^{p,\text{sp}}, \{w_g^{T,\text{ns}} = w^{p,\text{ns}}\}, \{w_g^{T,\text{off}} = w^{p,\text{off}}\}$
for all $ep = 1, \dots, \bar{N}$ do
    Select Random $s_0^{\text{sp}}, \{s_{0,g}^{\text{ns}}\}, \{s_{0,g}^{\text{off}}\}$
    $s^{\text{sp}} \leftarrow s_0^{\text{sp}}, \{s_g^{\text{ns}} \leftarrow s_{0,g}^{\text{ns}}, \quad s_g^{\text{off}} \leftarrow s_{0,g}^{\text{off}}, \forall g\}, \quad it = 0$
    while $i_{\text{sp}} \neq \mathcal{I}_{\text{sp}}$ do
        $i_{\text{sp}} = i_{\text{sp}} + 1$
        Select action $a^{\text{sp}} \in \mathcal{A}^{\text{sp}}$ with probability $e^{\text{sp}}$
        Determine next state $(s_{\text{new}}^{\text{sp}})$
        for all $g = 1, \dots, G$ do
            while $i_{\text{ns}} \neq I_{\text{ns}}$ do
                $i_{\text{ns}} = i_{\text{ns}} + 1$
                Select action $a^{\text{ns}} \in \mathcal{A}^{\text{ns}}$ with probability $e_g^{\text{ns}}$
                Determine next state $(s_{\text{new}}^{\text{ns}})$
                while $i_{\text{off}} \neq I_{\text{off}}$ do
                    Select $a^{\text{off}} \in \mathcal{A}^{\text{off}}$ with probability $e_g^{\text{off}}$
                    Find next state $s_{\text{new}}^{\text{off}}$ and reward $R^{\text{off}}$
                    Store $\mathcal{D}_g^{\text{off}} \leftarrow (s^{\text{off}}, a^{\text{off}}, R^{\text{off}}, s_{\text{new}}^{\text{off}})$
                    $w_g^{p,\text{off}}, w_g^{T,\text{off}} =$
                          $\text{DQN}(\mathcal{D}_g^{\text{off}}, k, w_g^{p,\text{off}}, w_g^{T,\text{off}}, i_{\text{off}}, \bar{i}, \epsilon_g^{\text{off}}, \gamma_g^{\text{off}})$
                    $s^{\text{off}} \leftarrow s^{\text{off,new}}$
                end while
                Use $w_g^{p,\text{off}}$ to generate feedback,
                 i.e, Reward Signal $R^{\text{ns}} = \sum_{\tau_j^{\text{off}}=\tau_i^{\text{ns}}}^{\tau_i^{\text{ns}}+\Delta^{\text{ns}}-\Delta^{\text{off}}} R^{\text{off}}(\tau_j^{\text{off}})$
                Store $\mathcal{D}_g^{\text{ns}} \leftarrow (s^{\text{ns}}, a^{\text{ns}}, R^{\text{ns}}, s_{\text{new}}^{\text{ns}})$
                $w_g^{p,\text{ns}}, w_g^{T,\text{ns}} =$
                      $\text{DQN}(\mathcal{D}_g^{\text{ns}}, k, w_g^{p,\text{ns}}, w_g^{T,\text{ns}}, i_{\text{ns}}, \bar{i}, \epsilon_g^{\text{ns}}, \gamma_g^{\text{ns}})$
                $s^{\text{ns}} \leftarrow s^{\text{ns,new}}$
            end while
        end for
        Use $w_g^{p,\text{ns}}$ to generate feedback,
             i.e, Reward Signal $R^{\text{sp}} = \frac{1}{G} \sum_{g=1}^{G} \sum_{\tau_j^{\text{ns}}=\tau_i^{\text{sp}}}^{\tau_i^{\text{sp}}+\Delta^{\text{sp}}-\Delta^{\text{ns}}} R^{\text{ns}}(\tau_j^{\text{ns}})$
        Store $\mathcal{D}^{\text{sp}} \leftarrow (s^{\text{sp}}, a^{\text{sp}}, R^{\text{sp}}, s_{\text{new}}^{\text{sp}})$
        $w^{p,\text{sp}}, w^{T,\text{sp}} =$
              $\text{DQN}(\mathcal{D}^{\text{sp}}, k, w^{p,\text{sp}}, w^{T,\text{sp}}, i_{\text{sp}}, \bar{i}, \epsilon^{\text{sp}}, \gamma^{\text{sp}})$
        $s^{\text{sp}} \leftarrow s^{\text{sp,new}}$
    end while
end for
return $w^{p,\text{sp}}, \{w_g^{p,\text{ns}}, w_g^{p,\text{off}}, \forall g \in K\}$
```

The DQN Function, as described in Algorithm 2, represents the core training process utilized by the DQN networks. This process entails randomly selecting a batch of k samples from the memory buffer $\mathcal{D}$ (Line 2), determining the value of the loss function based on the network's performance (Line 3-4) using discount factors $(\gamma^{\text{sp}}, \gamma_g^{\text{ns}}, \gamma_g^{\text{off}})$, learning rates $(\epsilon^{\text{sp}}, \epsilon_g^{\text{ns}}, \epsilon_g^{\text{off}})$, and gradient descent updates (Line 5), and updating the target network state after a certain number of epochs, denoted as $\bar{i}$.

