Deep Reinforcement Learning-based Service Function Chains (SFCs) Deployment

Xiao Kong[1] and Limei Peng[1]
[1]School of Computer Science and Engineering, Kyungpook National University,
Daegu, South Korea

Network Function Virtualization (NFV) on top of Mobile Edge Computing (MEC) architecture has gained significant attention for enabling efficient IoT service provisioning. NFV enhances resource utilization by deploying Virtual Network Functions (VNFs) on general-purpose servers, which are organized into ordered sequences known as Service Function Chains (SFCs). However, optimizing VNF placement within SFCs to balance resource utilization and Quality of Service (QoS) remains a major challenge in MEC environments. Existing SFC deployment methods often face limitations in adaptability and efficiency, as they rely on static or heuristic approaches that struggle to handle dynamic network conditions and diverse resource requirements effectively. To overcome these challenges, we propose DeepSFCOpt, a Deep Reinforcement Learning (DRL)-based optimization method for SFC deployment that integrates Graph Convolutional Networks (GCNs) to extract network features and Sequence-to-Sequence(Seq2Seq) models to capture the order of SFCs, enabling adaptive placement strategies that optimize resource allocation and maximize long-term average revenue, thus more effectively meeting the demands of IoT services.

***Index Terms*—SFC deployment, Virtual function, Deep reinforcement learning, GNN, CARU**

I. Introduction

INFRASTRUCTURE providers (InPs) play a critical role in managing and optimizing network infrastructures, which are essential for supporting the diverse traffic types required by various applications. To efficiently handle these traffic requests, deploying Service Function Chains (SFCs) using Mobile Edge Computing (MEC) and Network Function Virtualization (NFV) has emerged as a promising solution [1]. NFV enables the implementation of traditional hardware-based network functions on general-purpose computing devices in the form of software. These software-based implementations, known as Virtual Network Functions (VNFs), include fundamental functions such as Firewalls (FW), Deep Packet Inspection (DPI), Intrusion Detection Systems (IDS), etc. [2]. Traditionally, these network functions are realized through dedicated hardware appliances, which often incur significant capital and operational expenses. NFV technology, by leveraging software applications running on Commercial Off-The-Shelf (COTS) servers through Virtual Machines (VMs), offers a more flexible approach that can adapt to dynamic network environments and meet the demands of various network services.

Additionally, Software-Defined Networking (SDN) technology, by decoupling the data plane from the control plane, provides a flexible network framework and efficient resource management [3]. This enables IoT services to be managed in a more centralized and agile manner. In IoT service scenarios, traffic requests generated by IoT devices typically need to traverse multiple VNF instances sequentially, forming what can be termed as IoT SFC requests (IoT-SRs) [4].To address the challenges posed by resource constraints, the SDN controller must devise an optimal dynamic VNF placement strategy, a challenge often referred to as the dynamic SFC placement problem. This involves efficiently embedding ordered SFCs into physical network nodes within a continuously changing network environment, ensuring that traffic flows sequentially through nodes providing the required functions, thereby effectively integrating the virtual service function chain with the underlying physical network topology. Unlike the static nature of the Virtual Network Function Forwarding Graph (VNF-FG) and Virtual Network Embedding (VNE) problems, dynamic SFC placement requires adaptive deployment decisions based on real-time changes in network resources and service demands. Specifically, VNF-FG focuses on arranging VNF forwarding paths within a fixed topology, while VNE is concerned with mapping a virtual network onto a physical network.

In resource-constrained environments, the effective placement of sequential SFC requests within physical networks is a major challenge faced by InPs. Traditional methods, such as ILP models and heuristic algorithms [5][6][7], have been widely used to address SFC deployment problems. For example, [8] proposed a pair-based online approximation algorithm aimed at maximizing a profit function, but this algorithm only considered two types of VNFs, limiting its applicability in real-world scenarios. To address this limitation, [5] introduced a greedy algorithm; however, its computational complexity increases significantly as network conditions become more dynamic. Similarly, [6] designed a dynamic programming-based algorithm with the goal of minimizing resource costs. Despite these efforts, the effectiveness of traditional optimization methods in solving dynamic SFC placement problems remains challenging. In large-scale networks, ILP methods often encounter significant computational difficulties. Furthermore, heuristic methods are prone to getting trapped in local optima when dealing with rapidly changing network dynamics, limiting their applicability for VNF placement in dynamic IoT networks.

