

# In-Space Computing for IoT Data Processing via Low Earth Orbit Satellites

Swapnil Sadashiv Shinde\*, Gayathri Guruvayoorappan<sup>†</sup>, Tomaso De Cola<sup>†</sup> and Daniele Tarchi<sup>‡</sup>

\*CNIT—University of Florence Research Unit, 50139 Firenze, Italy. Email: swapnil.shinde@cnit.it

<sup>†</sup>Institute of Communications and Navigation, German Aerospace Center (DLR), Weßling, Germany  
Email: {gayathri.guruvayoorappan, tomaso.decola}@dlr.de

<sup>‡</sup>Department of Information Engineering, University of Florence, 50139 Firenze, Italy. Email: daniele.tarchi@unifi.it

**Abstract**—The Internet of Things (IoT) technology has become widespread in numerous domains, especially in remote regions where resources are scarce. The deployment of IoT technology has been facilitated by distributed edge computing (EC) systems spanning terrestrial to aerospace environments. This paper focuses on optimizing data routing, selection of computing nodes, and buffering policies for processing data in space by exploiting Low Earth Orbit (LEO) satellites. We introduce a constrained optimization problem to reduce the combined energy and latency costs associated with space data processing. This problem is structured as a hierarchical decision-making model using the Markov Decision Process (MDP) methodology and is addressed through Reinforcement Learning (RL) techniques. The proposed HRL framework employs two RL agents leveraging Deep Q Networks (DQN) to enhance data processing by optimizing node selection and improving buffering strategies for task execution. Simulation results show that the proposed framework outperforms existing methods in latency and energy efficiency.

**Index Terms**—Internet of Things, Edge Computing, On-board Processing, Non-Terrestrial Networks, Hierarchical Reinforcement Learning, Deep Q Networks

## I. INTRODUCTION

The Internet of Things (IoT) has been widely applied in diverse application scenarios [1]. With 6G, these demands will expand, offering societal benefits. IoT requires significant storage and processing, while traditional systems such as Edge Computing (EC) and Cloud Computing struggle with resource limitations, high costs, reliance on backhaul, unstable connections, and security challenges [2]. Remote IoT setups also face connectivity issues, prompting the need for joint innovative computing and networking solutions [3].

Recent studies on integrated Terrestrial and Non-Terrestrial Networks (T-NTN) aim to develop reliable, adaptable networks with increased capacity and coverage. NTN are crucial for enhancing the coverage and capacity of 6G networks [4], allowing distributed computing and IoT support [5]. These T-NTN systems seamlessly blend terrestrial and non-terrestrial layers [3], forming a distributed network with nodes at different altitudes. By merging EC with NTN, distributed computing that spans terrestrial to aerospace areas becomes accessible via

This work was supported by the European Space Agency under the framework of EDEGCOLB (“Joint communication and edge computing load balancing techniques for satellite networks”) project, ESA Contract n. 4000140850/23/NL/FGL. The view expressed herein can in no way be taken to reflect the official opinion of the European Space Agency.

various communication links. With the advent of intelligent LEO satellites that feature EC capabilities, the efficiency of data processing for remote IoT devices is significantly improved compared to satellite networks at higher altitudes [6].

With a large number of IoT devices and satellite nodes, an effective data distribution between satellites is essential for optimal utilization of resources [3]. It is crucial to determine effective communication paths for satellite relays and computation nodes to manage IoT data. Satellite nodes enhance the efficiency of on-board processing through buffering, as they temporarily store data when processors are occupied, resulting in a balance between communication and computation, where data can be either retained or sent for processing. Therefore, optimizing data routing, computation node selection, and buffering policies is essential for effective space-based IoT data processing. These choices affect each other; for example, selecting computation nodes might depend on path choices, while buffering strategies might rely on the condition of selected satellite nodes.

The joint solution to these challenges can be difficult when dealing with dynamic networks that have a variety of nodes. Recently, machine learning (ML) frameworks and algorithms have been developed to provide intelligent solutions in the realm of wireless networks. Reinforcement learning (RL) stands out as a particularly effective approach for multi-optimization problems that involve both task processing and node selection within dynamic satellite networks. For instance, [2] introduces intelligent task offloading strategies based on RL for multilayer non-terrestrial edge networks. Furthermore, [7] covers joint air-ground network edge facilities with a focus on dependency-sensitive task offloading.

**Main Contributions:** We focus on a LEO satellite network with EC capabilities to facilitate task offloading from remote IoT devices. These satellites are equipped with multicore processing and storage functions. Due to resource constraints and a large number of IoT devices, formulating efficient offloading strategies is essential. We propose a constrained optimization problem that aims to minimize latency and energy consumption in space-based IoT data processing. By decomposing this problem into joint subproblems, we employ both traditional optimization and Hierarchical RL (HRL) techniques to offer an effective multilevel decision-making strategy for managing

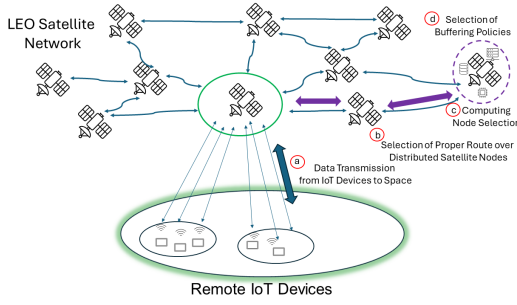


Fig. 1. Space Edge Computing for Remote IoT Scenario.

these hierarchical complexities.