Algorithm 2 DQN Function

```
Input: $\mathcal{D}, k, w^p, w^T, i, \bar{i}, \epsilon, \gamma$
Output: $\{w^p, w^T\}$

1: function DQN($\mathcal{D}, k, w^p, w^T, i, \bar{i}, \epsilon, \gamma$)
2:     Select Random batch of of $k$ samples from $\mathcal{D}$
3:     Preprocess and pass the batch to $w^p$
4:     Find Loss between primary and Target Q values using (21)
5:     With gradient descent step update $w^p$
6:     Update $w^T$ if $rem(i, \bar{i}) = 0$
7: end function
8: return $\{w^p, w^T\}$
```

The computation complexity for the basic DQN architecture can be defined as $O(\mathcal{S} \cdot \mathcal{A} \cdot \mathcal{I})$ with $\mathcal{S}$, $\mathcal{A}$, being dimensions of state and action spaces and $\mathcal{I}$ being training epoch performed per iteration [1]. Therefore, for a considered multi-time scale approach, the computation complexity is given by $O(\mathcal{S}_g \cdot \mathcal{A}_g \cdot \mathcal{I}_g)$ with $\mathcal{S}_g = \mathcal{S}^{\text{sp}} \bigcup \mathcal{S}_g^{\text{ns}} \bigcup \mathcal{S}_g^{\text{off}}$ as a union of state spaces for the particular gth scenario. Similarly, $\mathcal{A}_g = \mathcal{A}^{\text{sp}} \bigcup \mathcal{A}_g^{\text{ns}} \bigcup \mathcal{A}_g^{\text{off}}$ is the dimension of action space and $\mathcal{I}_g = I_{\text{sp}} + I_{\text{ns}} + I_{\text{off}}$. It should be noted that with the definition of multiple vehicular scenarios, the overall state space for the network selection and offloading processes can be reduced significantly, and with that, through the paralization approach the complexity can be limited.

5 Numerical Results

The proposed DQN-based multi-time scale MDP methods are simulated over a Python-based simulator for analyzing the performance. In order to have a proper assessment of the performance of the proposed solution, we resorted to the definition of some benchmarks. Given the unique nature of the scenario being discussed, to the best of our understanding, there is no existing algorithm from the literature that can be deemed suitable for this situation. For this reason, we consider one static approach and three partial approaches, where, in turn, one of the three parameters to be optimized is randomly chosen. To this aim, the following benchmark solutions are considered for comparison purposes:

a) **Static Approach (SA):** In this case, the services are randomly placed and their placement is fixed over time. The network selection operation is based on a minimum distance approach, where VTs select the nearest EN to offload their complete task.

b) **MDP (Network Selection and Offloading) with Static Service Placements (MDP-SSP):** In this approach, the service placement is performed randomly and is fixed over time. On the other hand, the network selection and offloading decisions are made through MDP with different time scales. This approach allows us to measure the impact of dynamic service placements over time.

c) **MDP (Service Placement and Offloading) with Random Network Selection (MDP-RS):** In this approach, each VT randomly selects the ENs while service placement and offloading decisions are made using the MDP approach with multiple time scales. This allows us to evaluate the performance of network selection operations performed by VTs over time.

d) **MDP (Service Placement and Network Selection) with Full Offloading (MDP-FO):** In this approach, service placement and network selection operations are performed through the MDP model with different time scales. Each VT performs the full offloading towards the selected EN.