Recently, with the rise of machine learning techniques,

Manuscript received September 27, 2024; revised November 23, 2024.
Corresponding author: Limei Peng (email: auroraplm@knu.ac.kr).

especially Deep Reinforcement Learning (DRL), VNF placement problems have been tackled using DRL-based solutions. For instance, [9] proposed a DRL algorithm to address the VNFs placement issue, demonstrating better performance than traditional methods without relying on human expertise or handcrafted models. The author in [10] proposed a distributed DRL method to address the challenges of multi-objective optimization in dynamic environments, leveraging deep neural networks to approximate the decision-making function. Another study focused on the reliability of SFC deployment in NFV environments, introducing a dual-path mechanism where each SFC has a primary and a backup path, ensuring seamless traffic rerouting in case of path failures [11].Furthermore, the author in [12] introduced a two-stage DRL-based SFC deployment algorithm has been developed, combining graph-based resource aggregation routing with DRL to improve acceptance rates while meeting latency constraints.Additionally, in a multi-data center environment, the integration of graph GCN with DRL has been shown to enhance SFC embedding efficiency by effectively handling complex network topologies [13].The author in [14] employed a meta-reinforcement learning (MRL) approach to facilitate rapid adaptation to varying network states and optimize SFC deployment strategies. The existing DRL-based methods have made certain progress in SFC deployment; however, most of these methods face challenges such as high computational complexity, poor adaptability, and slow convergence. Additionally, many approaches fail to adequately address real-time decision-making under complex network topologies and dynamic changes, leading to limited efficiency and reliability in large-scale practical environments.

Therefore, in this paper, we propose a DRL-based method called DeepSFCOpt. This method generates placement strategies by combining a GCN, which extracts physical network features, with Seq2Seq models, which capture the sequential information of SFC requests. The aim is to maximize long-term average revenue and achieve flexible and efficient resource allocation to meet the demands of IoT services. Unlike traditional approaches, it makes SFC placement decisions based on the performance of past decisions, rather than relying on assumptions about the environment. The proposed DeepSFCOpt deployment method leverages Deep Neural Networks (DNNs) to explore the non-Euclidean structural characteristics of the physical network and the ordered information of the SFC requests. Furthermore, we utilize Deep Q-Networks (DQNs) in the training process of DeepSFCOpt to select appropriate physical network nodes for SFC deployment. This approach improves the acceptance ratio and long-term average revenue while maintaining low running time, thereby achieving flexible and efficient resource allocation.

The rest of the paper is organized as follows. Secion II introduces the system model and formulate the problem. Section III presents the details of DeepSFCOpt Scheme. The numerical results and discussions are presented in Section IV. Finally, we conclude our work in Section V.

The main contributions of this paper are synthesized as follows:

- We formulate a viable NFV/MEC-enabled IoT architecture and propose DeepSFCOpt, a DRL-based framework for optimizing SFC deployment. This framework integrates GCN and Seq2Seq models to extract the topological features of the physical network and the sequential dependencies of SFC requests. Furthermore, it incorporates an innovative Compositional Attention-based Recurrent Unit (CARU), which strengthens the model's capability to analyze and process complex service chain sequences, enabling more efficient and effective deployment strategies.
- We incorporate DQNs into the training process of SFC deployment optimization, leveraging experience replay and target network techniques to significantly enhance training stability and convergence efficiency. Experimental results demonstrate that DeepSFCOpt achieves higher acceptance rates, long-term revenue, and resource utilization efficiency in dynamic IoT scenarios, showcasing its practical potential in addressing challenges within complex and dynamic environments.

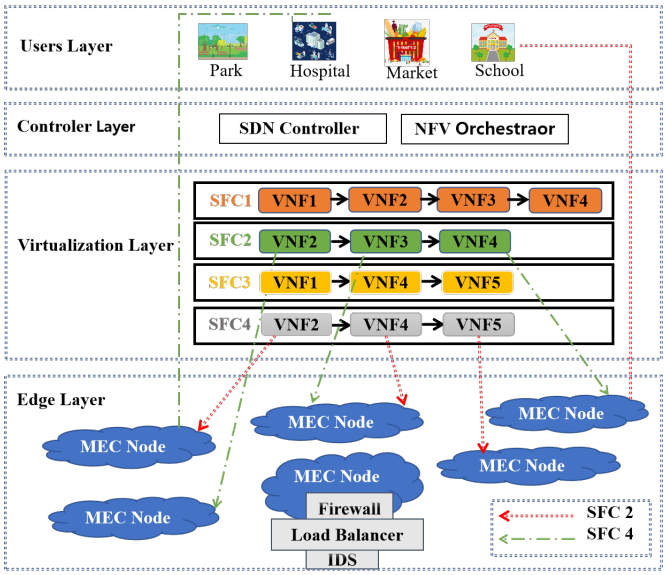

Fig. 1. NFV/MEC-enabled IoT architecture

II. SYSTEM MODEL AND PROBLEM FORMULATION

Figure 1 illustrates the hierarchical NFV/MEC-enabled IoT architecture. It shows that IoT service requests from users are sequentially assigned to the nearest adjacent nodes at the edge, which then communicate with the SDN controller and NFV orchestrator in the control layer to embed VNFs into appropriate nodes to fulfill the service requests.