## II. SYSTEM MODEL AND PROBLEM FORMULATION

The system model comprises a group of  $D$  remote IoT devices, denoted as  $\mathcal{I} = \{i_1, \dots, i_d, \dots, i_D\}$ , and  $S$  LEO satellites, represented by  $\mathcal{L} = \{l_1, \dots, l_s, \dots, l_S\}$ . These LEO satellites offer EC services and function as satellite relays for IoT devices, creating a distributed space-based EC environment. It is assumed that IoT devices are scattered over a large area  $\mathcal{A}$  and can connect to several nearby LEO satellites. For simplicity, IoT devices are connected to the nearest satellite node, named the source satellite  $l_s^{in}$ , indicating that the  $s$ th satellite acts as the source node in the considered LEO satellite system, routing IoT data to any other satellite node for distributed processing. We define  $\mathcal{L}^{in} = \{l_s, \forall s \in l_s^{in}\} \subset \mathcal{L}$  as a set of source satellites.

Fig. 1 illustrates the components of the system model and the related challenges in processing IoT data in space. The discrete-time system assumes constant network parameters during each time interval  $\tau$ . Here,  $\tau_i$  represents the  $i$ th interval, defined as  $\tau_i = \{t | t \in [i\tau, (i+1)\tau]\}$ . A generic IoT device  $d$  generates a task  $\rho_d(\tau_i)$  described by the tuple  $\langle D_{\rho_d}, D_{\rho_d}^r, \Omega_{\rho_d} \rangle$ , where  $D_{\rho_d}$  Bytes is the task size,  $D_{\rho_d}^r$  Bytes is the output size and  $\Omega_{\rho_d}$  are the required CPU cycles. The generic  $s$ th LEO satellite has a coverage radius  $R_{l_s}$ , a processing capacity of  $c_{l_s}$  FLOPS per CPU cycle, CPU frequency  $f_{l_s}$ , and  $\mathcal{L}_{l_s}$  CPU cores. It facilitates communication with terrestrial IoT devices and other satellites through a bandwidth  $B_{l_s}$ .

Let  $a_0(d, s, \tau_i)$  be a binary variable indicating whether the  $d$ th IoT device connects to the  $s$ th satellite node, defined as:

$$\sum_{s'} a_0(d, s', \tau_i) \leq 1, \quad \forall d \quad (1)$$

This constraint ensures that each IoT device connects to at most one LEO satellite for data transmission.

In a distributed EC system, the primary objective is to identify the most suitable node for remote data processing. The effective distribution of IoT data is critical for the efficient utilization of satellite resources. Thus, when data are not processed at the input node, the data transmission routes become crucial to serving multiple IoT devices and optimizing resource usage. To minimize transmission costs, it is essential to choose the appropriate satellite path from the source to any potential processing satellite node. We use a binary variable  $a(s, s', \tau_i)$

to indicate the chosen path between satellites  $l_s$  and  $l_{s'}$ , where  $a(s, s', \tau_i)$  equals one if selected, otherwise it is zero. Each satellite can choose almost one other to forward IoT data:

$$\sum_{s'} a(s, s', \tau_i) \leq 1, \quad \forall s \quad (2)$$

A path between satellites  $s$  and  $k$  is defined as  $P_{s,k}(\tau_i) = \{s, s', \dots, s'', k\}$ , connecting them. Furthermore,  $\mathcal{P}_{s,k}(\tau_i) = \{P_{s,k}(\tau_i)\}$  is the set of all possible paths between  $s$  and  $k$ . Our goal is to find the optimal  $P_{s,k}^*(\tau_i)$  to minimize transmission costs.

After identifying the optimal path  $P_{s,k}^*(\tau_i)$  among the satellite nodes, the IoT devices connected to the  $s$ th satellite can utilize the EC resources of the satellite nodes. Due to the numerous IoT devices, assigning the appropriate computing node is crucial. A computing node selection variable,  $b(d, s, \tau_i)$ , is proposed to allocate the IoT device  $i_d$  to the satellite  $l_s$  for data processing. If node  $l_s$  on the path  $P_{s',k'}^*(\tau_i)$  is chosen to process data from the device  $d$ , then  $b(d, s, \tau_i)$  equals one, where  $P_{s',k'}^*(\tau_i)$  represents the optimal path from the input satellite  $l_{s'}$  of the device  $d$  to the satellite  $l_{k'}$ . Moreover,

$$\sum_{s \in P_{s',k'}^*(\tau_i)} b(d, s, \tau_i) = 1, \quad \forall d \quad (3)$$

ensures each IoT device selects only one satellite node for data processing.