Simulation is performed considering an IoV scenario with varying numbers of VTs between 200 and 2000 and three edge layers (i.e., RSUs, UAVs, and HAP) are considered. We have considered $N = 50$ RSUs, $P = 30$ UAVs, and one HAP node to serve VTs with $\bar{S} = 5$ different services. Furthermore, $\hat{\mathcal{S}}_n^1 = 3, \forall n$, $\hat{\mathcal{S}}_p^2 = 2, \forall p$, and $\hat{\mathcal{S}}_h^3 = 5$ represent the limit on the number of services provided by different edge facilities. The generic mth VT generates a task request ρ_m

TABLE 2
Simulation parameters

HAP Coverage (R_h)	2 km
UAV Coverage $(R_{u,p})$	100 m
RSU Coverage $((R_{r,n}))$	50 m
VT Computation Cap. $(c_{v,m} \cdot f_{v,m})$	2 GFLOPS
RSU Computation Cap. $(\mathcal{L}_{r,n} \cdot c_{r,n} \cdot f_{r,n})$	10 GFLOPS
UAV Computation Cap. $(\mathcal{L}_{u,p} \cdot c_{u,p} \cdot f_{u,p})$	10 GFLOPS
HAP Computation Cap. $(\mathcal{L}_h \cdot c_h \cdot f_h)$	30 GFLOPS
HAP Altitude $(\bar{h}_h)$	10 km
UAV Altitude $(\bar{h}_{u,p})$	1 km
HAP Bandwidth (B_h)	250 MHz
UAV Bandwidth $(B_{u,p})$	75 MHz
RSU Bandwidth $(B_{r,n})$	25 MHz
VT Speed Range $(\vec{v}_{\text{min}}, \vec{v}_{\text{max}})$.	(8 m/s, 14 m/s)
VT Power $(P_{c,v_m}, Pt_{v_m}, Pr_{v_m})$	(1.1, 1.5, 1.3) W

with probability $p_a = 0.1$, having parameters $D_{\rho_m} = 5\,\text{MB}$, $D^r_{\rho_m} = 1\,\text{MB}$, $\Omega_{\rho_m} = 10^3 \cdot D_{\rho_m}$, and $T_{\rho_m} = 2\,\text{s}$. The service demand is based on the Zipf distribution with parameter $\beta = 0.8$. Also, the VT speed is defined as in (1), where $\mu = 10$ and $\sigma = 3$. Also, $\zeta = 0.1$, $\gamma_1 = 0.5$, and $\gamma_2 = 0.5$ are considered during the problem formulation. The vehicular density parameters, $M_1 = 500$ and $M_2 = 1200$, are considered in the scenario definition. For DQN simulation, primary and target networks with layers $\bar{K}^{\text{sp}} = 5$, $\bar{K}^{\text{ns}}_g = \bar{K}^{\text{off}}_g = 3$, are considered with learning parameters $e^{\text{sp}} = 0.7$, $e^{\text{ns}}_g = e^{\text{off}}_g = 0.65$, $\mathcal{D}^{\text{sp}} = 4000$, $\mathcal{D}^{\text{ns}}_g = \mathcal{D}^{\text{off}}_g = 2000$, $\epsilon^{\text{sp}} = \epsilon^{\text{ns}}_g = \epsilon^{\text{off}}_g = 0.05$, $\gamma^{\text{sp}} = \gamma^{\text{ns}}_g = \epsilon^{\text{off}}_g = 0.98$. In addition, the learning process includes $\bar{N} = 50$ with $I_{\text{sp}} = I_{\text{ns}} = I_{\text{off}} = 10^3$ and $\bar{\imath} = 50$. The other important system model parameters are provided in Table 2.

1) Joint Cost Analysis with Varying Vehicular Density: In this work, our aim is to minimize the joint cost function of latency, energy and additional penalty values for service placement. In Fig. 4, we present the average cost values for different approaches with varying numbers of VTs. The overall cost required for the SA approach is significantly higher compared to the other approaches, since all three decisions are made using a heuristic approach. On the other hand, the other three MDP approaches (MDP-FO/RS/SSP) optimize the decisions for two MDPs while using the static approach for the remaining one. However, such methods can reduce the overall cost requirements, resulting in suboptimal solutions. This highlights the importance of performing multi-time-scale optimization for service placement, network selection, and offloading processes together. Indeed, the cost required for the proposed MDP-MT approach improves overall performance.

