Proceedings Paper:
Shinde, S.S., Naseh, D. and Tarchi, D. (2024) In-Space Computation Offloading for Multi-layer LEO Constellations. In: European Wireless 2023. 28th European Wireless 2023, 02-04 Oct 2023, Rome, Italy. , pp. 75-82. ISBN: 978-3-8007-6225-5.

In-Space Computation Offloading for Multi-layer LEO Constellations

Swapnil Sadashiv Shinde, David Naseh, and Daniele Tarchi
Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi"
University of Bologna
Bologna, Italy
email: {swapnil.shinde2, david.naseh2, daniele.tarchi}@unibo.it

***Abstract*—Non-terrestrial Networks (NTN) will play a key role in enabling a fully connected, digitized, and intelligent society through their integration into traditional terrestrial communication networks in the upcoming 6G world. The space-based Orbital Edge Computing (OEC) has already shown some promising results in terms of boosting the capacity, coverage, security, and resilience of communication systems. Among others, Low Earth Orbit (LEO) satellite constellations can be beneficial in terms of reduced communication distances and easy and flexible deployments. With integrated OEC facilities into NTN platforms, a new era of space computing has begun. It has several benefits in terms of serving space/ground users through satellite computation, communication, and storage resources. However, with size limitations, each LEO satellite node can have a limited amount of resources. This restricts the number of services provided by the satellites and the users served. Therefore a proper user-satellite assignment is needed. Additionally, with a boost in communication technologies, different terrestrial and NTN layers can form multi-tier computing facilities with higher capacity and coverage. We aim to solve the network selection problem over multi-service multi-tier edge computing facilities provided by multiple LEO constellations and cloud computing facilities. A Hierarchical Reinforcement Learning (HRL) based solution is proposed for optimizing the multi-level network selection decisions aiming at minimizing the task processing latency. The simulation results show improvements in terms of latency performance and service reliability, considering user/system constraint satisfaction.**

***Index Terms*—Non-terrestrial Networks, Orbital Edge Computing, Hierarchical Reinforcement Learning, LEO Satellites**

I. Introduction

THE upcoming 6G technology is expected to create a fully connected intelligent society with the help of several innovative technologies such as the Internet of Things (IoT), Machine Learning (ML), network softwarization, and multi-access Edge Computing (MEC) [1]. Non-terrestrial Networks (NTN) are considered an important element for boosting the coverage and capacity of traditional terrestrial systems through their integration into the 6G networks thus satisfying the global coverage demands [2]. On the other hand, MEC technology is useful for enabling novel services and applications with stringent data-processing and latency demands by bringing the networking resources in the proximity of end-users [3]. The MEC-integrated NTN platforms, i.e., orbital edge computing (OEC), can enable computing in space. OEC-enabled LEO satellite networks can be extremely useful to serve ground users located in remote areas with limited or zero connectivity to terrestrial networks.

In recent times, several new LEO constellations such as Starlink, OneWeb, Telesat, etc, tailored to specific missions have populated the space [4]. These constellations can be located at different altitudes, having varying satellite densities, interplane and intra-plane satellite distances, speeds, etc. Thus, ground users can have access to multiple LEO satellites belonging to different constellations for OEC resources. Additionally, satellites can also act as relay nodes to route user requests to other nearby satellites or terrestrial cloud computing facilities. Given the size restrictions and limited coverage, satellites can have reduced OEC resources and thus can serve a limited number of users only. Additionally, with the limited storage resources, each LEO node can be able to store a limited number of services. In a multi-service scenario, users can demand different services based on their specific requirements [5]. This opens a new challenge of proper user-server assignments based upon the users' demands and the availability of the resources at servers in the space with multi-tier edge computing facilities.

In this work, we aim to solve the network selection problem over such multi-service multi-tier edge computing environments in the space hosted at different LEO satellite constellations and ground-based cloud facilities. Solving the network selection problem over such dynamic, and heterogeneous edge computing systems can be challenging. However, it can benefit in terms of improved performance, service reliability, and optimal network resource uses. Given the complex nature of the considered problem in a dynamic environment, traditional optimization methods can have limited performance. In recent times, several ML methods, including Reinforcement Learning (RL), have gained huge popularity to solve such complex networking problems [3]. Different types of RL frameworks, e.g., Hierarchical RL (HRL) [6], multi-agent RL [7], collaborative RL [3] and federated RL [8] are available for solving problems based upon their demands. The network selection problem over an edge computing environment can be modeled as a sequential decision-making process through the adaptation of a proper Markov Decision Process (MDP)

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")

model. Additionally given the complex nature of a problem with multilevel decision-making, the HRL-based techniques with advanced deep learning-based solutions can be adequate for finding the optimal policies. Therefore, for the considered multi-tier edge computing facilities we propose HRL-based solutions for optimizing the overall task processing latency through optimized network selection decisions at different levels. A deep Q-learning approach is used to find the optimal network selection policies.

II. System Model and Problem Formulation

We consider a set of multi-layer LEO satellite constellations $C = \{C_1, \cdots, C_i, \cdots, C_{\bar{C}}\}$ and a cloud facility C_0 able to provide edge computing services to the ground-based user set $\mathcal{U} = \{u_1, \cdots, u_j, \cdots, u_{\bar{U}}\}$. Each ith constellation includes a set of LEO satellites $\bar{L}^i$ distributed in the multiple planes for providing global coverage. Also, each ith constellation is characterized by the intra-plane distance d^i_{in} and inter-plane distance $d^i_{out} \in \{d^{i,min}_{out}, d^{i,max}_{out}\}$ with $d^{i,min}_{out}$ and $d^{i,max}_{out}$ denoting the minimum and maximum inter-plane distances over which satellites can communicate with a single hop communication.