In addition, satellite EC nodes are supposed to have both processing and transmission buffers to efficiently manage IoT processing requests and to lower transmission costs, each with limited capacity. The processing buffer  $\mathcal{B}_{pr}$  can hold  $|\mathcal{B}_{pr}|$  MByte of data from IoT devices. After processing, the results are sent to a transmission buffer  $\mathcal{B}_{tx}$ ,  $|\mathcal{B}_{tx}|$  MByte in size, saving on the use of the satellite transmission link. We introduce variables to allocate tasks to buffers:  $c_p(d, s, \tau_i)$  is one if device  $i_d$  tasks go to satellite processing buffer  $l_s$  and zero if processing occurs immediately in  $l_s$ ;  $c_{tx}(d, s, \tau_i)$  is one if the processed task  $\rho_d$  is stored in the transmission buffer, zero if sent immediately to the destination.

### A. Joint Latency and Energy Costs

Space task processing involves transmitting IoT data through satellites, processing, and buffering phases. The latency and energy spent for each stage are modeled as follows.

The formula for computing the  $\rho_d$ th task's time and energy on  $s'$ th satellite node is given by:

$$T_{c,s'}^{\rho_d} = \frac{\Omega_{\rho_d}}{c_{s'} f_{s'}}, \quad E_{c,s'}^{\rho_d} = T_{c,s'}^{\rho_d} P_{c,s'} \quad (4)$$

where  $c_{s'}$  are the FLOPS per cycle,  $f_{s'}$  the CPU frequency of a generic satellite ( $l'_s$ ). In (4),  $P_{c,s'}$  is the computing power of the satellite  $s'$ . IoT devices  $i_d$  send data to an edge facility and receive results back. The transmission time and energy for task  $\rho_d$  between nodes  $k$  and  $l$  are:

$$T_{tx,kl}^{\rho_d}(\tau_i) = \frac{D_{\rho_d}}{r_{kl}(\tau_i)}, \quad E_{tx,kl}^{\rho_d}(\tau_i) = T_{tx,kl}^{\rho_d}(\tau_i) P_{t_k} \quad (5)$$

where  $r_{kl}(\tau_i)$  is the data rate, and  $Pt_k$  the transmission power of node  $k$ . For reception, the time and energy for data size  $D_{\rho_d}^r$  by node  $k$  from  $l$  are given by:

$$T_{rx,kl}^{\rho_d}(\tau_i) = \frac{D_{\rho_d}^r}{r_{kl}(\tau_i)}, \quad E_{rx,kl}^{\rho_d}(\tau_i) = T_{rx,kl}^{\rho_d}(\tau_i)Pr_k \quad (6)$$

where  $Pr_k$  is the reception power. The  $\rho_d$  task propagation time between nodes  $e$  and  $e'$  is given by,

$$T_{pr,ee'}^{\rho_d}(\tau_i) = \frac{D_{\rho_d}}{c} \quad (7)$$

where  $c$  is the speed of light. The energy cost for propagation is negligible and is excluded from the model.

Task storage penalties impact system efficiency; this research aims to enhance task processing through optimized routing, computing node selection, and buffering. The penalty for storing tasks is given by:

$$T_{pb,e}^{\rho_d}(\tau_i) = \zeta_1 b(d, e, \tau_i) c_p(d, s, \tau_i), \quad E_{pb,e}^{\rho_d}(\tau_i) = T_{pb,e}^{\rho_d}(\tau_i) P_{s_{k,cb}} \quad (8)$$

where  $\zeta_1$  denotes the buffering latency cost, and  $P_{s_{k,cb}}$  is the power used for buffer activation. Similarly, the task transmission buffering costs can be modeled as:

$$T_{tx,e}^{\rho_d}(\tau_i) = \zeta_2 b(d, e, \tau_i) c_{tx}(d, s, \tau_i), \quad E_{tx,e}^{\rho_d}(\tau_i) = T_{tx,e}^{\rho_d}(\tau_i) P_{s_{k,txb}} \quad (9)$$

where  $\zeta_2$  indicates the transmission buffering latency cost, and  $P_{s_{k,txb}}$  is the power requirement for buffer activation.

We define the total latency and energy costs for processing the  $d$ th device data at an edge facility as:

$$\begin{aligned} T^{\rho_d}(\tau_i) &= T_{tx,ds}^{\rho_d}(\tau_i) \\ &+ \sum_{\substack{s' \in P_{s,k} \setminus k, \\ s'' \in P_{s,k} \setminus s}} a(s', s'', \tau_i) \left[ T_{tx,s's''}^{\rho_d}(\tau_i) + T_{pr,s's''}^{\rho_d}(\tau_i) + T_{rx,s's''}^{\rho_d}(\tau_i) \right] \\ &+ \sum_{s' \in P_{s,k}} b(d, s, \tau_i) \left[ T_{c,s'}^{\rho_d} + c_p(d, s', \tau_i) T_{pb,s'}^{\rho_d}(\tau_i) \right. \\ &\quad \left. + c_{tx}(d, s', \tau_i) T_{tx,s'}^{\rho_d}(\tau_i) \right] \quad (10) \end{aligned}$$