2) Number of Handover Required: In addition to the general reduction of costs, the reliability of the solutions may be an important criterion to analyze the performance. With this, in Fig. 5, we present the overall handover requirements in terms of sojourn time constraint failure during the offloading process. The SA approach with static service placement, minimum distance-based network selection, and complete offloading requires a large number of handovers. Although the MDP-FO approach performs the dynamic service placement and network selection operations, with full offloading, it suffers with higher number of failures. Similarly, the MDP-RS approach suffers due to imperfect node selection through random allocations. The MDP-SSP approach can reduce the number of failures with proper network selection and offloading; however, due to static service placement, the process still suffers with handover demands. The proposed MDP-MT approach can have the potential to provide a reliable solution with joint optimization and can be a useful solution for vehicular scenarios with higher reliability requirements.

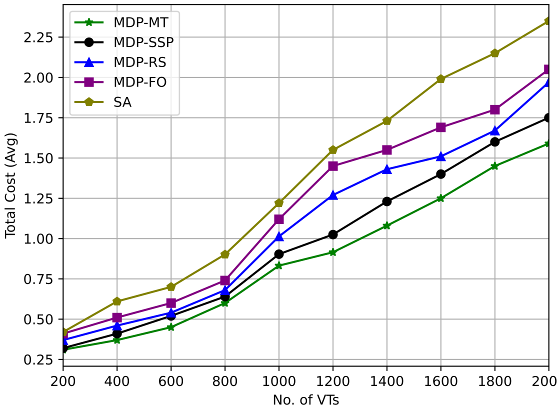

Fig. 4. Performance results in terms of overall cost function with variable number of active vehicles.

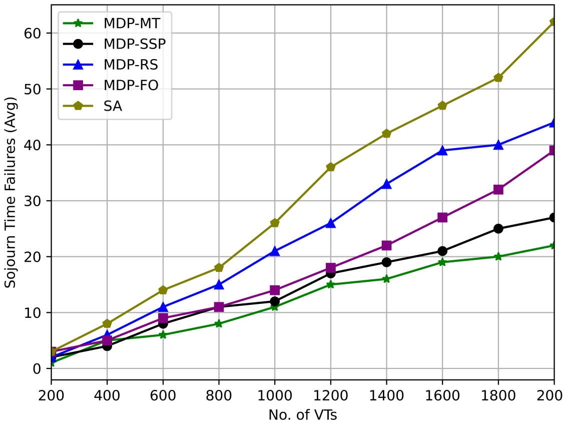

Fig. 5. Percentage of VUs with Handover Requirements.

3) Number of Failures in terms of Service Latency: Another way to measure the reliability of the proposed solutions is by assessing the demand for service latency. In Fig. 6, we present the amount of service latency constraint failures for different solutions. Similarly to the previous solutions, the SA approach with static decisions suffers from several failures. On the other hand, the proposed MDP-MT solutions can reduce service latency failures through dynamic service placements, proper network selections, and offloading decisions. The other MDP solutions can reduce the overall number of failures compared to the SA method; however, their performance is suboptimal. With reduced flexibility in the offloading process (i.e., full offloading in MDP-FO or random network selection in MDP-RA) both of these methods suffer with more failures compared to the MDP-MT solutions. Additionally, the MDP-SSP approach with static service placements has a higher number of failures. This highlights the importance of dynamic service

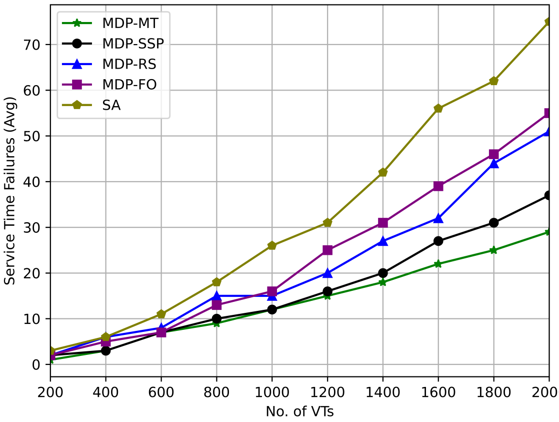

Fig. 6. Percentage of VUs with service time constraint violation.