A. System Model

The considered network can be modeled as a weighted undirected graph $G^{\mathrm{P}} = (N^{\mathrm{P}}, L^{\mathrm{P}})$, where G^{P} represents the graph model of the physical network, encompassing all nodes and links within the network; N^{P} represents the set of nodes and L^{P} denotes the set of links in the physical network. The set of node resources, such as central processing units

(CPUs), random access memory (RAM),etc., provided by the physical network is denoted by K. The vectors of remaining and maximum resources for a physical node $n^{\mathrm{P}} \in N^{\mathrm{P}}$ are represented as $R^r_{n^{\mathrm{P}}} = \left(R^r_{n^{\mathrm{P}},1}, \ldots, R^r_{n^{\mathrm{P}},k}, \ldots, R^r_{n^{\mathrm{P}},|K|}\right)$ and $R^m_{n^{\mathrm{P}}} = \left(R^m_{n^{\mathrm{P}},1}, \ldots, R^m_{n^{\mathrm{P}},k}, \ldots, R^m_{n^{\mathrm{P}},|K|}\right)$, respectively. Here, $R^r_{n^{\mathrm{P}},k}$ and $R^m_{n^{\mathrm{P}},k}$ denote the remaining and maximum amounts of resource $k \in K$ provided by node n^{P}. Link-oriented attributes may include bandwidth, latency, packet loss, etc. For simplicity, this paper considers bandwidth and latency as the primary link-oriented attributes.

B. SFC Request

Each SFC request i can be modeled as a weighted directed graph $G^{\mathrm{i}} = (N^{\mathrm{i}}, L^{\mathrm{i}})$, where G^{i} represents the graph model of SFC request i, including all VNFs and virtual links related to the request. This graph is used to describe the sequence and dependencies among the functions in the SFC; N^{i} denotes the set of VNFs in SFC request i, containing all the required functions for the request and L^{i} refers to the set of virtual links in SFC request i, which includes the connections between the VNFs. We denote the vector of requested resources for VNF $n^i \in N^i$ as $r^{n^i} = [r^{n^i}_1, \ldots, r^{n^i}_k, \ldots, r^{n^i}_{|K|}]$, where $r^{n^i}_k$ is the amount of resource $k \in K$ requested by VNF n^i. Additionally, the latency requirement for SFC i is denoted as Q^i. The remaining and maximum bandwidth of a physical link $l^P \in L^P$ are represented as $B^r_{l^P}$ and $B^m_{l^P}$, respectively.

C. Promblem Formulation

Figure.2 provides an example of SFC deployment, illustrating how an arriving SFC G^{i} is mapped to the physical network G^{P} under resource constraints.

According to the aforementioned network model, each VNF utilizes the resources allocated by the node to process the forwarded packets. For an IoT service with a packet size of d_j, the processing delay of a VNF at the node is expressed as:

$$\gamma(n_{i,j})(t) = \frac{d_j}{\alpha^n_i(t)} \tag{1}$$

where $\alpha^n_i(t)$ represents the portion of resources allocated to VNF n_i at time t. By dividing $\alpha^n_i(t)$ into three parts, the above equation can be rewritten as:

$$\gamma(n_{i,j})(t) = x^c_{i,j}\frac{d^c_j}{c^n_i} + x^s_{i,j}\frac{d^s_j}{s^n_i} + x^\omega_{i,j}\frac{d^\omega_j}{\omega^n_i} \tag{2}$$

Here, d^c_j denotes the size of the data packet in terms of CPU, d^s_j represents the size of the data packet in terms of storage, and d^ω_j indicates the size of the data packet in terms of transmission. c^n_i, s^n_i, and ω^n_i refer to the CPU, storage, and transmission resources allocated to VNF node n_i, respectively. $x^c_{i,j}$, $x^s_{i,j}$, and $x^\omega_{i,j}$ are binary variables. When they take the value of 1, it means that the request requires the relevant resources; when they take the value of 0, it means that the request does not need that type of resource.

The transmission delay between two VNF at nodes n_i and n_j is expressed as:

$$\gamma^l_{n_i,n_j}(t) = r_{n_i,n_j}\frac{d_j}{l(l_i)} \tag{3}$$

where $l(l_i)$ is the capacity of the virtual link l_i, which can be expressed as:

$$l(l_i) = B\log_2\left(1 + \frac{P_{n_i}|g(n_i,n_j)|^2}{\theta^2 + \sum_{n_k\in\xi, n_k\neq n_j} P_{n_k}|g(n_k,n_j)|^2}\right) \tag{4}$$

In this equation, B is the bandwidth, P_{n_i} is the transmission power of node n_i, θ^2 is the noise variance and $g(n_i, n_j)$ is the channel gain between the two edge nodes,reflecting the attenuation effect of distance on the channel. As the distance between nodes increases, the channel gain $g(n_i, n_j)$ decreases,resulting in a reduction in link capacity $l(l_i)$, which in turn leads to an increase in transmission delay. Consequently, the end-to-end delay for the service requested by terminal j primarily includes the processing delay at the nodes, the transmission delay, and the queuing delay γ_q (M/M/1 queue model) expressed as follows:

$$Q_j(t) = \sum_{n_i\in N_i}\gamma(n_{i,j})(t) + \sum_{l_i\in L_i}\gamma^l_{n_i,n_j}(t) + \gamma_q \tag{5}$$