$$\begin{aligned} E^{\rho_d}(\tau_i) &= E_{tx,ds}^{\rho_d}(\tau_i) \\ &+ \sum_{\substack{s' \in P_{s,k} \setminus k, \\ s'' \in P_{s,k} \setminus s}} a(s', s'', \tau_i) \left[ E_{tx,s's''}^{\rho_d}(\tau_i) + E_{pr,s's''}^{\rho_d}(\tau_i) + E_{rx,s's''}^{\rho_d}(\tau_i) \right] \\ &+ \sum_{s' \in P_{s,k}} b(d, s, \tau_i) \left[ E_{c,s'}^{\rho_d} + c_p(d, s', \tau_i) E_{pb,s'}^{\rho_d}(\tau_i) \right. \\ &\quad \left. + c_{tx}(d, s', \tau_i) E_{tx,s'}^{\rho_d}(\tau_i) \right] \quad (11) \end{aligned}$$

### B. Problem Formulation

Our aim is to minimize the total latency and energy consumption in processing IoT data through LEO satellite-enabled space EC. This involves optimizing device-to-satellite routes, selecting computing nodes, and improving IoT task buffering.

We therefore pose a constrained optimization problem as follows,

$$\mathbf{P1} : \min_{\mathbf{A}_0, \mathbf{A}, \mathbf{B}, \mathbf{C}_{pb}, \mathbf{C}_{tx}} \left\{ \frac{1}{D} \sum_{d=1}^D \gamma_1 T^{\rho_d}(\tau_i) + \gamma_2 E^{\rho_d}(\tau_i) \right\} \quad (12)$$

s.t.,

$$\mathbf{C1} : \text{Eq. (1)-(2)} \quad (13)$$

$$\mathbf{C2} : \text{Eq. (3)} \quad (14)$$

$$\mathbf{C3} : \sum_d c_p(d, s, \tau_i) D_{\rho_d} \leq |\mathcal{B}_{pr}| \quad \forall s \quad (15)$$

$$\mathbf{C4} : \sum_d c_{tx}(d, s, \tau_i) D_{\rho_d}^r \leq |\mathcal{B}_{tx}| \quad \forall s \quad (16)$$

$$\mathbf{C5} : \sum_d b(d, s, \tau_i) \Omega_{\rho_d} \leq c_{ls} f_{ls} \quad \forall s \quad (17)$$

$$\mathbf{C6} : \sum_s a(s, s', \tau_i) B_{s,s'} \leq B_{l_s} \quad \forall s' \quad (18)$$

$$\mathbf{C7} : \sum_d a_0(d, s, \tau_i) B_{d,l_s^{in}} \leq B_{l_{s^{in}}} \quad \forall s \quad (19)$$

where  $\mathbf{A}_0$  denotes the initial set of satellite nodes chosen by the IoT devices for orbital data transmission. The set  $\mathbf{A}$  represents the optimal routes from these satellite nodes to other computing facilities.  $\mathbf{B}$  is a matrix that indicates the selection policies of the computing nodes, while  $\mathbf{C}_{pb}$  and  $\mathbf{C}_{tx}$  are matrices that outline buffer policies for processing and transmission, respectively.  $\mathbf{C1}$  specifies route choice policies for IoT devices, which allow IoT data to travel over a single link between nodes.  $\mathbf{C2}$  deals with the selection of computation nodes, indicating that a IoT data is processed at a single edge facility.  $\mathbf{C3}$  sets the maximum processing buffer size, where each buffer can store IoT data up to a certain limit. Likewise,  $\mathbf{C4}$  sets the constraint on transmission buffer data. According to  $\mathbf{C5}$ , the computing resources allotted to the satellite nodes to IoT devices are limited by their computational capacity. Finally,  $\mathbf{C6}$  and  $\mathbf{C7}$  restrict the bandwidth resources of the edge nodes to transmit IoT data.

### III. PROPOSED SOLUTIONS

The problem focuses on remote IoT devices that select appropriate satellite nodes for remote processing. This challenge is divided into the following problems:

- $P_0$ : Choosing the appropriate satellite for IoT data upload.
- $P_1$ : Determining the optimal route for the data to the potential satellite-enabled EC nodes.
- $P_2$ : Selecting a satellite node to process the data.
- $P_3$ : Determining proper policies for processing and transmission buffers.