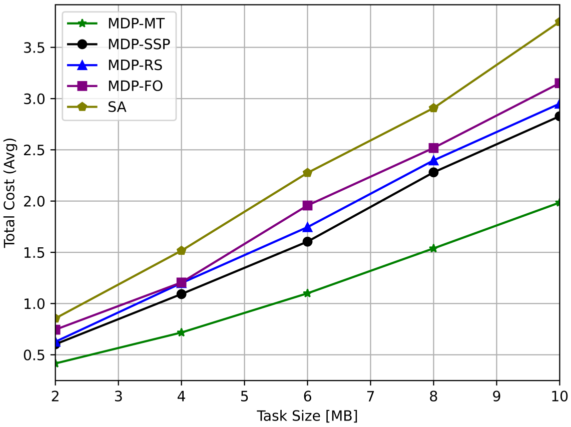

Fig. 7. Performance results in terms of overall cost function with variable task size.

placement in dynamic vehicular scenarios.

4) Joint Cost Analysis with Varying Task Size: In a dynamic vehicular environment, different parameters can vary over time, inducing additional complexities. VTs demand a proper solution with reliable performance under different vehicular conditions. In Figure 7, we have analyzed the combined cost values for various approaches with varying task sizes. The findings indicate that as the task size increases, the SA incurs significantly higher costs due to suboptimal service placements, network choices, and offloading strategies. In contrast, the three MDP solutions (MDP-FO/RS/SSP) demonstrate enhanced performance across a range of task sizes compared to SA. However, the performance is still limited, mainly due to the static policies adopted for one MDP solution. On the other hand, the proposed MDP-MT solution can have better performance compared to other solutions. Furthermore, the performance gap between other solutions and the MDP-MT approach improves with time, highlighting the reliability of proposed solutions over increasing values of computation tasks.

TABLE 3
Average Percentage of Data Offloading.

# VNs	200	400	600	800	1000	1200	1400	1600	1800	2000
MDP-MT	.76	.74	.73	.68	.66	.63	.59	.55	.48	.47
MDP-SSP	.78	.79	.78	.73	.72	.67	.65	.62	.58	.56
MDP-RS	.65	.59	.62	.56	.51	.44	.39	.35	.32	.35
MDP-FO	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

In the considered multi-time-scale approach it is important to analyze the performance of the proposed solutions for the decisions made on different time scales. In the following, we analyze the performance of the computation offloading, network selection, and service placement decisions made by MDP solutions on different time scales.

5a) Performance in terms of Offloading Percentage: The computation offloading decision where the amount of data to be offloaded towards a selected EN is determined is based on the state action data received from the MDPs of the upper layers and the parameters of the local environment. Additionally, these decisions are implemented on a faster time scale compared to the other two decisions. It can be challenging to select the appropriate amount of data to be offloaded to EN, especially in dynamic vehicular environments. On the other hand, improper offloading policies can induce higher latency and energy costs.

Table 3 presents the average percentage amount of data offloaded by VTs towards the EC facilities. Different MDPs can have different performance based on their decision-making strategies. MDP-FO adapts the full offloading strategy resulting in 100% offloading with reduced flexibility, while MDP-RS uses the random node selection strategy, resulting in suboptimal offloading decisions. Indeed, the overall percentage of offloading is significantly reduced for the MDP-RS process, adding an additional burden on local computing resources. Similarly, the MDP-SSP approach characterized by static service placement can suffer from suboptimal offloading decisions with reduced flexibility in terms of node selection and offloading. The proposed MDP-MT, with a hierarchical decision-making process, can adapt to changing vehicular densities through dynamic service placement, appropriate node, and adequate offloading parameter selections.

5b) Performance in terms of EN Selection: Given the existence of several EC layers comprising diverse nodes and a tailored multi-time-scale strategy, it is crucial to evaluate the efficiency concerning the utilization of edge resources across various vehicular densities. For this, in Fig. 8, we present the simulation results in terms of percentage number of VTs selecting different edge layers with varying numbers of VTs. At first, with a lower number of VTs on the road, VTs exploit RSUs and UAV resources to perform their tasks. With increasing vehicular density, the overall percentage of VTs selecting RSUs decreases, while the number of VTs exploiting HAP resources increases. It should be noted that these results can be strongly influenced by the underneath system elements. For example, since we have considered a single HAP node the overall percentage of VTs exploiting the HAP resources is always low, while RSUs with dense deployments can serve a large portion of VTs. Such trends can be explained further with other deployment options.