The system is modeled in a time-discrete manner, and the network parameters are supposed to be constant over each time interval τ, where τ_m identifies the mth time interval, i.e., $\tau_m = \{\forall t | t \in [m\tau, (m+1)\tau]\}$. The LEO constellations are located at incremental heights $h_1 < h_2 < \cdots h_{\bar{C}-1} < h_{\bar{C}}$ from the earth's surface, where the set of satellites in ith constellation is $\mathcal{L}^i = \{l^i_1, \cdots, l^i_k, \cdots, l^i_{\bar{L}_i}\}$. Each kth satellite from ith constellation is also characterized by the location $(x^i_k(\tau_m), y^i_k(\tau_m), z^i_k(\tau_m))$ coverage radius r^i_k, and speed v^k_i. Additionally, it also contains a processing capacity equal to c^i_k Floating Point Operations per Second (FLOPS) per CPU cycle, while its CPU frequency is f^i_k. Each satellite is supposed to be able to communicate on a bandwidth B^i_k to the users and other satellites in the vicinity. Also, with their limited storage resources, satellites can host a set of services $\mathcal{S}^k_i \subseteq \mathcal{S}$ where $\mathcal{S} = \{s_1, \cdots, s_p, \cdots, s_{\bar{S}}\}$ is the complete set of services provided by the network operators.

The generic jth user is characterized by a processing capacity equal to c_j FLOPs per CPU cycle, while its CPU frequency is f_j. Each user is supposed to be able to communicate on a bandwidth $B^i_{jk}(\tau_m)$ with the kth LEO satellite belonging to the ith constellation, provided that the coverage condition is satisfied. Also, user u_j is supposed to be located at (x_j, y_j) and covered by one LEO satellite from each constellation. Each jth user is supposed to be active in each time interval with a probability p_a within which it generates a computation task request $\rho_j(\tau_p)$ identified through the tuple $\left\langle D_{\rho_j}, D^r_{\rho_j}, \Omega_{\rho_j}, T_{\rho_j}, s_{\rho_j} \right\rangle$ corresponding to a task with size D_{ρ_j} Byte, expected to give in output a result with size $D^r_{\rho_j}$ Byte, requesting Ω_{ρ_j} CPU execution cycles and a maximum execution latency T_{ρ_j}. Here, $s_{\rho_j} \in \mathcal{S}$ corresponds to a specific service requested by user u_j that belongs to a set of services $\mathcal{S}$ provided by the network service provider.

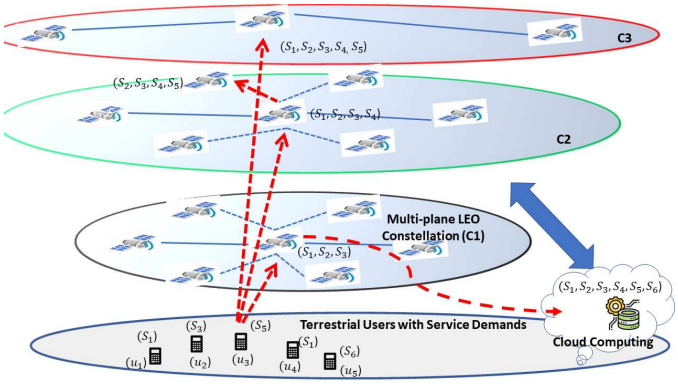

Fig. 1. Multiservice Multi plane LEO Constellations with Cloud Connectivity

Figure 1 shows a possible 3-tier joint Terrestrial and Non-terrestrial Networking (T/NTN) scenario with one user layer, three LEO satellite constellations, and one cloud computing facility. A set of 6 services are distributed over the edge/cloud facilities based on their storage capabilities. The cloud computing facilities, given their superior nature can provide all 6 services, while the LEO satellites with size limitations are able to provide a subset of services. Users are generating a specific service request and through their connection towards NTN and cloud facilities are able to explore edge/cloud computing resources. In the example, user u_3 is demanding service s_5 from edge/cloud facilities. The user demand can be satisfied in different ways. In the first case, the satellite node from C_1, covering the u_3 is unable to provide the demanded service. However, it can relay the user request to the cloud facilities. In the second case, a LEO satellite from C_2 is also unable to provide the demanded service. However, another node from its vicinity is having the demanded service, and thus user requests can be relayed to it. In the third case, the u_3 can connect directly to the satellite from C_3, having a pre-installed service. In each of these scenarios, the user can experience different transmission/computation costs. Additionally, with the presence of multiple users demanding different services, user-server allocation decisions need to be optimized for efficient resource utilization and limiting the overall costs. In the following, we provide a satellite coverage model and a detailed cost analysis for the network selection problem.

A. LEO Satellites Mobility Model

Terrestrial users can communicate with satellite nodes on specific time intervals, based on the locations and the mobility patterns of LEO satellites. LEO satellites move at a very high speed (i.e., a few km per second) compared with ground users. LEO satellite k from constellation i is located at height h_i with respect to the ground, and moves with a constant speed v^k_i km/s. At a certain instant τ_m, we define the elevation angle of the considered LEO satellite with respect to the jth user as $\theta_j(\tau_m)$, corresponding to a geocentric angle for the jth user [5], as:

$$\delta^i_{k,j}(\tau_m) = \arccos\left(\frac{R_e}{R_e + h_i} \cdot \cos\left(\theta_j(\tau_m)\right)\right) - \theta_j(\tau_m)$$

where R_e is the Earth radius. The total arc length over which the user can communicate with the considered LEO satellite is defined as:

$$L^i_{k,j}(\tau_m) = 2(R_e + h_i) \cdot \delta^i_{k,j}(\tau_m)$$

Finally, the total time for which the jth user can be in the coverage range of the considered LEO satellite is given as,

$$T^{i,soj}_{j,k}(\tau_m) = \frac{L^i_{k,j}(\tau_m)}{v^i_k} \tag{1}$$

The jth user should complete the offloading process towards the LEO satellite during the coverage time, for avoiding the additional costs in terms of LEO satellite handovers.

B. Network Selection Problem for User Task Processing

In the considered system model, each user is under the coverage of multiple LEO constellations having a set of satellites organized in the different orbital planes for global coverage. Each user can connect directly to one LEO satellite, called coverage satellite, from each constellation at each time interval. Thus, each user can have a direct connection to $\bar{C}$ satellites[1]. Additionally, satellites can relay the user processing requests to nearby satellites from the same constellations. If required, it can also act as a relay node between the user and a terrestrial cloud computing facility with abundant computing and storage resources. Thus, users can have multiple options to process the task data. They can process locally, at the coverage satellite, exploiting other neighboring satellites or cloud facilities through multi-hop communication links. Each of these decisions can impact the overall task processing cost and thus optimization is needed for proper edge node selections.