Addressing these tasks improves the efficiency of IoT data processing. Choosing nearby nodes reduces transmission costs but might not utilize computing resources effectively. Remote nodes may offer more computing power but at a higher cost. Tackling these issues separately or with static rules can be inefficient. We propose a joint optimization strategy for  $P_0, P_1,$

$P_2$ , and  $P_3$ , integrating traditional optimization with HRL for a comprehensive solution. For  $P_0$ , the appropriate node selection resorts to the the nearest satellite, modeled as:

$$a_0(d, s, \tau_i) = 1 \iff s = \underset{l_{s'}}{\operatorname{argmin}} \{\bar{d}_{d,s'}(\tau_i)\} \quad (20)$$

where  $\bar{d}_{d,s'}(\tau_i)$  measures the distance between IoT device  $d$  and satellite  $l_{s'}$  at time  $i$ . Regarding  $P_1, P_2, P_3$ , we propose an HRL-based solution for intelligent routing, choice of computing nodes, and buffering for optimal latency and energy efficiency in satellite IoT settings. Independent agents are designed to minimize the uncertainty of the HRL training process. A Markov Decision Process (MDP) is designed to develop the HRL strategy, which is detailed after the definition of the training scenarios.

#### A. Scenario Definition for Satellite Environment

Satellite networks represent distributed, dynamic and complex systems utilized for EC. Variations in regional properties, satellite node capabilities, and IoT application types lead to a diverse device distribution between source satellites. Consequently, a universal solution may not be effective everywhere. To reduce uncertainty in training and testing intelligent agents, we introduce specific satellite-IoT device scenarios based on device distribution across input satellites and the satellite node distribution around them. Let  $\mathcal{S} = \{E_1, \dots, E_{p_{max}}\}$  with  $p_{max}$  scenarios, where the  $p$ th scenario for the satellite  $l_s$  at time  $i$  is  $E_p(l_s(\tau_i)) = \{U_{l_s}(\tau_i), \bar{S}_{l_s}(\tau_i)\}$ . Here,  $U_{l_s}(\tau_i)$  is the number of devices connected to  $l_s$ , and  $\bar{S}_{l_s}(\tau_i)$  is the set of neighboring satellites directly connected to  $l_s$ . Due to the large coverage radius of satellite nodes, many IoT devices may be associated with one source satellite, making individual management computationally complex for RL-based solutions. However, tasks for each individual IoT device are often small, allowing device grouping. Each satellite  $l_s$  can host  $0 \leq U_{l_s} \leq U_{max}$  devices, classified into  $g_1, \dots, g_G$  groups. The  $\bar{g}$ th group has a device count given by:

$$|\bar{g}| = \begin{cases} \lfloor \frac{U}{G} \rfloor & \text{if } i \leq U \mod G \\ \lceil \frac{U}{G} \rceil & \text{if } i > U \mod G \end{cases}$$

Thus,  $E_p(l_s(\tau_i)) = \{g_i(l_s), \bar{S}_{l_s}(\tau_i)\}$  is defined as an ML training scenario for satellite  $l_s$  in network selection. Similarly, the buffering policy varies by device groups per satellite, defined by  $\mathcal{S}' = \{E_1, \dots, E_{q_{max}}\}$  with  $q_{max}$  scenarios. The  $q$ th scenario for satellite  $l_s$  is  $E_q(l_s(\tau_i)) = \{g_i(l_s)\}$ , for the selection of the buffering process based on its device groups.

#### B. MDP Formation

The subproblems  $P_1, P_2$ , and  $P_3$ , can create a hierarchy of problems that allows information sharing and performance optimization, where  $P_1$  and  $P_2$  can be merged into a single step to find the optimal computing node along a route of multiple satellites acting as relays. These problems are modeled using hierarchical decision-making via the MDP approach and resolved using HRL methods. Subproblems  $P_1$ - $P_3$  are characterized as

two MDPs to develop offloading policies, with state, action, and reward values facilitating information exchange. Generally, an MDP for any problem  $p$  is a tuple  $\langle \Sigma^p, \mathcal{A}^p, \mathcal{R}^p, \hat{\mathcal{P}}^p, \gamma^p \rangle$ , including the state space  $\Sigma^p$ , the action space  $\mathcal{A}^p$ , a reward  $\mathcal{R}^p$ , the state transition probabilities  $\hat{\mathcal{P}}^p$  and the discount factor  $\gamma^p$ .

For each satellite  $l_s$ , given its scenario  $E_p(l_s(\tau_i)) = \{g_i(l_s), \bar{S}_{l_s}(\tau_i)\}$ , a network selection decision is needed to assign device groups to neighboring satellites for load balancing. We define an MDP for the network selection in scenario  $p$ . Specifically,  $\Sigma^{2,p} = \{\sigma_1^{2,p}, \dots, \sigma_H^{2,p}\}$  represents a discrete state space with  $H$  states, where the state  $\sigma_h^{2,p}$  is given by  $\sigma_h^{2,p} = [n_i, \forall i \in \bar{S}_{l_s}(\tau_i)]$ , with  $n_i$  indicating the number of device groups assigned to the  $i$ th neighboring satellite. Furthermore,  $\mathcal{A}^{2,p} = \{a_1^{2,p}, \dots, a_{\hat{H}}^{2,p}\}$  represents the discrete action space, comprising  $\hat{H}$  actions, where an action  $a_h^{2,p}$  is defined as  $a_h^{2,p} = [m_j, \forall j \in g_i(l_s)]$ , with  $m_j$  referring to the satellite ID assigned to the  $j$ th device group from  $g_i(l_s)$ . This method extends across multiple levels to improve routing and solve  $P_1$ , and optimizing computation through  $P_2$ . Each routing decision adheres to MDP policies, progressively selecting an edge node for computation. The efficiency of  $P_1$  and  $P_2$  is influenced by the buffering policies of  $P_3$ , determining the reward received based on these agent policies.

