Service Function Chaining in LEO Satellite Networks via Multi-Agent Reinforcement Learning

Khai Doan*, Marios Avgeris*, Aris Leivadeas†, Ioannis Lambadaris*, Wonjae Shin‡
*Carleton University, Department of Systems and Computer Engineering, Ottawa, ON, Canada.
†École de Technologie Supérieure, Department of Software and IT Engineering, Montreal, QC, Canada.
‡Ajou University, Department of Electrical and Computer Engineering, Suwon, South Korea.
Email: {khaidoan@sce, mariosavgeris@cunet, ioannis@sce}.carleton.ca, aris.leivadeas@etsmtl.ca, wjshin@ajou.ac.kr

***Abstract*—In addition to offering enhanced global connectivity, low-earth-orbit satellite networks (LSNs) can be a potential solution for a large range of applications such as disaster response, environmental monitoring, and military operations, among others. In our context, each specific application is represented by a service function chain (SFC) in which each function is considered as a task in the application. Our objective is to optimize the long-term system performance by minimizing the average end-to-end delay of SFC deployments in LSNs. To achieve this, we formulate a dynamic programming (DP) problem to derive an optimal placement policy. To overcome the computational intractability, the need for statistical knowledge of SFC requests, and centralized decision-making challenges, we present a multi-agent Q-learning approach where satellites act as independent agents. To facilitate performance convergence in non-stationary agents' environments, we let agents to collaborate by sharing designated learning parameters. In addition, agents update their Q-tables via two distinct rules depending on selected actions. Extensive experimentation shows that our approach achieves convergence and performance relatively close to the optimum obtained by solving the formulated DP equation.**

***Index Terms*—Network Function Virtualization, Service Function Chaining, Satellite Networks, Multi-Agent Reinforcement Learning.**

I. Introduction

Satellite communication networks (SCN), or specifically, low-earth-orbit satellite networks (LSNs) have emerged as a popular solution for providing global connectivity [1]. Key technologies driving this progress include Network Function Virtualization (NFV) and Service Function Chaining (SFC) [2]. In addition, this type of network can also provide a wide range of applications. For example, it can be used for surveillance, tracking, and mapping, as well as for weather forecasting and disaster response [3], while it can support remote sensing and earth observation. Decomposing application tasks into a sequence of functions, also called Virtualized Network Functions (VNFs), that resembles the SFC paradigm for deployment in LSNs is a promising approach.

Many recent studies have focused on integrating NFV and service function chaining concepts into LSNs. Gao et al. [4] formulated it as an integer non-linear programming problem and proposed a distributed heuristic that utilizes neighboring satellite resources for SFC placement. In [5] and [6], two game-theoretic formulations were introduced, and a Nash equilibrium was found using potential game-based algorithms. However, none of these works addressed the dynamic nature of satellite communication networks. Conversely, the following works used a time-varying system model to address satellite SFC placement; in [7], the authors formulated a time-slotted integer linear programming (ILP) problem and developed two heuristic-based algorithms which minimized energy consumption. Jia et al. [8] explored joint SFC placement and routing optimization through elastic resource provisioning. They then solved the corresponding "multi-slot" ILP problem while minimizing resource consumption. The authors in [9] proposed an ILP formulation and a heuristic solution to minimize end-to-end delay in the SCN. However, despite their time-varying modeling, these works ignored long-term performance optimization and focused only on short-term system rewards.

This work outlines a methodology for SFC placement in LSNs. In our context, the entire SFC represents a particular application and each function is treated as a task within. Our study distinguishes itself from previous works by formulating the problem as a discrete-time stochastic control process and aiming to optimize long-term system performance, while accounting for the dynamic nature of the network. Our contribution is threefold:

1) We first devise a dynamic programming (DP)-based solution towards the optimal SFC placement policy. Although optimal, this solution is computationally complex and requires some unrealistic assumptions.
2) We then come up with a multi-agent Q-learning (MAQL) approach where satellites behave as independent agents. We address the convergence challenge caused by non-stationary environments by: (i) allowing satellites to share designated learning parameters and (ii) letting satellites update their Q-tables via distinct rules based on the selected actions.
3) Through extensive simulation we demonstrate convergence and efficiency, with a performance comparable to the optimum DP solution.

The rest of this work is organized as follows: Section II describes the system model. Section III provides the DP formulation. Section IV presents the MAQL-base service placement policy. Experimental results are provided in

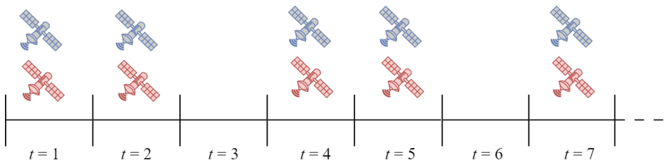


Fig. 1: An example of two satellites' mobility modeling, where the presence of both in a timeslot signifies the ISL availability: $e_{vu}(1) = 1, \tau_{vu} = 2, T_{vu} = 3$.