In the considered network selection problem, multi-level decision-making process is required. First, ground users aim to select a proper coverage satellite for offloading their data. In the next level, the selected satellite node decides whether to compute the task by itself, offload it to the nearby satellites, or relay it to cloud facilities. Based upon the nature of these decisions, hereafter, we call them higher/abstract level (user-coverage satellite selection) and lower/finer level (computation node selection) decisions. Here, we introduce a binary variable $a^{l^i_k}_{u_j}(\tau_m) = \{0, 1\}$ defining the jth users coverage node selection decision. It takes 1 if the jth user selects the coverage satellite k from constellation i for processing its tasks at the mth interval, while 0 indicates that the user prefers to process the task by itself. Another variable, $b^{\hat{k}}_{u_j,l^i_k}(\tau_m + \epsilon) = \{0, 1\}$, is considered for measuring the computation node selection decision of the kth satellite selected by the jth user. In this ϵ is a negligible time interval between the two decision levels (i.e., the decision made by user j for selecting the coverage satellite at a higher level and the next lower level decision made by the selected coverage satellite node for finding the suitable computation node). Here, 1 indicates that the kth satellite has decided to offload the users' task to the nearby edge computing facilities $\hat{k}$, while 0 indicates the satellite's decision for processing its task locally. Note that here $\hat{k}$ can be another nearby satellite from the same ith constellation $l^i_{\hat{k}} \in \mathcal{N}^i_l$ with $\mathcal{N}^i_l$ being a set of nearby satellites that kth coverage node can explore or the cloud computing facility C_0. Based on this here we define two decision matrices, $\mathcal{A} = \{a^{l^i_k}_{u_j}(\tau_m)\}$ and $\mathcal{B} = \{b^{\hat{k}}_{u_j,l^i_k}(\tau_m + \epsilon)\}$, indicating the network selection decisions.

To avoid additional complexity we consider that each user can be assigned to only one coverage satellite and each coverage satellite, if needed, can select at most one nearby satellite or cloud facility for relaying the user data. Thus,

$$\begin{aligned} &\sum_{i=1}^{\bar{C}} \sum_{k=1}^{\bar{L}_i} a^{l^i_k}_{u_j}(\tau_m) \leq 1, \quad \forall j \\ &\sum_{\hat{k}=0}^{|\mathcal{N}^i_l|} b^{\hat{k}}_{u_j,l^i_k}(\tau_m + \epsilon) \leq 1, \quad \forall j, k, i \end{aligned} \tag{2}$$

where $\hat{k} = 0$ indicates the selection of a cloud computing facility for task processing.

The number of users requesting computation services from the kth satellite node of ith constellation are:

$$\hat{K}^{i,comp}_k(\tau_m) = \sum_{j=1}^{\bar{U}} \left[\left(a^{l^i_k}_{u_j}(\tau_m) - b^{C_0}_{u_j,l^i_k}(\tau_m + \epsilon) - \sum_{\hat{k}=1}^{|\mathcal{N}^i_l|} b^{\hat{k}}_{u_j,l^i_k}(\tau_m + \epsilon) \right) + \sum_{k'=1}^{\bar{L}_i} b^{k}_{u_j,l^i_{k'}}(\tau_m + \epsilon) \right], \forall k, i$$

The number of users requesting communication services from the kth satellite node of ith constellation are:

$$\hat{K}^{i,comm}_k(\tau_m) = \sum_{j=1}^{\bar{U}} \left[a^{l^i_k}_{u_j}(\tau_m) + \sum_{k'=1}^{\bar{L}_i} b^{k}_{u_j,l^i_{k'}}(\tau_m + \epsilon) \right], \forall k, i$$

With multiple users requesting services, during the offloading process, the following constraints need to be taken into account for each kth LEO satellite form ith constellation:

$$\begin{cases} \sum_{j=1}^{\hat{K}^{i,comp}_k(\tau_m)} c^i_{j,k}(\tau_m) \cdot f^i_{j,k}(\tau_m) \leq (c^i_k \cdot f^i_k) & \text{(3a)} \\ \sum_{j=1}^{\hat{K}^{i,comm}_k(\tau_m)} b^i_{j,k}(\tau_m) \leq B^i_k & \text{(3b)} \end{cases}$$

where, $c^i_{j,k}(\tau_m) \cdot f^i_{j,k}(\tau_m)$ is the processing resources assigned by the kth satellite from ith constellation for processing the jth users task. Also, $b^i_{j,k}(\tau_m)$ are the bandwidth resources assigned to the user j by selected satellite k of constellation i. We consider that the EN resources are shared equally among the requesting users/satellites.

[1]Each user can have a direct connection to more than one satellite per constellation at a time, however, we consider that at least one per each constellation is present.

1) Task Computation Model: The generic expression for the time spent for the ρ_jth task computation on a nth device is given by

$$T_{c,n}^{\rho_j} = \frac{\Omega_{\rho_j}}{c_n^{\rho_j} f_n^{\rho_j}},$$

where $c_n^{\rho_j}$ and $f_n^{\rho_j}$ are the number of FLOPS, and CPU-frequency per CPU-cycle assigned to the jth user, respectively, whether n identifies a user (u_j), an LEO (l_k^i) or a cloud computing facility (C_0) [3].

2) Task Communication Model: For the case of a computation offloading, the transmission time between a generic node $\hat{j}$ and a generic node $\hat{k}$ for task $\rho_{\hat{j}}$ is given by[2]

$$T_{tx,\hat{j}\hat{k}}^{\rho_{\hat{j}}}(\tau_m) = \frac{D_{\rho_{\hat{j}}}}{r_{\hat{j}\hat{k}}(\tau_m)},$$

where $r_{\hat{j}\hat{k}}(\tau_m)$ is data-rate of the link between the two nodes. Similarly, the reception time at the $\hat{j}$th node to receive the task of size $D_{\rho_j}^r$ from $\hat{k}$th EN is

$$T_{rx,\hat{k}\hat{j}}^{\rho_{\hat{j}}}(\tau_m) = \frac{D_{\rho_{\hat{j}}}^r}{r_{\hat{k}\hat{j}}(\tau_m)}.$$

A symmetric channel is considered between $\hat{j}$ and $\hat{k}$.