The subproblem  $P_3$  obtains information from the upper layers  $P_1$  and  $P_2$ , including the selected edge node and the routing strategy. It then formulates a buffering policy for its scenario  $E_q$ , handling task allocation to storage buffers and edge processors. This choice depends on current storage and processor loads. The task groups for each satellite determine the training scenario. The discrete state space for  $P_3$  is  $\Sigma^{3,q} = \{\sigma_1^{3,q}, \dots, \sigma_m^{3,q}, \dots, \sigma_M^{3,q}\}$  with  $M$  states where  $\sigma_m^{3,q}$  represents each state according to the processor and buffer status. The action space  $\mathcal{A}^{3,q} = \{a_1^{3,q}, \dots, a_m^{3,q}, \dots, a_M^{3,q}\}$  includes  $\hat{M}$  actions. The agent decides to assign tasks to the processor (0) or to a buffer (1). Reward  $\mathcal{R}^3$  evaluates path, node, and buffer selection efficiency:

$$\mathcal{R}^3 = \gamma_1 T^{\rho_d}(a_h^2(\tau_i), a_m^3(\tau_i)) + \gamma_2 E^{\rho_d}(a_h^2(\tau_i), a_m^3(\tau_i))$$

assessing latency and energy outcomes of the policies.

#### C. Deep Q Network based solution

Q learning is used to find optimal policies in complex dynamic settings. It is a widely-used model-free strategy for determining optimal policies in unknown environments. Our goal is to show the effectiveness of the proposed HRL solution. The policy  $\pi_p$  for the problem  $p$  associates each state  $\sigma \in \Sigma^p$  with an action  $a \in \mathcal{A}^p$ . The Q-learning strategy employs a state-action function, the Q function, defined as

$$Q^{\pi_p}(\sigma', a') = R^p(\sigma', a') + \gamma \sum_{\hat{\sigma} \in \Sigma^p} \hat{\mathcal{P}}_{\sigma', \hat{\sigma}}^p(a') V^{\pi_p}(\hat{\sigma}),$$

where  $Q^{\pi_p}(\sigma', a')$  is the state-action value function for policy  $\pi$ , expressing the discounted cumulative reward from state  $\sigma'$  after taking action  $a'$  and then following  $\pi_p$ .  $R^p(\sigma', a')$  is

the immediate reward from the state-action pair  $(\sigma', a')$ . The optimal Q-value is expressed as

$$Q^{\pi_p^*}(\sigma', a') = R^p(\sigma', a') + \gamma \sum_{\hat{\sigma} \in \Sigma^p} \hat{P}_{\sigma' \hat{\sigma}}^p(a') V^{\pi_p^*}(\hat{\sigma}),$$

where  $V^{\pi_p^*}(\hat{\sigma}) = \min_{\hat{a} \in \mathcal{A}^p} Q^{\pi_p^*}(\sigma', \hat{a})$  for the optimal policy  $\pi_p^*$ . Q values can be estimated recursively:

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

with  $\epsilon$  as the learning rate. The Q function can be estimated via traditional methods such as temporal differences and tabular methods. For RL problems with high-dimensional state/action spaces, methods such as function approximation can reduce training complexity and improve generalization. An approximation of a neural network-based function for Q,  $Q(\sigma', a'; \theta) \approx Q(\sigma', a')$ , uses weights  $\theta$ , adjusted during training to minimize the mean square error.

DQN, a widely studied method in neural networks for RL agent policy estimation in unstable environments, is preferred for its simplicity. The approach employs primary and target Q networks for each layer to accurately estimate Q function values. The primary network calculates the actual Q-value, while the target network provides target Q-values. The RL agent minimizes the difference between these Q-values using backpropagation and gradient descent, guided by a loss function based on mean square error (MSE):

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

Here,  $Q(\sigma, a, \theta)$  is derived from primary network parameters  $\theta$ , and  $r + \gamma \max_{\hat{a}} Q_t(\sigma', \hat{a}, \theta')$  is the target Q-value, calculated using target network parameters  $\theta'$ .

The proposed HRL framework employs two RL agents that utilize DQN to optimize IoT data processing for  $P_1$ ,  $P_2$ , and  $P_3$ , focusing on computing node selection and buffering policies. The agent for  $P_1$  and  $P_2$  evaluates the environment state by observing the distribution of the devices and neighboring satellite nodes at each input node, which guides its routing policies and the choice of computing nodes. For  $P_3$ , the agent considers the chosen edge node, device assignment, and processing load to evaluate the edge processor and storage buffer status, informing its task allocation policies. The intrinsic reward  $\mathcal{R}^3$  is applied to refine the policies of both agents  $P_2$  and  $P_3$ .