Section V with two different parameter settings considered. A conclusion of our work is given in Section VI.

II. System Model

A. Communication Model

We consider an LSN with a set of satellites, $\mathbb{V} = \{1, \ldots, V\}$, orbiting around the Earth. The satellites gather data (e.g., images of Earth's surface) and cooperatively execute computational tasks for some designated applications (e.g., flooding and wildfire detection, or military support and weather forecasting [3]). The movements of satellites are periodic and we assume that an inter-satellite link (ISL) between a pair of satellites is available when the two satellites enter the communication range of each other. Each satellite v is equipped with R_v computation resource (CR), i.e., CPU and memory, and Z_v storage resource (SR) units.

Our system operates in a slotted, infinite time horizon, where $t \in \mathbb{N}_0 = \{1, 2, \ldots\}$ denotes the timeslot indices. We represent the availability of the ISL between v and u, hereafter referred to as $v - u$, at timeslot t with a binary parameter $e_{vu}(t)$, where $e_{vu}(t) = 1$ if the link is available and 0 otherwise. Also, we use τ_{vu} to denote the duration of the $v - u$. Since the movements of the satellites are periodic, the availability of ISLs is also periodic. Therefore, let T_{vu} denote the period of the $v - u$, i.e., $T_{vu} \geq \tau_{vu}$ and $e_{vu}(t) = e_{vu}(t + T_{vu})$. Then, the period T of the entire network can be defined as the least common multiple of $T_{uv}, \forall u, v \in \mathbb{V}$. We assume that initial conditions $e_{vu}(1)$ are known for all $v - u$ links. This assumption and the definitions of $e_{vu}(t), \tau_{vu}$, and T_{vu} comprise the satellite mobility modeling (Fig. 1). We assume that data transferring between a pair of satellites is able to complete in a single timeslot.

B. Service Function Chaining and Requests

The data collected by each satellite v is saved in its dedicated database. The Z_v units of SR mentioned earlier are meant for storing data generated during the execution of an application, the details of this process will be given later. For simplicity, we assume that this storage is separated from the resources used for the database. We denote by $\mathbb{H} = \{1, \ldots, H\}$ the index set of available SFCs. Each SFC h represents an application and is comprised by an ordered sequence of l_h functions. Each satellite v comes with a pre-installed set of function images for which it can execute. Specifically, we denote by F_f^h the f^{th} function of SFC h, and $\mathbb{F}_v = \{F_f^h | h \in \mathbb{H}, f = 1, \ldots, l_h\}$ is the set of functions that v is capable of executing. The input to each SFC h, i.e., to F_1^h, comes from a satellite's database. It is assumed that the databases of all satellites are identical and can deliver input to F_1^h for every SFC h. The output of F_f^h becomes the input of F_{f+1}^h, and the output of $F_{l_h}^h$ is the desired result.

When a satellite needs the collected data to be processed on an SFC, it initiates a *service request*. We assume that there is at most one service request in each timeslot whose occurrence probability is denoted by μ. The satellite that initiates the request is called the *requester*. Provided that there is a service request, we let μ_v^r and μ_h^s be the probability that satellite v will be the requester and h will be requested SFC, respectively.

Considering an SFC h, we denote by q_f^h and g_f^h the number of CR and SR units required to execute F_f^h and to store the output of F_f^h, respectively. To execute F_1^h, a satellite will load the input data from its database to memory and process. If a satellite executes a sequence of $l < l_h$ consecutive functions $F_f^h, f = 1, \ldots, l$, the data will be continuously processed in memory, hence, SR is not consumed. If a satellite v completes F_f^h and forwards the results to u instead of deploying F_{f+1}^h, the output of F_f^h will be moved from the memory to the storage for transferring. Thus, g_f^h units of satellite v's SR are occupied until the transfer completion. The following are transferred from satellite v to u: the request (which is assumed not to cost any resources), and the output of F_f^h. Hence, g_f^h units of satellite u's SR will be occupied upon receiving the data, and the transfer cannot be done if the available SR of u is less than g_f^h. In the case when none of the functions has been executed, only the request will be transferred, and u will get the initial input from its own database. We also define d_{fv}^h as the running time of F_f^h at satellite v, and d^h as the maximum end-to-end delay tolerance of SFC h.

III. Dynamic Programming Formulation

In this section, we present the dynamic programming formulation that leads to an optimal service placement policy. In our model, the system state denoted by $\mathbf{x}$ is defined at the beginning of a timeslot, followed by a placement denoted by $\mathbf{p}$, and finally by the service request. For the logical flow, we first provide the definition of the service placement.