In this work, for modeling the characteristics of the channel between the $\hat{j}$th and the $\hat{k}$th node at mth interval, we consider that the link gain can be modeled as $h_{\hat{j}\hat{k}}(\tau_m) = \beta_0 \cdot d_{\hat{j}\hat{k}}^{\theta^{\hat{k}}}(\tau_m)$, where $d_{\hat{j}\hat{k}}(\tau_m)$ is the distance between node $\hat{j}$ and $\hat{k}$ at mth interval, β_0 is the channel power gain at 1 m reference distance, while $\theta^{\hat{k}}$ is the path loss coefficient for the the communication link between node $\hat{j}$ and $\hat{k}$ [5]. The expression for the channel transmission rate is based on the Shannon capacity formula.

The communication time associated with the jth users task is given by,

$$\begin{aligned} T_{\rho_j}^{comm}(a_{u_j}^{l_k^i}(\tau_m), \mathbf{b}_{u_j,l_k^i}(\tau_m+\epsilon)) = \\ a_{u_j}^{l_k^i}(\tau_m)\left(T_{tx,jk}^{\rho_j}(\tau_m) + T_{rx,jk}^{\rho_j}(\tau_m)\right) \\ + \sum_{\hat{k}\in\mathcal{N}_l^i\cup C_0} b_{u_j,l_k^i}^{\hat{k}}(\tau_m+\epsilon)\left(T_{tx,k\hat{k}}^{\rho_j}(\tau_m+\epsilon) + T_{rx,k\hat{k}}^{\rho_j}(\tau_m+\epsilon)\right) \end{aligned} \tag{4}$$

with $\mathbf{b}_{u_j,l_k^i}(\tau_m+\epsilon) = \{b_{u_j,l_k^i}^{\hat{k}}(\tau_m+\epsilon)\forall\hat{k} \in \mathcal{N}_l^i \cup C_0\}$. Similarly, the computation time required is,

$$\begin{aligned} T_{\rho_j}^{comp}(a_{u_j}^{l_k^i}(\tau_m), \mathbf{b}_{u_j,l_k^i}(\tau_m+\epsilon)) = \\ \left(a_{u_j}^{l_k^i}(\tau_m) - \sum_{\hat{k}\in\mathcal{N}_l^i\cup C_0} b_{u_j,l_k^i}^{\hat{k}}(\tau_m+\epsilon)\right) T_{c,k}^{\rho_j}(\tau_m) \\ + \sum_{\hat{k}\in\mathcal{N}_l^i\cup C_0} b_{u_j,l_k^i}^{\hat{k}}(\tau_m+\epsilon) T_{c,\hat{k}}^{\rho_j}(\tau_m) \end{aligned} \tag{5}$$

[2]In the following we identify with $\hat{j}$ and $\hat{k}$ the indexes of any generic node. Hence, j and k can have any index among u_j, l_k^i, and C_0.

From (4), (5), the total latency required to process the jth users task is given by,

$$\begin{aligned} T_{\rho_j}^{p}(a_{u_j}^{l_k^i}(\tau_m), \mathbf{b}_{u_j,l_k^i}(\tau_m+\epsilon)) = \\ T_{\rho_j}^{comp}(a_{u_j}^{l_k^i}(\tau_m), \mathbf{b}_{u_j,l_k^i}(\tau_m+\epsilon)) \\ + T_{\rho_j}^{comm}(a_{u_j}^{l_k^i}(\tau_m), \mathbf{b}_{u_j,l_k^i}(\tau_m+\epsilon)) + \mathcal{SP}^i \end{aligned} \tag{6}$$

where $\mathcal{SP}^i$ is the additional handover cost when the selected processing edge node from the constellation i is unable to provide the requested service. The constant cost value $\mathcal{SP}^i = \kappa^i$ is added when the selected edge node is unable to provide the requested service else $\mathcal{SP}^i = 0$ is considered. From (1), the user should perform the offloading process before it passes through the coverage space of a connecting satellite. Thus, the following sojourn time constraint is defined:

$$T_{\rho_j}^{p}(a_{u_j}^{l_k^i}(\tau_m), \mathbf{b}_{u_j,l_k^i}(\tau_m+\epsilon)) \le T_{j,k}^{i,soj}(\tau_m) \tag{7}$$

C. Problem Formulation

We aim to solve the network selection problem over multi-tire edge computing facilities provided by several LEO constellations and terrestrial cloud facilities. By selecting proper edge nodes through a hierarchical decision-making process, we aim to minimize the overall task processing latency defined in (6), while respecting several user and system-related constraints. Thus we formed a constrained optimization problem:

$$\mathbf{P}: \mathcal{A}^*, \mathcal{B}^* = \underset{\mathcal{A},\mathcal{B}}{\operatorname{argmin}} \frac{1}{\bar{U}} \sum_{j=1}^{\bar{U}} T_{\rho_j}^{p}(a_{u_j}^{l_k^i}(\tau_m), \mathbf{b}_{u_j,l_k^i}(\tau_m+\epsilon)) \tag{8}$$

subject to the following constraints,

$$\mathbf{C1}: \text{Eq. (2)} \tag{9}$$

$$\mathbf{C2}: \text{Eq. (3a), (3b)} \tag{10}$$

$$\mathbf{C3}: T_{\rho_j}^{p}(a_{u_j}^{l_k^i}(\tau_m), \mathbf{b}_{u_j,l_k^i}(\tau_m+\epsilon)) \le T_{\rho_j} \quad \forall u_j \in \mathcal{U}, \tag{11}$$

$$\mathbf{C4}: \text{Eq. (7)} \tag{12}$$

$$\mathbf{C5}: b_{u_j,l_k^i}^{\hat{k}}(\tau_m+\epsilon) = 1 \Leftrightarrow s_{\rho_j} \in \mathcal{S}_{\hat{k}}^i \quad \forall j \tag{13}$$

$$\mathbf{C6}: a_{u_j}^{l_k^i}(\tau_m) = 1, \mathbf{b}_{u_j,l_k^i}(\tau_m+\epsilon) = 0, \Leftrightarrow s_{\rho_j} \in \mathcal{S}_k^i \quad \forall j \tag{14}$$

where $\mathcal{A}$ and $\mathcal{B}$ are the decision matrices representing the decisions made by the user and the covering satellites respectively. According to **C1** the number of edge nodes selected by the user/coverage satellite should not exceed one. Since each edge node can have a limited number of computation and bandwidth resources, **C2** puts the limit over the number of resources demanded by the users. Given the latency constraint nature of user task requests, according to **C3** the task processing operations should be performed before the task latency deadline. **C4** models the sojourn time constraint where each user's task should be processed before it passes through the selected coverage satellite. According to **C5** and **C6** the processing node should be able to provide a requested service to the user.