#### D. Benchmark Solutions

The following benchmark solutions have been taken into account.

1) *Nearest Satellite Processing Approach*: This method considers input satellite nodes as defined by the policies in  $\mathbf{A}_0$  to handle the IoT data. Devices are statically assigned to buffers if processing units are unavailable. Although this static method reduces data communication costs by processing data near the site, it does not utilize distributed edge resources.

2) *Load Balancing Approach with Static Node Selection Policies*: For comparison, we developed a load-balancing solution. We used the traditional Dijkstra algorithm [8] to solve  $P_1$ , or routing, by identifying the best path between the satellite node and the terrestrial cloud through a satellite relay. Subsequently, devices are allocated to satellite nodes sequentially, filling each node up to capacity before moving to the next. Buffer sizes depend on the processing unit and buffer availability, enabling local distributed computing but potentially restricting performance.

3) *Load Balancing Approach with Random Node Selection Policies*: Here, IoT devices are assigned to satellites in randomly chosen paths. The buffering policies remain static, based on the availability of the processing unit and buffer space.

## IV. PERFORMANCE ANALYSIS

In the following, we analyze the performance of a proposed HRL solution and other benchmark approaches over a Python-based simulation environment. In order to reduce the complexity, we consider a portion of the space characterized by 15 satellites distributed in 3 orbital planes (i.e., 5 satellites per plane) with integrated EC facilities, where each satellite can communicate with up to 4 nearby satellites (two of the same plane and two of the adjacent planes). A variable number of IoT devices (i.e., 500 to 3000) is considered. The probability of an IoT device being active and requesting computing services is considered 0.1. Each IoT device generates a computation task of size  $D_{\rho_d} \sim \text{Uniform}(1, 6)$  MB which is expected to be processed over space computing facilities. The other parameters include  $\Omega_{\rho_d} = 625$ ,  $D_{\rho_d}^r = D_{\rho_d}/5$  and  $\gamma_1, \gamma_2 = 0.5$ . The processing capacity of the LEO satellites is 0.25 GFLOPs. For DQN simulation, primary and target networks with five layers are considered with 50 training iterations.

Fig. 2 illustrates the total cost in terms of latency and energy to process IoT data in space. For the latency, the HRL method outperforms other solutions with appropriate offloading policies. Its multi-hop approach effectively utilizes distributed satellite resources. Nearest Satellite and Load-Balancing incur high costs due to static policies. Although the random method surpasses benchmark solutions, its performance is unpredictable and may suffer from an uneven IoT data distribution in space. With regard to energy values, HRL performs better than load-balancing and random allocation approaches. The improved performance of the nearest satellite method is largely due to many tasks in processing buffers, but it can result in significantly higher computational latency.

Fig. 3 shows the latency communication and computation to process IoT data in space. This involves sending IoT data to space and relaying them through a satellite to a chosen computing node. The proposed HRL scheme slightly increases communication latency but leverages distributed computing resources in space, unlike other methods that reduce latency but cannot do so. The proposed HRL method efficiently utilizes distributed computing in space to decrease computation latency, although it incurs a slightly increased communication cost.

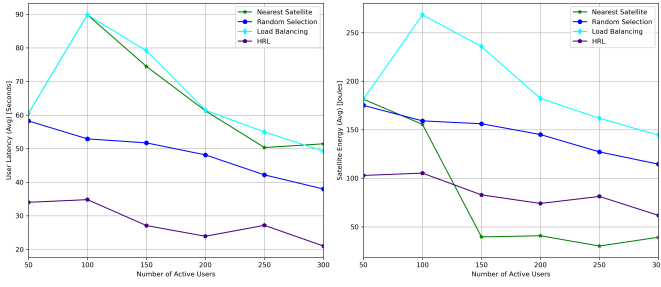


Fig. 2. Average Latency and Energy Cost.

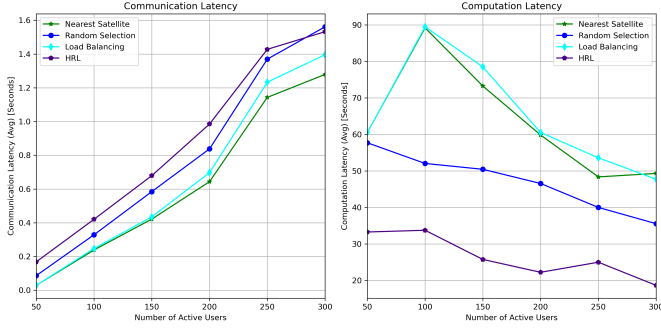


Fig. 3. Average Latency Cost.

The nearest node, load balancing, and random offloading face challenges in efficiently managing IoT data, leading to higher computation latency than the HRL method.