1) Service placement: let $\mathbf{p} = (\mathbf{u}, \mathbf{m})$ denote a service placement, where $\mathbf{u}$ is defined as

$$\mathbf{u} = \{u_k | u_k \in \mathbb{V}, k \in \mathbb{N}_0\}, \tag{1}$$

with u_k being the index of the satellite that handles the service in the k^{th} timeslot with $k = 1$ is the current timeslot. Thus, the number of elements in $\mathbf{u}$, i.e., $|\mathbf{u}|$, is the end-to-end delay (in timeslots) of placement $\mathbf{p}$. Intuitively, the first element of $\mathbf{u}$ is the requester's index. Next, $\mathbf{m}$ is defined as

$$\mathbf{m} = \{m_f | m_f \in \mathbb{N}_0, f \in [1, \ldots, l_h]\}, \tag{2}$$

where m_f is the timeslot that the F^h_f is deployed with $m_f = 1$ being the current time slot. We also assume that $\mathbf{u} = \mathbf{m} = \emptyset$ represents the blocking of requests (requests are rejected from being served). For intuition, we provide the following example:

Example 1: Satellite v requests SFC h comprised of two functions, and a possible placement $\mathbf{p}$ is described as follows:

- $t = 1$, v executes F^h_1.
- $t = 2$, v forwards the output to u.
- $t = 3$, u receives the data and executes F^h_2. [1]
- $t = 4$, u forwards the results back to v - the requester.

This placement $\mathbf{p}$ spans 4 timeslots, including the current one, and $\mathbf{u} = (v, v, u, u)$. The two functions are deployed at $t = 1$ and $t = 3$, respectively, hence $\mathbf{m} = (1, 3)$.

We note that when a placement $\mathbf{p}$ is made at the current timeslot, the required satellite resources in future timeslots are reserved accordingly. However, the end-to-end delay of every service placement cannot exceed a Service-Level Agreement (SLA) value $K = \max_{h \in \mathbb{H}} d^h$, which defines the maximum delay tolerance for all SFCs. This suggests that satellites' resources beyond K timeslots from the current one have to be available.

2) System state: a system state carries information regarding the available resources at every satellite in the network, and the request that needs to be served. To define the system states, let us first define the resource availability for a satellite $v \in \mathbb{V}$ by $\mathbf{x}_v = (\mathbf{r}_v, \mathbf{z}_v)$, where $\mathbf{r}_v = \{r_{vk} | r_{vk} \leq R_v, \forall k \in [1, ..., K]\}$ denotes the available CR and $\mathbf{z}_v = \{z_{vk} | z_{vk} \leq Z_v, \forall k \in [1, ..., K]\}$ the available SR for the K consecutive timeslots, respectively. Then, the system state at the current time slot would be:

$$\mathbf{x} = \{\mathbf{x}_v, h, n | \forall v \in \mathbb{V}, h \in \mathbb{H}, n \in \mathbb{V}\}, \tag{3}$$

where h and n are indices of the requested SFC and the requester satellite in the previous timeslot, respectively, with $h = n = 0$ if there is no service request in the earlier slot. Given a current placement $\mathbf{p} = (\mathbf{u}, \mathbf{m})$, we let $\tilde{\mathbf{x}} = \{\tilde{\mathbf{x}}_v, \tilde{h}, \tilde{n} | \forall \tilde{v} \in \mathbb{V}, \tilde{h} \in \mathbb{H}, n \in \mathbb{V}\}$ be the transitioned state in the next timeslot where $\tilde{\mathbf{x}}_v = (\tilde{\mathbf{r}}_v, \tilde{\mathbf{z}}_v)$ is the corresponding resource availability. $\tilde{\mathbf{x}}_v$ is defined as follows: let $\kappa = 0$ if $m_1 = 1$. Otherwise, let κ be the smallest index such that $u_\kappa \neq u_{\kappa+1}$. This means that $u_{\kappa+1}$ is the index of the first satellite that consumes either its CR or SR according to the considered placement $\mathbf{p}$. Then (i) if $v = u_{m_f} \in \mathbf{u}$ and $m_f \in \mathbf{m}$ where $m_f \geq \kappa + 1$: this implies that v executes F^h_f, hence, the data are processed at v until F^h_f finishes running, i.e., $v = u_{m_f + m}, \forall m = 0, \ldots, d^h_{kv} - 1$. Then, $\tilde{r}_{vk} = r_{v(k+1)} - q^h_f$, $\tilde{z}_{vk} = z_{v(k+1)}$, $k = m_f, \ldots, m_f + d^h_{fv} - 1$. (ii) if (i) does not hold and the following holds: $v = u_m \in \mathbf{u}$ and $m \notin \mathbf{m}$ and $\exists m_f, m_{f+1} \in \mathbf{m}$ such that $m_f < m < m_{f+1}$, this implies that v is storing the output of F^h_f. Hence, $\tilde{r}_{v(m-1)} = r_{vm}$, $\tilde{z}_{v(m-1)} = z_{vm} - g^h_f$. We note that the condition $m_f < m$ with $m_f \in \mathbf{m}$ implies $m \geq 2$. (iii) if neither (i) nor (ii) hold, i.e., v is not involved in the placement, then $\tilde{r}_{vk} = r_{v(k+1)}, \tilde{z}_{vk} = z_{v(k+1)}$ for $k = 1, \ldots, K - 1$.