III. Solutions Methods

In the considered multi-level network selection problem, at a higher level, users aim to select a proper coverage satellite for offloading its data. Based upon the local environment's conditions, e.g., service availability, nearby satellite conditions, the coverage satellite then in the lower level, decides whether to compute the task by itself or to relay it to other satellite/cloud facilities. The higher-level decisions made by the users can impact the lower-level decision and their performance. Such problems can be embedded into a MDP, where decisions can be made sequentially at multiple levels with certain interdependencies. The Hierarchical Reinforcement Learning (HRL) process can be implemented to solve such challenging problems over dynamic scenarios.

A. MDP Models

In general, MDP for problem p can be defined as a tuple $\langle \mathcal{S}^p, \mathcal{A}^p, \mathcal{R}^p, \hat{\mathcal{P}}^p, \gamma^p \rangle$, with state space $\mathcal{S}^p$, action space $\mathcal{A}^p$, reward $\mathcal{R}^p$, state-transition probabilities $\hat{\mathcal{P}}^p$ and discount factor γ^p. Here, we introduce the MDP models for higher-level coverage satellite selection ($\mathcal{P}_1$) and lower-level processing node selection ($\mathcal{P}_2$) problems.

For the case of ($\mathcal{P}_1$), we aim to select the suitable coverage satellite node among different LEO constellations based on the user demands and satellite properties. We consider a discrete state space $\mathcal{S}^1 = \{s_1^1, \cdots, s_h^1, \cdots, s_H^1\}$ with H being a maximum number of states. The individual state s_h^1 is based upon the current state of the user and coverage satellite in terms of service requirements, location properties, coverage properties, distance measures, and resource availability. Thus, we define $s_h^1 = \{s_{\rho_j}, \hat{s}_k^i, \hat{d}_{jk}^i, \hat{U}_k^i, d_{jk}^{i,sj}\}$, where s_{ρ_j} is the requested service type, $\hat{s}_k^i$ is the binary variable modeling the kth satellites state w.r.t. requested service[3], $\hat{d}_{jk}^i \in \{0, \cdots, \bar{C}\}$ is the distance state modeling the distance between user j and the selected constellation i, $\hat{U}_k^i$ is the kth satellite resource state modeling the number of competing users that have already selected the satellite k[4], and $d_{jk}^{i,sj}$ models the coverage state through a binary variable which takes value 0 if the kth satellite is able to cover the user with more than ζ of its coverage space else 0.

Next, the action space $\mathcal{A}^1 = \{a_1^1, \cdots, a_{\hat{h}}^1, \cdots, a_{\hat{H}}^1\}$ includes a discrete set of $\hat{H}$ actions with $a_{\hat{h}}^1 = \{0,1\}_{1\times\bar{C}}$ being a binary vector modeling the user's network selection decision for $\mathcal{P}^1$. Note that each action should follow $\sum a_{\hat{h}}^1 \leq 1$ for satisfying the constraint (2). The state transition probability $Pr(s_{h'}^1 | s_h^1, a_{\hat{h}}^1)$ models the state transition dynamics from a current state action pair $(s_h^1, a_{\hat{h}}^1)$, to the next state $s_{h'}^1$. The reward $\mathcal{R}^1$ is based upon the global reward signal which depends upon the performance of lower-level edge node selection MDP.

By solving ($\mathcal{P}_2$), we aim to select the suitable edge node for processing the users' task based on the selected coverage satellite associated with the specific LEO constellation. We define a discrete state space $\mathcal{S}^2 = \{s_1^2, \cdots, s_l^2, \cdots, s_L^2\}$ of L states. The individual state s_l^2 can be based upon a selected coverage node, constellation properties, service availability, distance measures, etc. We define $s_l^2 = \{\hat{s}_k^i, \hat{s}_{\hat{k}}^i, \bar{d}_{k\hat{k}}^i, \bar{U}_{k\hat{k}}^i\}$, where $\bar{d}_{k\hat{k}}^i$ is the distance state modeled through a ternary variable. If the k and $\hat{k}$ are satellites from the same plane, it takes the value 0. If they are from different planes it takes the value 1. In case $\hat{k} = C_0$ the distance state becomes 2. $\bar{U}_{k\hat{k}}^i$ is another binary state variable modeling the impact of competing users associated with k and $\hat{k}$. It takes value 1 if the selected node $\hat{k}$ is having a high number of users associated with it than the coverage node k, else 0.

The action space $\mathcal{A}^2 = \{a_1^2, \cdots, a_{\hat{l}}^2, \cdots, a_{\hat{L}}^2\}$ contains a discrete set of $\hat{L}$ actions with $a_{\hat{l}}^2 = \{0,1\}_{1\times(\mathcal{N}_l^{i,max}+2)}$ being a binary vector modeling the processing node selection decision for $\mathcal{P}^2$. Note that each action should follow $\sum a_{\hat{l}}^2 \leq 1$ for satisfying (2). Here, $\mathcal{N}_l^{i,max}$ is the maximum number of LEO satellites that can connect with any satellite in a selected ith constellation. In addition to this nearby satellites, selected coverage satellites can prefer to compute the data by themselves or can relay it to the cloud facilities.