To ensure that the end-to-end delay does not exceed the maximum threshold $Q_{\max}$, the following constraint must be satisfied:

$$Q_j(t) \le Q_{\max} \tag{6}$$

In this paper, we assume that the InP charges for SFC using the widely adopted "pay-as-you-go" pricing model commonly seen in cloud platforms. The revenue obtained from each SFC request i is defined as:

$$\mathrm{rev}(i) = \begin{cases} \alpha_k \sum_{n_i\in N_i} r_{n_i,k} + \beta\sum_{l_i\in L_i} b_{l_i}, & \text{if } i \text{ is accepted} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where α_k is the unit price of resource k in physical nodes, and β is the unit price of bandwidth. We use $R(\pi)$ to denote the long-term average revenue, which is defined as:

$$R(\pi) = \lim_{\tau\to\infty}\frac{1}{\tau}\sum_{i\in I_\tau}\mathrm{rev}(i) \tag{8}$$

where $I_\tau = \{i \mid 0 < t_i < \tau\}$ represents the set of SFC that arrive before time τ. Considering the variability of the channel gain over time, our objective is to determine an optimal policy π^* that maximizes the long-term average revenue:

$$\pi^* = \arg\max_{\pi} R(\pi) \tag{9}$$

Satisfy the other constraints:

$$\sum_{n_i\in N_i}\varphi^{n_i}_{n_p} r_{n_i,k} \le R^r_{n_p,k} \tag{10}$$

$$\sum_{l_i\in L_i}\varphi^{l_i}_{l_p} b_{l_i} \le B^r_{l_p} \tag{11}$$

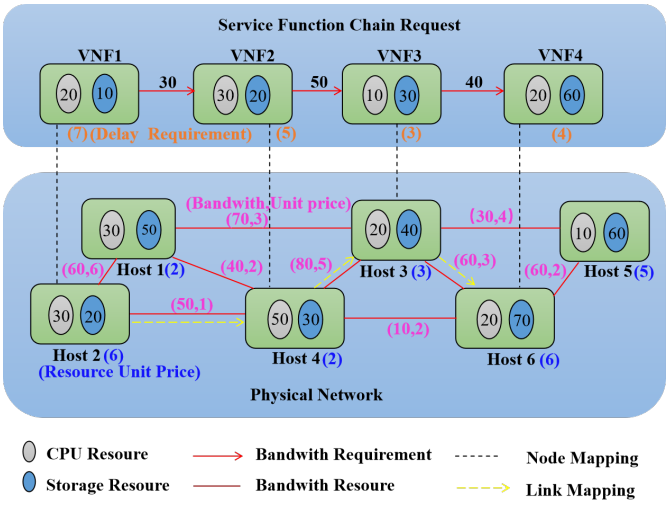

Fig. 2. An example of SFC placement

$$\sum_{l_p \in I(n_p)} \varphi_{l_p}^{l_i} - \sum_{l_p \in O(n_p)} \varphi_{l_p}^{l_i} = \varphi_{n_p}^{n_i^t} - \varphi_{n_p}^{n_i^s} \tag{12}$$

$$\sum_{n_p \in N_p} \varphi_{n_p}^{n_i} = 1 \tag{13}$$

$$\sum_{l_p \in l_i} \varphi_{l_p}^{l_i} = 1 \tag{14}$$

Constraint (10) ensures that the total demand for resource K by all virtual network functions n_i mapped to a physical node does not exceed the remaining resources of the physical node. Constraint (11) ensures that the total bandwidth demand of all virtual links l_i mapped to a physical link does not exceed the remaining bandwidth of that link. Constraint (12) stipulates that if a SFC request is accepted, the path mapped in the physical network must traverse the VNF in the specified order as defined in the request. Constraint (13) requires that each virtual network function n_i must be mapped to exactly one physical node n_p. Constraint (14) requires that each virtual link l_i must be mapped to exactly one physical link l_p.

III. The Proposed DeepSFCOpt Scheme

A. Model Architecture

The model architecture of DeepSFCOpt, illustrated in Fig.3, consists of three components: (i) a GCN for extracting physical network features, (ii) a Seq2Seq model capable of capturing the sequential information of SFC requests, (iii) and a module for generating placement strategies. We will explain each component in detail.

1) GCN for Network Features: We utilize a semi-supervised GCN to extract features from physical networks in order to explore their topological structures. GCNs are particularly well-suited for analyzing complex network topologies as they efficiently handle non-Euclidean data and capture both local and global network features by aggregating information from neighboring nodes.At each time step t, the current state of the physical network $s_t^p = (Z, Y)$ is fed into a GCN layer to learn a new representation matrix $Z_t \in \mathbb{R}^{|N_P| \times U_{\text{gcn}}}$, where U_{gcn} is the number of units in the GCN layer. The mathematical operation of the GCN is briefly formalized as follows:

$$Z_t = \text{GCN}(s_t^p) = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}YW) \tag{15}$$

Here, σ is the activation function, and W represents the trainable parameters. The term $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ serves as the approximated graph convolution filter, analogous to the filters used in Convolutional Neural Networks (CNNs).