Moreover, $\mathbf{p}$ cannot span $K+1$ timeslots. Therefore, the K^{th} component of $\tilde{\mathbf{x}}_v$ must be $\tilde{r}_{vK} = R_v, \tilde{z}_{vK} = Z_v$. The presented cases cover all the possibilities of $\tilde{r}_{vk}$, $\tilde{z}_{vk}$, $\forall v \in \mathbb{V}, k = 1, \ldots, K$. In addition, $\tilde{h}$ and $\tilde{v}$ are the requested SFC and requester satellite in the current time slot. The transition probability from $\mathbf{x}$ to $\tilde{\mathbf{x}}$, given a placement $\mathbf{p}$, is defined as:

$$\mathcal{P}\left\{\mathbf{x} \underset{\mathbf{p}}{\rightarrow} \tilde{\mathbf{x}}\right\} = \begin{cases} 1 - \mu, & \text{if no request,} \\ \mu \times \mu^r_{\tilde{v}} \times \mu^s_{\tilde{h}}, & \text{otherwise.} \end{cases} \tag{4}$$

In this formulation, we define a policy as a rule that maps each state to a placement in every timeslot.

3) Placement cost: let us now define $\mathcal{C}(\mathbf{x}, \mathbf{p})$ as the cost for placement $\mathbf{p} = (\mathbf{u}, \mathbf{m})$ with respect to state $\mathbf{x}$, as follows:

$$\mathcal{C}(\mathbf{x}, \mathbf{p}) = \begin{cases} |\mathbf{u}|, & \text{if the request is served,} \\ C_p \gg 0, & \text{if the request is blocked,} \end{cases} \tag{5}$$

where C_p is a predefined penalty cost.

4) Dynamic programming equation: to define a long-term, time-depended cost, we assign a timeslot index t to the state and placement definitions: $\mathbf{x}(t)$ and $\mathbf{p}(t)$. This allows for defining the long-term discounted cost as: $\sum_{t=1}^{\infty} \gamma^t \mathbb{E}\left[\mathcal{C}\left(\mathbf{x}(t), \mathbf{p}(t)\right)\right]$, where $\gamma \in [0, 1]$ is the discount factor. The goal of this work is to determine a placement policy that minimizes this cost. The minimum long-term discounted cost, denoted by $J(\mathbf{x})$, can be expressed by the following DP Equation, the solution of which provides us with the optimal placements:

$$J(\mathbf{x}) = \min_{\mathbf{p}} \left\{\mathcal{C}(\mathbf{x}, \mathbf{p}) + \gamma \mathcal{P}\left\{\mathbf{x} \underset{\mathbf{p}}{\rightarrow} \tilde{\mathbf{x}}\right\} J(\tilde{\mathbf{x}})\right\}. \tag{6}$$

IV. Learning-Based Service Placement

Due to its recursive nature, solving the DP Eq. (6) can be computationally intractable for large inputs. Furthermore, this equation requires the statistics of service requests, i.e., $\mu^r_v, \mu^s_h\ \forall v, h$, which are often unavailable. To address these issues, we re-formulate the discussed problem in a MAQL context where satellites act as independent agents and learn the optimal placement policy via observations of stochastic events. In a given timeslot, there can be multiple requests handled by a satellite v including those forwarded from other satellites and/or the ones initiated by v itself. Satellite v buffers those requests and takes an action on all of them within the given timeslot. The concept of states, actions, and costs are defined as follows:

[1] In Section II, the data transferring has been assumed to span 1 timeslot. This assumption does not restrict the presented 2 methods.

1) State: the available resources and buffered requests of a satellite v are conveyed in its perceptual state which is represented by $\mathbf{s}_v = \{r_v, z_v, \mathbf{y}_i | \forall i \in [1,\dots,b]\}$, where b is the number of requests buffered in v at the considered timeslot; r_v and z_v are the available CR and SR, respectively; $\mathbf{y}_{vi}$ denotes the i^{th} buffered request and is defined as $\mathbf{y}_i = \left(h_i, n_i, f_i, \hat{t}_i\right)$, in which $h_i \in \mathbb{H}$ and $n_i \in \mathbb{V}$ denote the requested SFC and requester satellite, respectively; index $\hat{t}_i$ is the timeslot when the request is initiated and we assume that $\hat{t}_i < \hat{t}_{i+1}$ $\forall i$; f_i denotes the order of the function in SFC h_i that needs to be deployed next. For example, in the current timeslot, if $l < l_{h_i}$ consecutive functions $F^h_f, f = 1,\dots,l$ have been executed, either by other satellites before forwarding to v or by v itself, then, $f_i = l+1$. If none among the l_{h_i} functions have been deployed, $f_i = 1$. If all l_{h_i} functions have been deployed, $f_i = l_{h_i} + 1$, implying that the final result has been obtained, and needs to be returned to the requester.