Fig. 4 illustrates the average energy costs of communication and computation. The proposed HRL scheme, similar to its latency profile, experiences higher communication latency while leveraging distributed computing resources in space. Other approaches, with restricted IoT data communication in space, see a reduction in communication costs. The HRL scheme's average computation energy demands are lower compared to load balancing and random allocation strategies. In particular, with nearest-node, only a limited number of satellite nodes handle the tasks, resulting in large IoT data buffers and lower processing energy requirements.

Fig. 5 illustrates the utilization of processing buffer capacity in various methods. The efficient use of processing buffers is crucial for minimizing processing costs. Random allocation and closest processing strategies do not adequately distribute IoT data, leading to a surge in IoT tasks occupying the

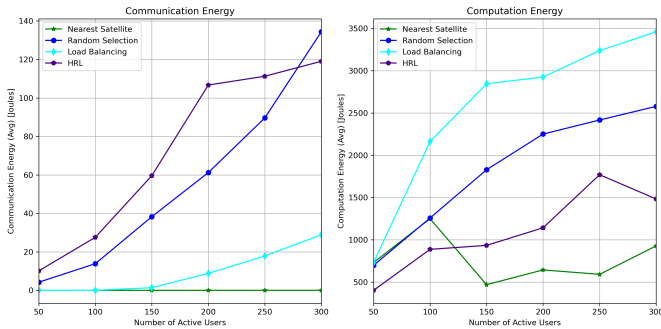


Fig. 4. Average Energy Cost.

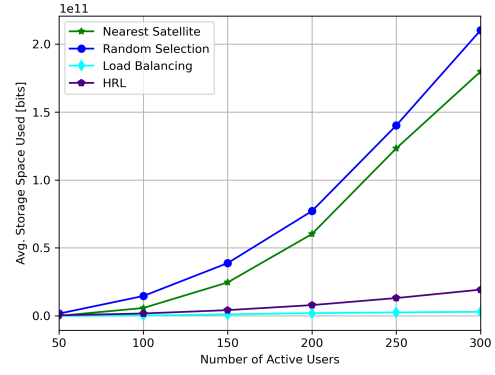


Fig. 5. Utilization of Processing Buffer Space.

processing buffer, thus requiring more buffer space. In contrast, the load-balancing strategy with static buffering policies fails to efficiently utilize the processing buffer space. The proposed HRL approach, which incorporates effective offload and buffering policies, successfully adjusts buffer usage in response to processing demands.

## V. CONCLUSION

This paper focuses on optimizing IoT data processing in space through LEO satellites. A constrained optimization problem was introduced to minimize energy and latency in processing space data. Our DQN-based HRL framework greatly enhanced data routing, node selection, and buffering compared to traditional methods. Simulations reveal the HRL framework surpasses current methods in latency and energy efficiency. Utilizing distributed EC in space, our approach is scalable and efficient for processing IoT data, especially in remote, resource-constrained areas. Future research will refine the HRL framework, explore more optimization strategies, and expand to complex satellite networks. We also plan to study integration with 6G networks and advanced machine learning to enhance in-space IoT computing performance and applicability.

## REFERENCES

- [1] O. Aouedi *et al.*, "A survey on intelligent Internet of Things: Applications, security, privacy, and future directions," *IEEE Commun. Surveys Tuts.*, 2024, early access, doi:10.1109/COMST.2024.3430368.
- [2] S. S. Shinde and D. Tarchi, "Hierarchical reinforcement learning for multi-layer multi-service non-terrestrial vehicular edge computing," *IEEE Trans. Mach. Learn. Commun. Netw.*, vol. 2, pp. 1045–1061, 2024.
- [3] S. S. Shinde and D. Tarchi, "Towards a novel air-ground intelligent platform for vehicular networks: Technologies, scenarios, and challenges," *Smart Cities*, vol. 4, no. 4, pp. 1469–1495, 2021.
- [4] S. Mahboob and L. Liu, "Revolutionizing future connectivity: A contemporary survey on AI-empowered satellite-based non-terrestrial networks in 6G," *IEEE Commun. Surveys Tuts.*, vol. 26, no. 2, pp. 1279–1321, 2024.
- [5] R. Xie, Q. Tang, Q. Wang, X. Liu, F. R. Yu, and T. Huang, "Satellite-terrestrial integrated edge computing networks: Architecture, challenges, and open issues," *IEEE Netw.*, vol. 34, no. 3, pp. 224–231, 2020.
- [6] C. Li, Y. Zhang, R. Xie, X. Hao, and T. Huang, "Integrating edge computing into Low Earth Orbit satellite networks: Architecture and prototype," *IEEE Access*, vol. 9, pp. 39 126–39 137, 2021.
- [7] S. S. Shinde and D. Tarchi, "Integrated aerial-ground computation offloading for dependency-aware IoV multitask services," in *2024 IEEE International Mediterranean Conference on Communications and Networking (MediCom)*, 2024, pp. 317–322.
- [8] J. A. Bondy and U. S. R. Murty, *Graph theory*, ser. Graduate Texts in Mathematics. Springer London, 2008.