2) Seq2Seq model for SFCs: This paper utilizes the capabilities of the Seq2Seq model for efficient sequence mapping by employing an encoder based on a Compositional Attention-based Recurrent Unit (CARU) network [15]. This design effectively captures the sequential information inherent in SFC requests. The CARU processes an input sequence $s_r = (s_r^1, \ldots, s_r^T)$ to produce hidden states e_t. At each time step t, the CARU unit takes the current input s_r^t along with the hidden state from the preceding time step e_{t-1} as inputs, generating the hidden state e_t for the current time step. The CARU operation is defined as follows:

$$e_t = \text{CARU}(s_r^t, e_{t-1}). \tag{16}$$

The following equations represent the operations within CARU, where $[W; B]$ denotes the training parameters, including weights and biases, which form the linear layers. We set $t = 0$ as the initial state. At the initial state, CARU directly returns:

$$e^{(1)} = W_{vn}v^{(0)} + B_{vn} \tag{17}$$

For $t > 0$, the complete recursion loop for CARU is as follows:

$$x^{(t-1)} = W_{vn}v^{(t-1)} + B_{vn} \tag{18}$$

$$n^{(t-1)} = \phi\left(W_{hn}e^{(t-1)} + B_{hn} + x^{(t-1)}\right) \tag{19}$$

$$z^{(t-1)} = \sigma\left(W_{hz}e^{(t-1)} + B_{hz} + W_{vz}v^{(t-1)} + B_{vz}\right) \tag{20}$$

$$l^{(t-1)} = \sigma\left(x^{(t-1)}\right) \odot z^{(t-1)} \tag{21}$$

$$e^{(t)} = (1 - l^{(t-1)}) \odot e^{(t-1)} + l^{(t-1)} \odot n^{(t-1)} \tag{22}$$

where $\odot$ denotes the Hadamard product, σ and ϕ are the activation functions, specifically the sigmoid and hyperbolic tangent functions, respectively.

3) Placement Strategy Generation: The Seq2Seq model's decoder is initialized with the encoder's final state, e_T. At each time step t, the decoder leverages the current state d_t, the context vector c_t [16], and the output from the GCN layer Z_t to generate an action sequence $a = (a_1, \ldots, a_T)$. To derive the probability distribution over potential actions, these components—d_t, c_t, and Z_t are concatenated and processed through a fully con-

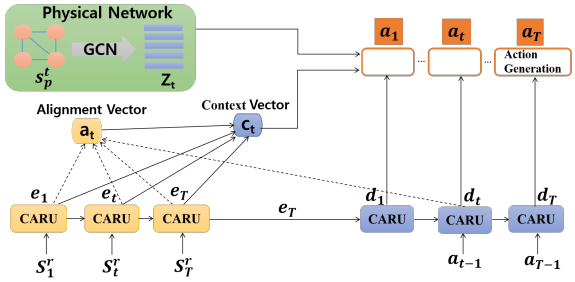

Fig. 3. The network architecture of DeepSFCOpt

nected layer. A softmax function is subsequently applied to align the output size with the number of available physical network nodes. The conditional probability for each action is calculated as follows:

$$\pi[\cdot \mid \{a_1, \ldots, a_{t-1}\}, d_t, c_t, Z_t] = \text{softmax}(d_e^t), \quad (23)$$

where

$$d_e^t = \mathbf{v}_b^T \tanh(\mathbf{W}_b[d_t; c_t; Z_t]). \quad (24)$$

Based on this conditional probability, the agent selects the action a_t, determining the physical node for the placement of the current VNF.

We utilize the Dijkstra algorithm to determine the shortest path between nodes a_t and a_{t-1} within the physical network. If this path satisfies the bandwidth and latency requirements, the VNF is successfully deployed at a_t. Otherwise, the placement attempt fails, and the previously allocated physical resources are released.

B. Preliminary Work

In dynamic and complex environments, this study's SFC deployment scheme aims to determine sequential strategies that satisfy Markov properties. Based on the DRL framework, the interaction between the agent and the environment is defined as a finite-horizon Markov Decision Process (MDP). For each SFC request, we model the deployment problem as an MDP, with the basic elements including state S, action A, and reward R.

1) State: The state should include two parts: the current physical network characteristics and the SFC request characteristics. Therefore, we define the state as $s_t = (s_t^p, s_t^r)$. Here, $s_t^p = (Z, Y)$ represents the current state of the physical network, where $Z \in \mathbb{R}^{|N_P| \times |N_P|}$ is the adjacency matrix of the network, and $Y \in \mathbb{R}^{|N_P| \times M}$ is the feature matrix of the physical nodes. Each row of Y is an M-dimensional feature vector of the physical node n_p. s_t^r represents the current state of the SFC request, including the resource requirements of each VNF in the SFC chain, the required link bandwidth, and the end-to-end delay threshold $Q_{\max}$.
2) Action: Selecting nodes from physical nodes with available resources exceeding the VNF requirements for SFC deployment. The action set is defined as: $A_t = \{\gamma\} \cup \{n_p \in N_P \mid r_{n_t^i,k} \leq R^r_{n_p,k}, \forall k \in K\}$.