2) Action: an action for a satellite v is defined as $\mathbf{a} = \{a_i | a_i \in \{1,\dots,V+2\}, \forall i \in [1,\dots,b]\}$, where a_i is the placement decision regarding request $\mathbf{y}_i$. As presented in the previous section, the requests are handled in a first-come-first-served manner meaning that decision a_i is taken before a_{i+1}. Let us denote as r_{vi} and z_{vi} the available CR and SR after actions $a_j, j = 1,\dots,i$ have been taken, respectively. We identify the following cases:

- $a_i = u \in \{1,\dots,V\} \setminus \{v\}$: forward the request to satellite u. This action is valid if $v-u$ ISL is available at the current timeslot.
- $a_i = V+1$: deploy $F^{h_i}_{f_i}$. This action is valid if the following three conditions are satisfied: (i) not all the functions of SFC h_i have been deployed, i.e., $f_i \le l_{h_i}$; (ii) the function is installed at satellite v, i.e., $F^{h_i}_{f_i} \in \mathbb{F}_v$; (iii) the CR is sufficient, i.e., $r_{v(i-1)} \ge q^{h_i}_{f_i}$.
- $a_i = V+2$: block the request.
- $a_i = v$: carry the request to the next timeslot.

Let $\mathbb{A}_v(\mathbf{y}_i)$ be the set of valid actions of v for request $\mathbf{y}_i$. In this MAQL setup, we determine a policy, denoted by $\pi(\mathbf{y}_i) \to a_i$, as a rule that maps each request $\mathbf{y}_i$ to an action $a_i \in \mathbb{A}_v(\mathbf{y}_i)$.

3) Instant Cost: for a pair of request $\mathbf{y} = \left(h, n, f, \hat{t}\right)$ and action a, satellite v receives a cost $c(\mathbf{y}, a)$ as follows:

- if $a = u \in \{1,\dots,V\} \setminus \{v\}$, then if u has sufficient SR to receive the request, the cost is equal to the transferring time (1 timeslot). Otherwise, a penalty C_p is incurred. Therefore, $c(\mathbf{y}, u) = \mathbb{1}\left\{z_u \ge g^h_f\right\} + C_p \mathbb{1}\left\{z_u < g^h_f\right\}$, where $\mathbb{1}\{\cdot\}$ is the indicator function.
- if $a = V+1$, then the cost is equal to the function's execution time, $c_v(\mathbf{y}, V+1) = d^{h_i}_{f_i v}$.
- if $a = V+2$, then a blocking penalty is applied, $c_v(\mathbf{y}, V+2) = C_p$.
- if $a = v$, then a waiting cost for one timeslot is applied, $c_v(\mathbf{y}, v) = 1$.

Then, $C_v(\mathbf{s}, \mathbf{a}) = \sum_{i=1}^{b} \tilde{\gamma}(\mathbf{y}_i) c_v(\mathbf{y}_i, a_i)$ is defined as the total cost of satellite v for performing action $\mathbf{a}$ on state $\mathbf{s}$ where $\tilde{\gamma}(\mathbf{y}_i)$ is the discount factor associated with request $\mathbf{y}_i$. As presented in Sec. III, the placement for a request in timeslot t is performed in timeslot $t+1$, and is discounted by γ^{t+1}. Therefore, $\tilde{\gamma}(\mathbf{y}_i) = \gamma^{\hat{t}_i + 1}$.