The state transition probability $Pr(s_{l'}^2 | s_l^2, a_{\hat{l}}^2, a_{\hat{h}}^1)$ models the state transition dynamics from a current state action pair $(s_l^2, a_{\hat{l}}^2)$, to the next state $s_{l'}^2$. The high-level action $a_{\hat{h}}^1$, i.e., selected coverage satellite can also impact the state transitions. If the coverage node k takes action $a_{\hat{l}}^2$ from certain state by selecting the edge facility $\hat{k}$, the intrinsic reward $\mathcal{R}^2$ received by the agent is defined as

$$\mathcal{R}^2(s_l^2, a_{\hat{l}}^2) = \hat{s}_k^i + \hat{s}_{\hat{k}}^i + \bar{d}_{k\hat{k}}^i + \bar{U}_{k\hat{k}}^i.$$

This reward is used to learn the lower level policies. We also define a global reward $\mathcal{R}^1$ for measuring the overall performance of network selection decisions given by:

$$\begin{aligned}\mathcal{R}^1(s_h^1, a_{\hat{h}}^1, a_{\hat{l}}^2) &= T_{\rho_j}^p(a_{\hat{h}}^1, a_{\hat{l}}^2) \\ &+ w_1 \left(\max\left\{T_{\rho_j}^p(a_{\hat{h}}^1, a_{\hat{l}}^2) - T_{j,k}^{i,soj}, 0\right\}\right) \\ &+ w_2 \left(\max\left\{T_{\rho_j}^p(a_{\hat{h}}^1, a_{\hat{l}}^2) - T_{\rho_j}, 0\right\}\right) \quad (15)\end{aligned}$$

where the first element measures the latency performance. The next two elements are measuring the performance in terms of service time and sojourn time constraints. If the node selection policies violate the constraints, an additional weighted penalty value is added with positive weights w_1, w_2.

B. Proposed Solutions

Given the complex and dynamic nature of the considered scenario, we consider Q learning as a model-free RL for finding optimal policies. It is one of the highly explored model-free strategies for determining the optimal policy in unknown environments. For example, policy π_p for problem p maps every state $s \in \mathcal{S}^p$ to action $a \in \mathcal{A}^p$. The Q-learning

[3]It takes value 0 if the requested service is available at k else 1.

[4]It takes value 0 if the number of users selecting the satellite k is less than $\bar{U}_i$ else 1.

strategy is based upon a state-action function, i.e., Q-function, defined as,

$$Q^{\pi_p}(s', a') = R^p(s', a') + \gamma \sum_{\hat{s} \in \mathcal{S}^p} \hat{\mathcal{P}}^p_{s'\hat{s}}(a') V^{\pi_p}(\hat{s})$$

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

$$Q^{\pi^*_p}(s', a') = R^p(s', a') + \gamma \sum_{\hat{s} \in \mathcal{S}^p} \hat{\mathcal{P}}^p_{s'\hat{s}}(a') V^{\pi^*_p}(\hat{s})$$

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

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

where ϵ is a learning rate. The Q-function can be estimated through 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 primary and the target Q-values where the loss function is defined as:

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{\hat{a}} Q_t(s', \hat{a}, \theta') - Q(s, a, \theta) \right)^2 \right] \quad (16)$$

where the primary values $Q(s, a, \theta)$ are based upon 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'.

In the considered HRL framework two RL agents are considered to find a proper coverage node selection policy for $\mathcal{P}_1$ and to determine proper edge computing facilities in $\mathcal{P}_2$ for computing the users' data (based upon the selected coverage node). The higher-level agent senses the environment state by processing the data associated with users' demands, constellation properties, etc., and selects the proper coverage node. This information is then received by the lower-level agent along with the current state information. The lower-level agent then uses this information to select a proper edge node for data processing. The intrinsic reward $\mathcal{R}_2$ is used to update the lower-level policies based upon the agents' observations, while the global reward $\mathcal{R}^1$ is considered while updating the higher-level policies.

Algorithm 1 details the DQN-based HRL process for the considered network selection problem. The process begins with the initialization of the primary/target of neural networks (w^P_H, w^T_H), (w^P_L, w^T_L) associated with $\mathcal{P}^1$ and $\mathcal{P}^2$ (lines 1-2). The neural networks associated with $\mathcal{P}^1, \mathcal{P}^2$ are having $\bar{L}^1, \bar{L}^2$ fully connected layers with $n_l, \forall l \in \bar{L}^1/\bar{L}^2$ neurons. The training process lasts for $\bar{N}$ episodes with a maximum $\mathcal{I}$ epoch per episode. Each training episode begins with the random initial state s_0 (lines 4-5). In each iteration it, DQNs are trained through the batch gradient descent approach described in Algorithm 2. Iteration begins by selecting a higher-level action $a^1_{\hat{h}}$ through Epsilon Greedy Policy (EGP) with parameter e^1 (line-8). The information related to the selected coverage node and its properties is then passed to the lower-level DQNs. Next, from state s^2_l, action $a^2_{\hat{l}}$ is selected through EGP with parameter e^2 and corresponding intrinsic reward $\mathcal{R}^2$ is determined (lines 10-12). The tuple $\langle s^2_l, a^2_{\hat{l}}, \mathcal{R}^2, s^2_{l,\text{new}} \rangle$ is then saved into replay memory $\mathcal{D}^2$ (line-13). Next, the DQN function is called for updating the parameters of w^P_L, w^T_L (line-14), which is then used to determine the global reward $\mathcal{R}^1$ (line-15). After that, the higher-level DQNs are updated through a gradient descent approach (lines 16-17). The DQN function defined in Algorithm 2 takes input as replay memory $\mathcal{D}^1/\mathcal{D}^2$, the batch size $\hat{k}^1/\hat{k}^2$, learning rate ϵ^1/ϵ^2 and discount factors γ^1/γ^2. The steps include the random batch selection (lines 1-2), loss value generation (line-3), primary network parameter update through gradient descent step (line-4), and target network parameter update step (line-6).

Algorithm 1 HRL for Network Selection over Multiple LEO Constellations

Input: $\mathcal{S}^1, \mathcal{A}^1, \mathcal{S}^2, \mathcal{A}^2, \mathcal{I}, \bar{N}, e^1, e^2, |\mathcal{D}^1|, |\mathcal{D}^2|, \epsilon^1, , \epsilon^2, \gamma^1, \gamma^2, \bar{it}$
Output: w^P_H, w^P_L