Algorithm 1 DeepSFCOpt Algorithm

H

Input: Physical network G_P, SFC requests $\{G_i\}$, episodes E, learning rate α, discount factor γ
Output: Optimal SFC placement policy π^*
Initialize experience replay buffer D, target network parameters θ, $\theta_{\text{target}} \leftarrow \theta$
for each episode $e = 1$ to E **do**
  Reset environment and initialize state s
  for each SFC request $G_i \in \{G_i\}$ **do**
    Observe current physical network state s_t^p and SFC request s_t^r
    Combine s_t^p and s_t^r as current state s_t
    $Z_t \leftarrow \text{GCN}(s_t^p)$
    for each VNF in G_i **do**
      $e_t \leftarrow \text{CARU}(s_t^r, e_{t-1})$
    end for
    for each VNF n_i in SFC G_i **do**
      $a_t \leftarrow$ Select physical node n_p using ϵ-greedy policy based on state s_t
      if n_p satisfies resource and bandwidth constraints **then**
        Deploy VNF n_i on node n_p
      else
        Reject placement and rollback resources
      end if
      Store transition (s_t, a_t, r_t, s_{t+1}) in buffer D
    end for
    if G_i is successfully deployed **then**
      $r_t \leftarrow$ Calculate reward based on resource utilization and revenue
    else
      $r_t \leftarrow$ Negative reward
    end if
    Update state to s_{t+1} after SFC deployment
  end for
  Sample mini-batch of experiences from D
  for each (s_t, a_t, r_t, s_{t+1}) in mini-batch **do**
    $Q_{\text{target}} \leftarrow r_t + \gamma \cdot \max(Q(s_{t+1}, a_{t+1}; \theta_{\text{target}}))$
    Update Q-network parameters θ using gradient descent on loss: $(Q(s_t, a_t; \theta) - Q_{\text{target}})^2$
  end for
  if step % update_freq == 0 **then**
    Update target network: $\theta_{\text{target}} \leftarrow \theta$
  end if
end for
return Optimal policy π^*

TABLE I. Simulation Parameters

Name	Value	Description
μ_k	0.001	The unit price of resource k
ω	0.001	The unit price of bandwidth
ϵ	0.1	The initial exploration rate (epsilon-greedy policy)
ϵ_{min}	0.001	The minimum exploration rate
ϵ_{decay}	0.995	The decay rate of exploration rate per episode
α	0.00025	The learning rate of Q-network
γ	0.95	The discount factor of TD error
δ	0.125	The reward coefficient
B	64	The batch size
$\|D\|$	10000	Size of the experience replay buffer
$U_{\text{gcn}}, U_{\text{emd}}, U_{\text{enc}}, U_{\text{dec}}$	64	The units number of GCN layer, embedding layer, encoder hidden states, and decoder hidden states
$\theta_{\text{update\_freq}}$	1000	The frequency of updating target network parameters

3) Reward: The reward signal is designed to encourage the agent to deploy SFC with the aim of maximizing long-term average revenue. To this end, when the SFC request i is accepted at time step $t = T$, the agent receives a reward $r_t = \text{rev}_i$, where

$$\text{rev}_i = \alpha_k \sum_{n_i \in N_i} r_{n_i,k} + \beta \sum_{l_i \in L_i} b_{l_i} \tag{25}$$

During the intermediate steps ($t < T$), if all resource constraints, including bandwidth and latency, are satisfied, the agent receives a small reward given by $r_t = \xi \text{rev}_i$. Otherwise, the reward is $r_t = -\xi \text{rev}_i$, where ξ is the reward coefficient.

The complete execution process of our DeepSFCOpt approach is shown in Algorithm 1.

IV. NUMERICAL RESULTS AND DISCUSSIONS

A. Experimental Setup

During the experiment, we randomly generated a physical network consisting of 200 nodes and 500 links to simulate a medium-sized InP. The system sequentially receives 2,500 SFC requests, with an average arrival rate of 30 requests within 100 time units. Each SFC request contained between 2 to 10 VNFs, following a uniform distribution.During the training phase, the DQN algorithm is employed[17]. The agent stores interactions with the environment using an experience replay mechanism and utilizes a target network to stabilize the training process. Parameters of the Q-network are updated through mini-batch training using samples from the experience replay buffer. In the testing phase, only the trained Q-network is used to handle the SFC requests. A greedy strategy is adopted for action selection to ensure the utilization of the optimal strategy learned during training.More parameters can be found in TABLE I.

B. Evaluation and Analysis

To evaluate the effectiveness of the proposed DeepSFCOpt approach, we compare it with the two following approaches:

- **DRL-A3C** : An asynchronous actor-critic algorithm where multiple agents learn in parallel, updating a shared model for improved decision-making.
- **TD (Temporal Difference Learning)** : A RL method that updates value estimates using the difference between predicted and received rewards.
- **SFC3D** : A dynamic SFC placement algorithm in distributed data center networks based on DRL,which was proposed in [18].