4) Learning mechanism & optimal action estimation: each satellite's goal is to learn its own optimal policy that minimizes the following long-term discounted cost: $\sum_{t=1}^{\infty} \mathbb{E}\left[C_v(\mathbf{s}(t), \mathbf{a}(t))\right] = \sum_{\mathbf{y}(t)\in\mathbf{s}(t)} \Psi_{\pi_v}(\mathbf{y}(t))$ where $\Psi_{\pi_v}(\mathbf{y}(t)) = \sum_{t=1}^{\infty} \mathbb{E}_{\pi_v}\left[\tilde{\gamma}(\mathbf{y}(t)) c_v(\mathbf{y}(t), a(t))\right]$ and $\pi_v(\mathbf{y}(t)) \to a(t)$ is the employed policy. To approximate the optimal policy, v approximates the optimal action for every request in each time slot. This is done by estimating the cost $\Psi_{\pi_v}(\mathbf{y}(t))$ for every given request $\mathbf{y}(t)$. Let $\mathbb{Q}_v$ be the Q-table of v storing all the learned Q-values. According to our action definition, a policy depends on the available resources and ISLs at the considered timeslot. Since the network topology varies deterministically and periodically, the availability of ISLs can be determined by τ, a time slot index relative to the system period T. Specifically, a relative index τ with respect to a given timeslot t can be computed as: $\tau = t\mathbb{1}\{t < T\} + \mathrm{mod}(t, T)\mathbb{1}\{t \ge T\}$ where $\mathrm{mod}(t, T)$ returns the remainder of $t \div T$. To this end, we denote by $Q^{(\tau)}_v(\mathbf{y}(t), a(t))$ the Q-value that estimate the cost $\Psi_{\pi_v}(\mathbf{y}(t))$. For our discussion presented hereafter, we will consider the time slot at the present moment and remove the index t for simplicity.

In the proposed MAQL mechanism, the environment of each agent/satellite is non-stationary as it is affected by the policies employed by other agents. What is more, these policies keep updating, therefore, convergence is not guaranteed. To tackle this issue, we promote multi-agent cooperation among satellites in the form of task transferring; that is when satellite $v \in \mathbb{V}$ transfers a request $\mathbf{y}$ to satellite $u \in \mathbb{V} \setminus \{v\}$, u shares its knowledge of the Q-table as follows:

- if the request $\mathbf{y}$ has been encountered by satellite u with some action a applied, i.e., $Q^{(\tau)}_u(\mathbf{y}, a)$ can be found in $\mathbb{Q}_u$, then $\min\limits_{a} Q^{(\tau)}_u(\mathbf{y}, a)$ is shared with satellite v which indicates the minimum discounted cost that u can achieve for request $\mathbf{y}$.
- otherwise, Q_{init} is shared which is a constant predefined initial Q-value.

Having this information available, satellite v performs an iterative update using the Bellman Equation, $Q^{(\tau)}_v(\mathbf{y}, a) \leftarrow (1-\alpha) Q^{(\tau)}_v(\mathbf{y}, a) + \alpha\left(c(\mathbf{y}, a) + \gamma_B \hat{Q}\right)$, where:

$$\hat{Q} = \begin{cases} \underset{a' \in \mathbb{A}_v(\mathbf{y}')}{min} \; Q^{(\tau)}_v(\mathbf{y}', a'), & \text{if } a = v, \text{ or } a = V+1, \\ \underset{a' \in \mathbb{A}_u(\mathbf{y}')}{min} \; Q^{(\tau)}_u(\mathbf{y}', a'), & \text{if } a = u \in \{1,\dots,V\} \setminus \{v\}. \end{cases}$$

Here $\alpha, \gamma_B \in [0, 1]$ are the learning rate and the discount factor of the iterative update step, respectively. Given an action a, we define $\mathbf{y}'$, a request transitioning from $\mathbf{y} =$

$\left(h, n, f, \hat{t}\right)$ in the next timeslot, by:

$$\mathbf{y}' = \begin{cases} \left(h, n, f, \hat{t}\right), & \text{if } a = 1, 2, \ldots, V, \\ \left(h, v, f+1, \hat{t}\right), & \text{if } a = V+1. \end{cases} \tag{7}$$

Blocking a request results in a penalty C_p, and the request is permanently removed from the system. Therefore, $Q_v^{(\tau)}\left(\mathbf{y}, V+2\right) = C_p$ always. Subsequently, satellite v estimates its optimal action, a^*, by:

$$a^* = \underset{a \in \mathbb{A}_v(\mathbf{y})}{min} \; Q_v^{(\tau)}\left(\mathbf{y}, a\right). \tag{8}$$

V. Numerical Results

In this section, we demonstrate the achieved convergence and performance of the proposed method through four experiments. We consider a system with $V = 3$ satellites, $H = 2$ types of SFC, $\gamma = 0.6$ as the discount factor, and $e_{vu}(1) = 1, \forall v, u \in \mathbb{V}$ initially. The rest of the parameters are set as outlined in Table I, where SFC 1 consists of two functions in both setups; SFC 2 consists of 3 functions in Setup 1 and 2 functions in Setup 2. The pre-installed functions at satellites are as follows: for Setup 1, $\mathbb{F}_1 = \{F_1^1, F_1^2, F_2^2\}$, $\mathbb{F}_2 = \{F_1^1, F_2^1, F_1^2, F_3^2\}$, $\mathbb{F}_3 = \{F_2^1, F_2^2, F_3^2\}$; for Setup 2, $\mathbb{F}_1 = \{F_1^2\}$, $\mathbb{F}_2 = \{F_1^1\}$, $\mathbb{F}_3 = \{F_2^1, F_2^2\}$. In our experiments, training and testing are conducted alternatively, with the procedures being identical, except that satellites select actions uniformly randomly during training, and calculate their optimal actions using Eq. (8) during testing. The testing is performed periodically after every 10^3 training timeslots in the 1st experiment, and 10^5 training timeslots in the other experiments. In the 2nd and 4th experiments, the optimal solution is attained via the DP Equation (6).