5c) Performance in terms of Service Placements: In Fig. 9, the performance of the service placement strategies under different levels of vehicular densities is illustrated. Performance is evaluated based on the average occurrence of service failures, which signifies situations where VTs are

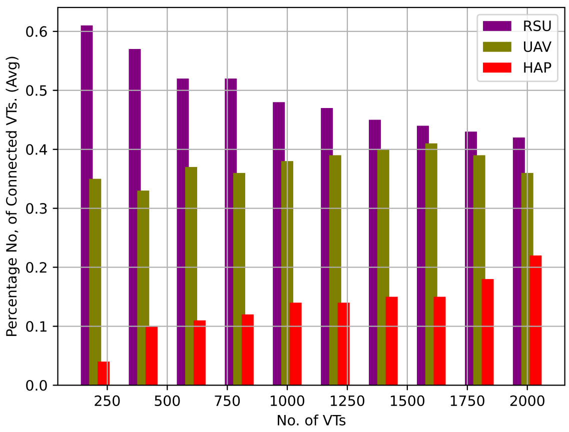

Fig. 8. Avg. No. of EN Selected for Offloading.

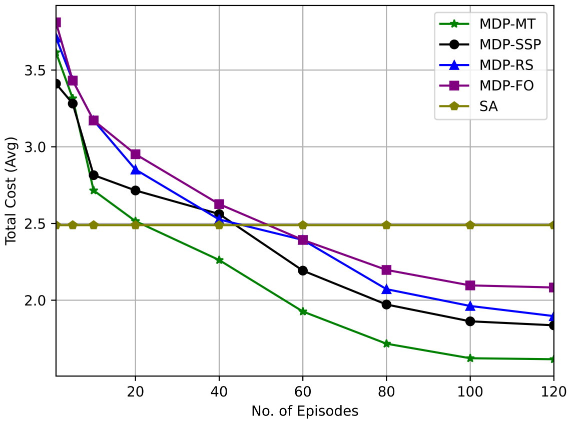

Fig. 10. Avg. Cost Vs Training Episodes.

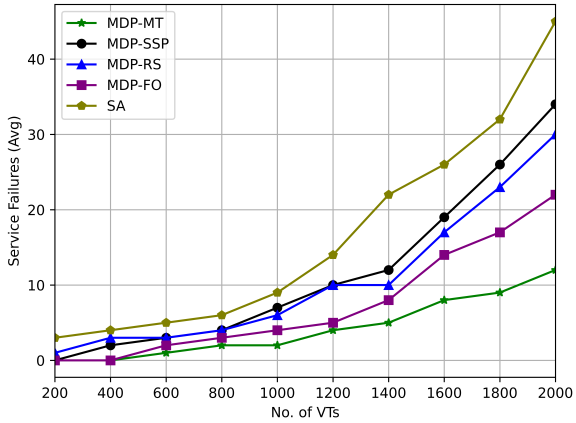

Fig. 9. Avg. No. of Service Failures.

unable to access the necessary service from a chosen EN. The decisions related to service placement and network selection underneath can influence the average number of service failures observed. As illustrated in Figure 9, the MDP-MT solutions suggested, along with appropriate service placement and network selection strategies, can result in reduced failures. In contrast, the use of static policies, such as SA solutions, may lead to significantly increased failures. Moreover, MDP solutions with inadequate decision-making may experience more service failures in comparison to MDP-MT. It is worth mentioning that the MDP-FO solution, when combined with appropriate network selection and service placement in a static offloading policy, may result in more failures compared to MDP-MT. This is primarily because the upper-level MDPs receive suboptimal feedback from lower-level offloading agents following a static policy. This underscores the significance of collectively optimizing all three decision-making processes across various time scales to enhance performance.

6) Training Performance over Learning Episodes: In Figure 10, the RL approach training process is depicted in various training episodes. Specifically, training is evaluated based on the average cost resulting from service placements, network selection, and offloading choices. In addition, performance is compared with the SA solution to assess the convergence of the RL methods. Over time, i.e., learning episodes, the effectiveness of MDP solutions tends to improve, potentially resulting in lower costs. Specifically, MDP-MT solutions, when coupled with appropriate feedback from MDP agents, can effectively decrease cost values. In contrast, the three MDP solutions (MDP-FO/RS/SSP) with suboptimal feedback can gradually improve performance in comparison to the SA solutions. Nevertheless, they result in increased expenses in contrast to the MDP-MT alternatives primarily because of the fixed selection made by one of the MDP agents. It is important to note that the training procedure can also be influenced by the specific hardware utilized during the training phase. The suggested approaches rely on a standard computer system that features the Intel Core i5 processor. The efficacy can be further evaluated on powerful processing units, such as Tensor Processing Units, which is not within the scope of this study.