Figure.4 shows the acceptance ratios of three algorithms during the testing phase for SFC requests. We observe that initially, the acceptance ratios of all three algorithms demonstrate a declining trend, as the resources of the physical network, including link bandwidth, are gradually consumed by incoming SFC requests.In the latter half of the testing phase, DeepSFCOpt achieves the highest acceptance rate, reaching 76.8%. Compared to DRL-A3C and TD, DeepSFCOpt's performance is improved by 2.9% and 9.3%, respectively.

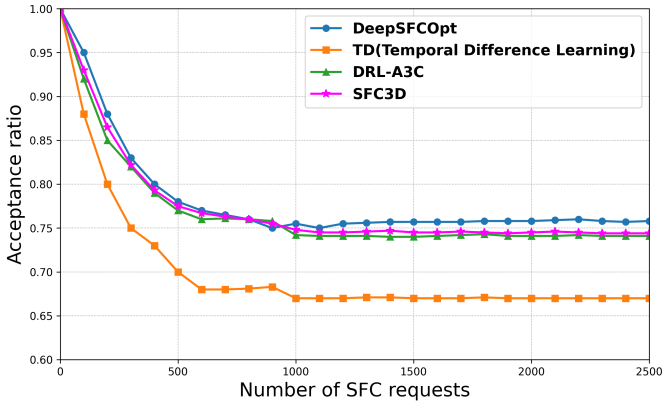

Fig. 4. Acceptance ratio over different SFC requests.

In Figure.5, we compare the average revenue of all three algorithms as they sequentially process SFC requests. After processing 2,500 SFC requests, DeepSFCOpt achieves the highest long-term average revenue of 0.068, which is an improvement of 4.6% over DRL-A3C, 11.4% over TD, and 1.5% over SFC3D. This indicates that our algorithm can

intelligently make SFC placement decisions, benefiting from the effective utilization of physical network characteristics and SFC request features.

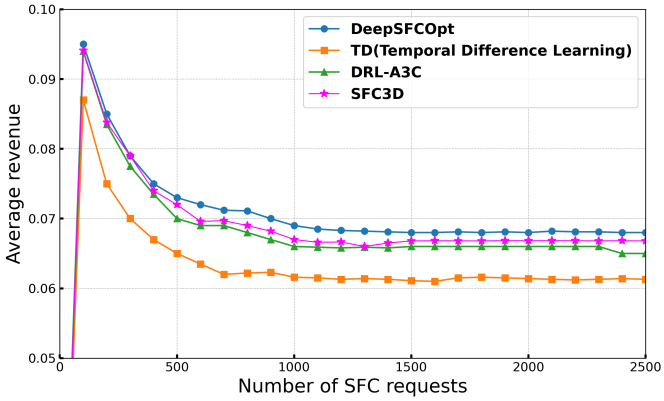

Fig. 5. Average revenue over different SFC requests.

Figures. 6 and 7 illustrate the acceptance ratios and long-term average revenue under different arrival rates. To simulate the demand variance of SFC requests between busy and idle hours, we increase the arrival rate from an average of 12 requests per 100 time units to 30 requests, with an increment of 3 requests per step. The results indicate that as the arrival rate increases, the acceptance ratio decreases while the long-term average revenue increases. Additionally, DeepSFCOpt outperforms DRL-A3C,TD and SFC3D in both acceptance ratio and revenue.

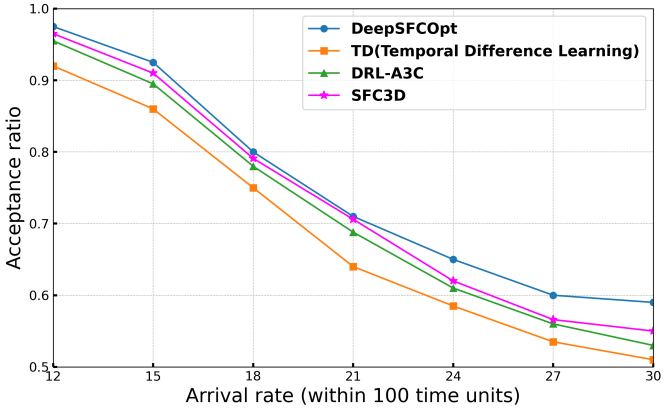

Fig. 6. Acceptance ratio over different arrival rates.

V. Conclusion

This study investigates how to efficiently deploy SFC onto physical networks to maximize the long-term average revenue of InP. Specifically, we propose a DeepSFCOpt algorithm that combines GCN and Seq2Seq models to extract features from the physical network and SFC requests, while utilizing the DQN algorithm to accelerate the training process and intelligently generate SFC placement strategies. Numerical results demonstrate the superiority and effectiveness of this method. In the future, we plan to further investigate the deployment and resource coordination of SFCs within multi-domain network