We use two metrics to evaluate the system performance - the average cost and request serving rate. In order to calculate the average cost during the testing phase, we repeatedly calculate the discounted cost over 100 timeslots for 10^3 trials, and then, compute the average value. Additionally, the serving rate is determined by dividing the number of fulfilled requests by the total requests. We select a variable learning rate via fine-tuning: $\alpha = 0.1$ initially; $\alpha = 0.01$ if we detect 10 consecutive tests without improvement in the average cost.

Experiment 1: This experiment uses Setup 1 and demonstrates that satellites are able to cooperatively learn an optimal placement for a given SFC. This experiment is conducted as follows: the service request happens at the initial timeslot, $t = 1$, with probability 1. The designated requester and requested SFC are satellite 1 and SFC 2, respectively. No further service requests occur in subsequent timeslots until the current request has been fulfilled, blocked or expired. Then, the same service request occurs again, and so on. The resulting cost is the end-to-end delay of the placement cooperatively deployed by satellites, with the penalty cost C_p added if the request expires or is blocked. The results are presented in Fig. 2a where the solutions found respectively are: blocking the request which results in a penalty $C_p = 100$, a placement with a delay of 9 timeslots, and finally, a placement with a delay of 7 timeslots. The obtained best placement is described by the following process:

- $t = 1, 2$, satellite 1 deploys F_1^2 and F_2^2.
- $t = 3$, satellite 1 sends the result to satellite 2.
- $t = 4-6$, satellite 2 deploys F_3^2 and holds the result.
- $t = 7$, satellite 2 returns the final result to satellite 1.

This corroborates that the described placement is optimal and poses the smallest end-to-end delay of 7 timeslots.

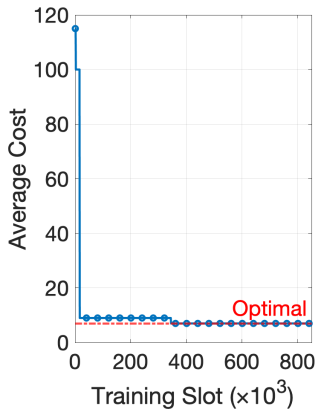
Experiment 2: A general context is considered in this experiment where we allow service requests to occur in every timeslot according to the given probabilities. This experiment aims to demonstrate the achieved convergence in terms of request serving rate for Setup 1 (Fig. 2b) and the average cost for Setup 2 (Fig. 3a). For Setup 1, as there are more functions installed at satellites, the number of possible placements is large, hence, solving the DP Equation is computationally intractable. Therefore, the results of the DP Equation is omitted in Fig. 2b. The two figures suggest that our learning model can converge under both parameter settings. Furthermore, the achieved cost differs from the optimal only by a little, as depicted in Fig. 3a.

Experiment 3: We utilize Setup 1 and examine the influence of the γ_B parameter on the overall rate of request fulfillment of our MAQL model. We recall that γ_B is the discount factor in the Q-table's iterative update step. Our findings, as illustrated in Fig. 2c, reveal that varying γ_B correspond to different serving rates at the performance convergence. The sweet spot for γ_B is around 0.6, according to our parameter setting, with approximately 83% of requests fulfilled. We note that γ_B is involved in the updating of Q-values, and not in the average cost.

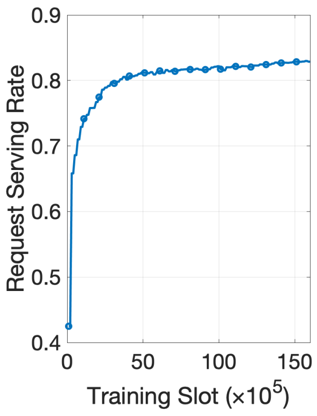
Experiment 4: Setup 2 is used in the fourth experiment where we compare the achieved request serving rate of our MAQL model with two other approaches: a random placement policy where satellites select their actions randomly and the optimal policy from the DP formulation.

TABLE I: System parameters in two setups.