1: Initialize w^P_H, w^P_L
2: Duplicate policy networks to Target Networks,
i.e., $w^T_H = w^P_H$, $w^T_L = w^P_L$
3: **for all** $ep = 1, \ldots, \bar{N}$ **do**
4: Select Random s_0
5: $s^1_h \leftarrow s_0, it = 0$
6: **while** $it \neq \mathcal{I}$ **do**
7: $it = it + 1$
8: Select action $a^1_h \in \mathcal{A}^1$ with probability e^1
9: Determine next state $(s^1_{h,\text{new}})$
10: Select s^2_l based upon selected coverage node.
11: Select action $a^2_{\hat{l}} \in \mathcal{A}^2$ with probability e^2
12: Find next state $s^2_{l,\text{new}}$ and reward R^2
13: Store $\mathcal{D}^2 \leftarrow \langle s^2_l, a^2_{\hat{l}}, R^2, s^2_{l,\text{new}} \rangle$
14: w^P_L, w^T_L =DQN($\mathcal{D}^2, \hat{k}^2, w^P_L, w^T_L, it, \bar{it}, \epsilon^2, \gamma^2$)
15: Find Global Reward $\mathcal{R}^1$
16: Store $\mathcal{D}^1 \leftarrow (s^1_h, a^1_{\hat{h}}, R^1, s^1_{h,\text{new}})$
17: w^P_H, w^T_H =DQN($\mathcal{D}^1, \hat{k}^1, w^P_H, w^T_H, it, \bar{it}, \epsilon^1, \gamma^1$)
18: **end while**
19: **end for**
20: **return** w^P_H, w^P_L

IV. Numerical Results

The proposed HRL-based network selection solutions are simulated over the Python environment with several ML-related libraries, including NumPy, Pandas, Math, Matplotlib. The following benchmark solutions are considered for comparison purposes.

Algorithm 2 DQN Function

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

function DQN($\mathcal{D}, k, w^P, w^T, i, \bar{i}, \epsilon, \gamma$)
  Select Random batch of of k samples from $\mathcal{D}$
  Preprocess and pass the batch to w^P
  Find Loss between primary and Target Q values using (16)
  With gradient descent step update w^P
  Update w^T if $rem(i, \bar{i}) = 0$
end function
return $\{w^P, w^T\}$

Random Method (RM) -: In this case, the coverage node and computation nodes are selected randomly with uniform probability distributions. Thus,

$$a^{l_k^i}_{u_j}(\tau_m) = 1 \quad with \quad \left(C_i \in_R C \wedge T^{i,soj}_{j,k}(\tau_m) > 0\right)$$
$$b^{\hat{k}}_{u_j,l_k^i}(\tau_m + \epsilon) = 1 \quad with \quad \hat{k} \in_R (\mathcal{N}_l^i \cup C_0) \quad \forall m, j$$

Cloud Computing Method (CCM)-: In this case, the coverage node is selected randomly with a uniform probability distribution. Then the selected coverage node routes the data to the cloud facility for computation purposes. Thus,

$$a^{l_k^i}_{u_j}(\tau_m) = 1 \quad with \quad \left(C_i \in_R C \wedge T^{i,soj}_{j,k}(\tau_m) > 0\right)$$
$$b^{\hat{k}}_{u_j,l_k^i}(\tau_m + \epsilon) = 1 \quad with \quad \hat{k} = C_0, \quad \forall m, j$$

With this approach, users can explore rich sets of cloud computing resources.

Coverage Satellite Approach (CSA)-: In this static approach, the selected coverage node prefers to compute the users' task by itself without relaying it towards other edge facilities. Thus,

$$a^{l_k^i}_{u_j}(\tau_m) = 1 \quad with \quad \left(C_i \in_R C \wedge T^{i,soj}_{j,k}(\tau_m) > 0\right)$$
$$b^{\hat{k}}_{u_j,l_k^i}(\tau_m + \epsilon) = 0, \quad \forall m, j$$

This approach can be efficient in terms of overall communication cost. However, it can have a limited impact in terms of distributed resource utilization over highly dispersed LEO networks.

We have considered 3 LEO constellations with a different number of satellites. A service area of a radius of 1 km is considered with randomly located users. A set of users between 200 and 2000 are considered with a probability of being active $P_a = 0.1$. Users are served by 3 LEO constellations and a cloud facility with six different services. LEO satellites are able to host 3 services each. Additionally, $\mathcal{SP}^i = 0.1$, $w_1 = 0.4$ and $w_2 = 0.4$ is considered. For DQN simulation primary and target networks with layers $\bar{L}^1 = \bar{L}^2 = 4$ are considered with learning parameters $e^1 = e^2 = 0.7$, $\mathcal{D}^1, \mathcal{D}^1 = 4000$, $\epsilon^1 = \epsilon^2 = 0.05$, $\gamma^1, \gamma^2 = 0.98$. Also, the learning process includes $\bar{N} = 50$ with $\mathcal{I} = 10^3$ and $\bar{it} = 50$. Main simulation parameters are provided in Table I.

In Fig. 2, the average latency cost for processing the users' data over multi-tier T/NTN edge computing facilities is

TABLE I
SIMULATION PARAMETERS

LEO Heights (h_1, h_2, h_3)	250, 700, 1000 km
Intra-plane Distance $(d^1_{in}, d^2_{in}, d^3_{in})$	100, 150, 200 km
Inter-plane Distance $(d^{i,min}_{out}, d^{i,max}_{out})$	800, 1200 km
LEO Computation $(c^i_k \cdot f^i_k)$	50 GFLOPS
User Computation $(c_j \cdot f_j)$	8 GFLOPS
Task Size $(D_{\rho_j}, D^r_{\rho_j})$	5 MB, 1 MB
Task Computation (Ω_{ρ_j})	$D_{\rho_j} \cdot 10^3$
Task Latency (T_{ρ_j})	5 Sec.

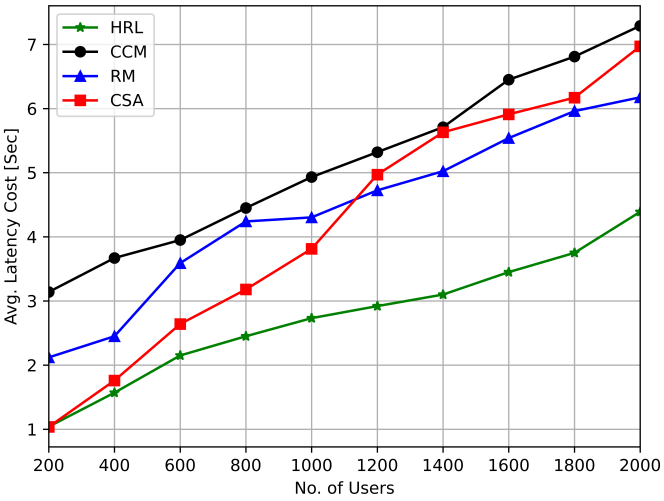

Fig. 2. Average Latency Cost

presented. The CCA approach considered the cloud computing facilities for computing the users' tasks while selecting the coverage LEO nodes randomly, therefore inducing higher latency costs. The CSA approach avoids relaying the users' task data to other nodes and prefers to compute them by itself. With this approach, it can reduce the overall communication time. However, with such static computation policies, it is unable to explore the distributed computation resources, and as the number of users grows, the performance of the CSA method worsens. In addition to this, it also suffers from a large number of service handovers inducing additional costs (presented later in Fig. 4). Similarly, RM also suffers from higher latency requirements due to suboptimal coverage/computation node selection policies. The proposed HRL method through proper node selection policies is able to distribute the computation load over different LEO satellites and cloud facilities, thus reducing the overall latency cost.