6 Conclusion

In this work, we have proposed a multi-time-scale MDP approach to solve the problem of joint service placement, network selection, and computation offloading over dynamic vehicular scenarios enabled by multiple EC platforms. The proposed approach can model the decision-making process on different time scales based on the network and user requirements. The optimization problem is formed to jointly minimize latency, energy and service placement costs in different vehicular scenarios. Advanced DQN-based solutions are considered to solve complex MDP in finding optimal policies that reduce overall cost with improved reliability. The numerical results acquired through a Python-based simulator show several advantages over traditional benchmark solutions. With the complex nature of a considered joint optimization process and the proposed multi-time scale MDP, to avoid excessive discussions, in this work, we have resorted to the basic DQN approach. In recent years, several new advanced forms of DRL algorithms have been proposed with additional benefits. In the future, our aim is to extend the proposed framework to accommodate such advanced DRL solutions with improved efficiency and reduced training costs.

References

[1] S. S. Shinde and D. Tarchi, "Collaborative reinforcement learning for multi-service Internet of Vehicles," *IEEE Internet Things J.*, vol. 10, no. 3, pp. 2589–2602, Feb. 2023.

[2] D. Han, Q. Ye, H. Peng, W. Wu, H. Wu, W. Liao, and X. Shen, "Two-timescale learning-based task offloading for remote IoT in integrated satellite-terrestrial networks," *IEEE Internet Things J.*, vol. 10, no. 12, pp. 10 131–10 145, Jun. 2023.

[3] S. S. Shinde and D. Tarchi, "Joint air-ground distributed federated learning for intelligent transportation systems," *IEEE Trans. Intell. Transp. Syst.*, 2023, early access, doi:10.1109/TITS.2023.3265416.

[4] ——, "Network selection and computation offloading in non-terrestrial network edge computing environments for vehicular applications," in *2022 11th Advanced Satellite Multimedia Systems Conference and the 17th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, Graz, Austria, Sep. 2022, pp. 1–8.

[5] Y. He, Y. Wang, Q. Lin, and J. Li, "Meta-hierarchical reinforcement learning (MHRL)-based dynamic resource allocation for dynamic vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 4, pp. 3495–3506, Apr. 2022.

[6] A. Bozorgchenani, D. Tarchi, and W. Cerroni, "On-demand service deployment strategies for Fog-as-a-Service scenarios," *IEEE Commun. Lett.*, vol. 25, no. 5, pp. 1500–1504, May 2021.

[7] S. S. Shinde and D. Tarchi, "A markov decision process solution for energy-saving network selection and computation offloading in vehicular networks," *IEEE Trans. Veh. Technol.*, 2023, early access, doi:10.1109/TVT.2023.3264504.

[8] 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.*, vol. 21, no. 12, pp. 4233–4248, Dec. 2022.

[9] 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.

[10] F. Tang, B. Mao, N. Kato, and G. Gui, "Comprehensive survey on machine learning in vehicular network: Technology, applications and challenges," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 2027–2057, Third Quarter 2021.

[11] K. Sharma, B. Butler, and B. Jennings, "Scaling and placing distributed services on vehicle clusters in urban environments," *IEEE Trans. Serv. Comput.*, vol. 16, no. 2, pp. 1402–1416, Mar.-Apr. 2023.

[12] A. Talpur and M. Gurusamy, "DRLD-SP: A deep-reinforcement-learning-based dynamic service placement in edge-enabled Internet of Vehicles," *IEEE Internet Things J.*, vol. 9, no. 8, pp. 6239–6251, Apr. 2022.

[13] H. Sami, R. Saado, A. E. Saoudi, A. Mourad, H. Otrok, and J. Bentahar, "Opportunistic UAV deployment for intelligent on-demand IoV service management," *IEEE Trans. Netw. Service Manag.*, 2023, early access, doi:10.1109/TNSM.2023.3242205.

[14] X. Gao, R. Liu, A. Kaushik, and H. Zhang, "Dynamic resource allocation for virtual network function placement in satellite edge clouds," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 4, pp. 2252–2265, 2022.