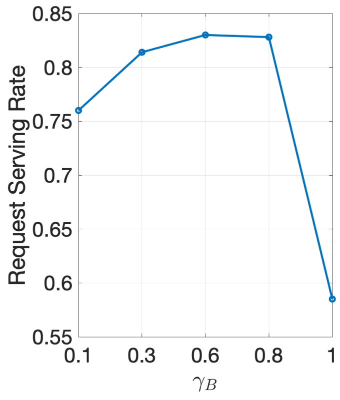
Parameters	Setup 1	Setup 2
$R_v, \forall v$	9	7
$Z_v, \forall v$	18	7
$g_f^1, \forall f$	$3, 1$	$1, 1$
$q_f^1, \forall f$	$5, 4$	$2, 2$
$g_f^2, \forall f$	$1, 3, 1$	$3, 3$
$q_f^2, \forall f$	$3, 4, 4$	$6, 6$
$d_{f1}^h, d_{f2}^h, d_{f3}^h \forall h, f$	1, 2, 2	1, 1, 1
d^1, d^2	15	7
μ	0.9	0.9
$\mu_v^r, \forall v$	1/3	1/3
$\mu_h^s, \forall h$	1/2	1/2
T_{12}, T_{13}, T_{23}	2, 4, 4	2, 2, 2
$\tau_{vu}, v, u = 1, 2, 3$	1	1
C_p	100	100
γ_B	0.6	1.0



(a)

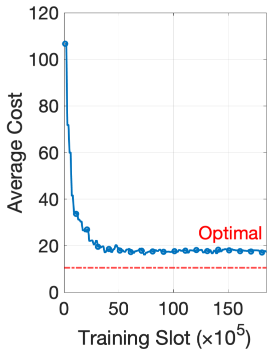


(b)

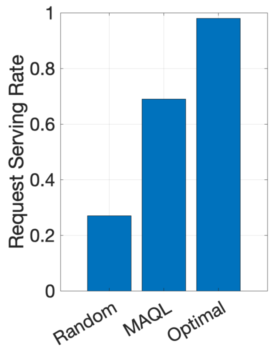


(c)

Fig. 2: Experiments (Setup 1) presenting the achieved convergence and performance in (a) a single-request context, (b) a general context where service request can possibly occur in every timeslot, and (c) the influence from γ_B.



(a)



(b)

Fig. 3: Benchmarking (Setup 2) illustrating the achieved (a) convergence and (b) request serving performance.

In Fig. 2b, although the optimal rate is not presented, the difference between it and the MAQL model is less than 0.2. Compared to Fig. 3b, the difference is bigger, which is a result of the satellites being equipped with more functions in Setup 1 than in Setup 2. Therefore, in the former case we have more possibilities for placements, allowing more requests to be fulfilled.

VI. Conclusion

In this work we tackled the SFC placement problem for LSNs in a discrete-time stochastic control framework and aimed to optimize the long-term system performance. First, we developed an optimal service placement policy by formulating a DP Equation. However, solving the DP Equation has certain drawbacks such as a high computational complexity, the need for knowing the service request probability, and the requirement of a centralized implementation. To overcome the aforementioned challenges, we presented a MAQL approach. Still, due to the non-stationary nature of the satellite environment, convergence is not guaranteed in our context. To address the problem, (i) we promoted satellites to share designated learning parameters for every transferred request, and (ii) we let satellites update their learning parameters according to two distinct rules depending on their selected actions. Experimental results presented to exhibit the achieved convergence and performance in comparison to the optimal solution from solving the DP Equation. For our future work, extending the system model towards reflecting more practical assumptions, such as dealing with a bulk arrival of requests or introducing randomness into the satellites' orbits, are potential directions.

References

[1] 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.

[2] A. Leivadeas, G. Kesidis, M. Ibnkahla, and I. Lambadaris, "VNF placement optimization at the edge and cloud," *Future Internet*, vol. 11, no. 3, p. 69, 2019.

[3] B. Ko and S. Kwak, "Survey of computer vision-based natural disaster warning systems," *Optical Engineering*, vol. 51, pp. 901–936, 2012.

[4] X. Gao, R. Liu, A. Kaushik, J. Thompson, H. Zhang, and Y. Ma, "Dynamic resource management for neighbor-based VNF placement in decentralized satellite networks," in *6GNet*, 2022, pp. 1–5.

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

[6] X. Qin, T. Ma, Z. Tang, X. Zhang, X. Liu, and H. Zhou, "Sfc enabled data delivery for ultra-dense leo satellite-terrestrial integrated network," in *IEEE GLOBECOM*, 2022, pp. 668–673.

[7] Z. Jia, M. Sheng, J. Li, D. Zhou, and Z. Han, "VNF-based service provision in software defined leo satellite networks," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 9, pp. 6139–6153, 2021.

[8] Z. Jia, M. Sheng, J. Li, R. Liu, K. Guo, Y. Wang, D. Chen, and R. Ding, "Joint optimization of VNF deployment and routing in software defined satellite networks," in *VTC*, 2018, pp. 1–5.

[9] Y. Cai, Y. Wang, X. Zhong, W. Li, X. Qiu, and S. Guo, "An approach to deploy service function chains in satellite networks," in *IEEE/IFIP NOMS*, 2018, pp. 1–7.