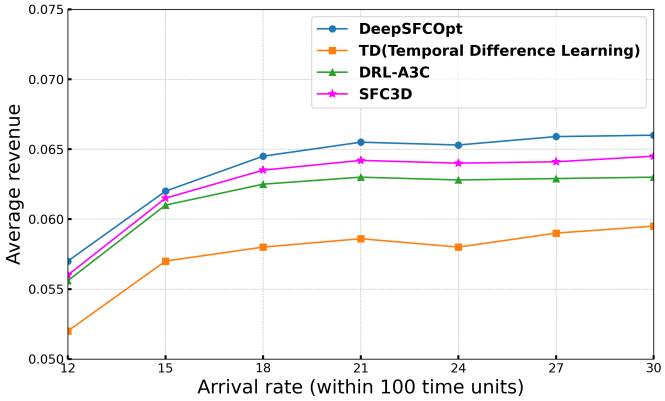

Fig. 7. Average revenue over different arrival rates.

environments to address the limitations of current single-domain deployment assumptions.In multi-domain networks, such as those spanning heterogeneous networks or networks managed by different operators, SFC deployment should not only resolve inconsistencies in network protocols across domains but also address the challenges of inter-domain trust and collaboration.Federated Learning (FL) or distributed optimization methods offer potential solutions for achieving efficient cross-domain collaboration while ensuring data privacy.

VI. Acknowledgement

This research was partially supported by the Brain Pool Program funded by the Ministry of Science and ICT through the National Research Foundation of Korea (NRF) (Grant No. NRF-2022H1D3A2A01063679), the NRF grant funded by the Korean government (MSIT) (Grant No. RS-2024-00457947), and the 2024 Kyungpook National University BK21 FOUR Graduate Innovation Project (International Joint Research Project for Graduate Students).

References

[1] A. Abouaomar, S. Cherkaoui, Z. Mlika, and A. Kobbane, "Service function chaining in mec: A mean-field game and reinforcement learning approach," *IEEE Systems Journal*, vol. 16, no. 4, pp. 5357–5368, 2022.

[2] F. Schardong, I. Nunes, and A. Schaeffer-Filho, "Nfv resource allocation: A systematic review and taxonomy of vnf forwarding graph embedding," *Computer Networks*, vol. 185, p. 107726, 2021.

[3] J. C. C. Chica, J. C. Imbachi, and J. F. B. Vega, "Security in sdn: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 159, p. 102595, 2020.

[4] X. Fu, F. R. Yu, J. Wang, Q. Qi, and J. Liao, "Dynamic service function chain embedding for nfv-enabled iot: A deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 507–519, 2019.

[5] B. Yuan and B. Ren, "Embedding the minimum cost sfc with end-to-end delay constraint," in *2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*. IEEE, 2020, pp. 2299–2303.

[6] I. Sarrigiannis, K. Ramantas, E. Kartsakli, P.-V. Mekikis, A. Antonopoulos, and C. Verikoukis, "Online vnf lifecycle management in an mec-enabled 5g iot architecture," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4183–4194, 2019.

[7] P. Jin, X. Fei, Q. Zhang, F. Liu, and B. Li, "Latency-aware vnf chain deployment with efficient resource reuse at network edge," in *IEEE INFOCOM 2020-IEEE conference on computer communications*. IEEE, 2020, pp. 267–276.

[8] O. Alhussein and W. Zhuang, "Robust online composition, routing and nf placement for nfv-enabled services," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1089–1101, 2020.
[9] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou, "Optimal vnf placement via deep reinforcement learning in sdn/nfv-enabled networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 263–278, 2019.
[10] H. Xing, Y. Pu, X. Wang, F. Song, Z. Xiao, L. Feng, and L. Xu, "Big data oriented multi-objective sfc placement in dynamic mec: A distributed drl approach," in *ICC 2024 - IEEE International Conference on Communications*, 2024, pp. 666–671.
[11] Y. Zhong, D. Zheng, and X. Cao, "A drl approach with network service deployment transformer for reliable sfc deployment," in *ICC 2024 - IEEE International Conference on Communications*, 2024, pp. 177–182.
[12] A. Tian, B. Feng, Y. Huang, H. Zhou, S. Yu, and H. Zhang, "Drl-based two-stage sfc deployment approach under latency constraints," in *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2024, pp. 1–6.
[13] D. Xiao, J. A. Zhang, X. Liu, Y. Qu, W. Ni, and R. P. Liu, "A two-stage gcn-based deep reinforcement learning framework for sfc embedding in multi-datacenter networks," *IEEE Transactions on Network and Service Management*, vol. 20, no. 4, pp. 4297–4312, 2023.
[14] S. Guo, Y. Du, and L. Liu, "A meta reinforcement learning approach for sfc placement in dynamic iot-mec networks," *Applied Sciences*, vol. 13, no. 17, p. 9960, 2023.
[15] D. Hu, S. Lyu, S. Y. Chang, L. Peng, and P.-H. Ho, "Dynamic siot network status prediction," *Journal of Networking and Network Applications*, vol. 2, no. 2, pp. 78–85, 2022.
[16] D. Bahdanau, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.
[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[18] J. Jia and J. Hua, "Dynamic sfc placement with parallelized vnfs in data center networks: A drl-based approach," *ICT Express*, vol. 10, no. 1, pp. 104–110, 2024.