In a considered multi-user scenario, it is also important to highlight the reliability of proposed solutions in terms of service latency demands. In Fig. 3, we have presented the average number of service latency failures for different methods. With the appropriate network selection policies, the proposed HRL method is able to serve the users with reduced latencies and hence have better reliability in terms of service latency requirements. However, with the static/random network selection policies, other benchmark solutions suffer

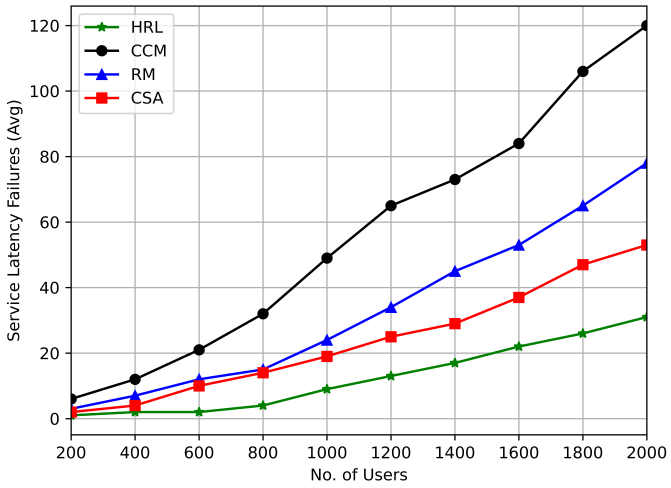

Fig. 3. Avg. No. of Service Latency Failures

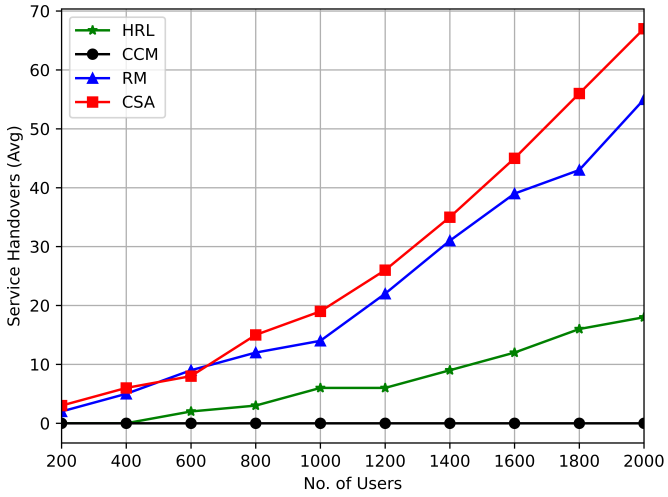

Fig. 4. Average Number of Required Service Handovers

(RM, CCM, CSA) with the large number of failures. In particular CCM approach, with a static policy of selecting cloud computing facilities for serving users, is unable to support the low latency demands.

In multi-service edge computing scenarios, it is important to highlight the number of service handovers required while serving the ground users. With huge amounts of storage resources, cloud facilities can host multiple services, and thus considered CCM approach can be highly reliable. However, as shown in the previous figures, with such static policies, the latency cost can be huge. On the other hand, the satellite nodes with limited resources can host a certain number of services only. Therefore, static approaches such as CSA, where only coverage LEO nodes are exported for serving the ground users, can suffer from a large number of service handovers. The RM method can reduce the number of handovers required by distributing the user requests. However, with suboptimal node selection policies, it can still suffer from many handovers. The proposed HRL approach can effectively distribute the user demands at various edge computing facilities, having proper services through optimal policies and thus, reducing the overall number of handovers.

V. Conclusion

In this work, we have considered a multi-level network selection problem over multi-tier edge computing facilities. In particular, a set of multi-plane LEO satellite constellations and terrestrial cloud computing facilities are considered for serving ground users with multiple services. With this, a multi-level network selection problem is formed as a constrained optimization problem for minimizing the overall latency costs. A nover HRL-based solution is proposed for enabling the multi-level decision-making process of network selection problems. Next, DQN methods are applied for finding the optimal network selection policies over a dynamic environment. In the end, performance is analyzed through a Python-based simulation, showing the improvements in terms of overall cost and reliability.

References

[1] C.-X. Wang *et al.*, "On the road to 6G: Visions, requirements, key technologies, and testbeds," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 2, pp. 905–974, 2023.

[2] G. Araniti, A. Iera, S. Pizzi, and F. Rinaldi, "Toward 6G non-terrestrial networks," *IEEE Netw.*, vol. 36, no. 1, pp. 113–120, 2022.

[3] 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.

[4] H. Xie, Y. Zhan, G. Zeng, and X. Pan, "LEO mega-constellations for 6G global coverage: Challenges and opportunities," *IEEE Access*, vol. 9, pp. 164 223–164 244, 2021.

[5] S. S. Shinde and D. Tarchi, "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.

[6] T. Ren, J. Niu, B. Dai, X. Liu, Z. Hu, M. Xu, and M. Guizani, "Enabling efficient scheduling in large-scale UAV-assisted mobile-edge computing via hierarchical reinforcement learning," *IEEE Internet Things J.*, vol. 9, no. 10, pp. 7095–7109, 2022.

[7] Z. Gao, L. Yang, and Y. Dai, "Large-scale computation offloading using a multi-agent reinforcement learning in heterogeneous multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 6, pp. 3425–3443, 2023.

[8] Z. Tianqing, W. Zhou, D. Ye, Z. Cheng, and J. Li, "Resource allocation in IoT edge computing via concurrent federated reinforcement learning," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1414–1426, 2022.