[15] X. Gao, R. Liu, and A. Kaushik, "Virtual network function placement in satellite edge computing with a potential game approach," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 2, pp. 1243–1259, 2022.

[16] L. Liu, M. Zhao, M. Yu, M. A. Jan, D. Lan, and A. Taherkordi, "Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 2169–2182, Feb. 2023.

[17] 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 Things J.*, vol. 8, no. 12, pp. 9763–9773, Jun. 2021.

[18] K. Fan, B. Feng, X. Zhang, and Q. Zhang, "Network selection based on evolutionary game and deep reinforcement learning in space-air-ground integrated network," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 3, pp. 1802–1812, 2022.

[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.*, vol. 71, no. 2, pp. 2041–2057, Feb. 2022.

[20] X.-Q. Pham, T. Huynh-The, E.-N. Huh, and D.-S. Kim, "Partial computation offloading in parked vehicle-assisted multi-access edge computing: A game-theoretic approach," *IEEE Trans. Veh. Technol.*, vol. 71, no. 9, pp. 10 220–10 225, Sep. 2022.

[21] J. Shi, J. Du, J. Wang, and J. Yuan, "Deep reinforcement learning-based V2V partial computation offloading in vehicular fog computing," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, Nanjing, China, Mar.-Apr. 2021.

[22] Z. Jia, Q. Wu, C. Dong, C. Yuen, and Z. Han, "Hierarchical aerial computing for internet of things via cooperation of HAPs and UAVs," *IEEE Internet Things J.*, vol. 10, no. 7, pp. 5676–5688, 2023.

[23] Q. Shen, B.-J. Hu, and E. Xia, "Dependency-aware task offloading and service caching in vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 71, no. 12, pp. 13 182–13 197, Dec. 2022.

[24] Y. Zheng, T. Zhang, and J. Loo, "Dynamic multi-time scale user admission and resource allocation for semantic extraction in MEC systems," *IEEE Trans. Veh. Technol.*, vol. 72, no. 12, pp. 16 441–16 453, 2023.

[25] L. Ma, H. Liu, L. Zhou, C. Yang, W. Dai, and G. Wang, "Security control for multi-time-scale CPSs under DoS attacks: An improved dynamic event-triggered mechanism," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 3, pp. 1813–1826, 2022.

[26] Y. Cao, S.-Y. Lien, Y.-C. Liang, D. Niyato, and X. Shen, "Collaborative computing in non-terrestrial networks: A multi-time-scale deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, 2023, early Access. DOI:10.1109/TWC.2023.3323554.

[27] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10 190–10 203, Nov. 2018.

[28] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.

[29] X. Gu and G. Zhang, "Energy-efficient computation offloading for vehicular edge computing networks," *Computer Communications*, vol. 166, pp. 244–253, Jan. 2021.

[30] S. Mao, S. He, and J. Wu, "Joint UAV position optimization and resource scheduling in space-air-ground integrated networks with mixed cloud-edge computing," *IEEE Syst. J.*, vol. 15, no. 3, pp. 3992–4002, Sep. 2021.

[31] H. S. Chang, P. Fard, S. Marcus, and M. Shayman, "Multitime scale markov decision processes," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 976–987, Jun. 2003.

Swapnil Sadashiv Shinde (Student Member, IEEE) received the Ph.D. degree in Automotive Engineering for Intelligent Mobility at the University of Bologna, Italy, in 2024, working on connected vehicles for beyond 5G scenarios.

From 2015 to 2017, he worked as a Project Engineer at the Indian Institute of Technology, Kanpur, India. Since November 2023, he is a Researcher with the Consorzio Nazionale Interuniversitario delle Telecomunicazioni (CNIT), Italy. His research interests include edge computing, reinforcement learning, distributed machine learning, and non-terrestrial networks.

Daniele Tarchi (Senior Member, IEEE) received the Ph.D. degree in Informatics and Telecommunications Engineering from the University of Florence, Florence, Italy, in 2004.

Since 2019 he is an Associate Professor at the University of Bologna, Italy. He is the author of more than 160 published articles in international journals and conference proceedings. His research interests are mainly on Wireless Communications and Networks, Edge Computing, Distributed Learning, and Optimization Techniques.

Prof. Tarchi has been an IEEE Senior Member since 2012. He is an Editorial Board member for IEEE Transactions on Vehicular Technology, IEEE Open Journal of the Communication Society and IET Communications.