Capacitated Shortest Path Tour-Based Service Chaining Adaptive to Changes of Service Demand and Network Topology

Takanori Hara, *Member, IEEE*, and Masahiro Sasabe, *Member, IEEE*

***Abstract*—To achieve sustainable networking, network service providers have expressed significant interest in employing automated network operations that integrate network functions virtualization (NFV), software-defined networking (SDN), and machine learning (ML). In the context of NFV/SDN, a certain network service is regarded as a sequence of virtual network functions (VNFs) forming a service chain. The service chaining (SC) problem aims at establishing an appropriate service path from an origin node to a destination node where the VNFs are executed at intermediate nodes in the required order under resource constraints on nodes and links. SDN enables programmable configurations on forwarding devices (i.e., switches and routers) for traffic forwarding between VNFs. In our previous work, we formulated the SC problem as an integer linear program (ILP) based on the capacitated shortest path tour problem (CSPTP), which is an extended version of SPTP with additional node and link capacity constraints. Furthermore, we developed Lagrangian heuristics to solve the problem by considering the balance between optimality and computational complexity. In this paper, we propose a deep reinforcement learning (DRL) framework coupled with the graph neural network (GNN) to realize CSPTP-based SC that adapts to changes of service demand and/or network topology. Numerical results show that the proposed framework achieves nearly optimal SC with higher learning speed compared to the conventional deep Q-Network based approach. Moreover, it performs well when confronted with variations in service demand and exhibits competitive performance compared to the ILP solutions across the majority of 243 real-world topologies.**

***Index Terms*—Network functions virtualization (NFV), software defined networking (SDN), service chaining (SC), capacitated shortest path tour problem (CSPTP), deep reinforcement learning (DRL), graph neural network (GNN).**

Manuscript received 28 February 2023; revised 27 July 2023 and 22 December 2023; accepted 6 January 2024. Date of publication 9 January 2024; date of current version 15 April 2024. This work was supported in part by the Japan Society for the Promotion of Science (JSPS) Research Activity Start-up under Grant 21K21288, the JSPS KAKENHI (B) under Grant 22H03586, and the Support Center for Advanced Telecommunications Technology Research (SCAT). This article is an extended version of the paper presented at the 2022 International Conference on Network and Service Management (CNSM 2022) [1] [DOI: 10.23919/CNSM55787.2022.9965166]. The associate editor coordinating the review of this article and approving it for publication was S. Kanhere. *(Corresponding author: Takanori Hara.)*

Takanori Hara is with the Division of Information Science, Nara Institute of Science and Technology, Ikoma 630-0192, Nara, Japan (e-mail: hara@ieee.org).

Masahiro Sasabe is with the Faculty of Informatics, Kansai University, Takatsuki 569-1095, Osaka, Japan (e-mail: m-sasabe@ieee.org).

Digital Object Identifier 10.1109/TNSM.2024.3351737

I. Introduction

With rapidly spreading smartphones and Internet of things (IoT) devices, diverse services have constantly been created and the network traffic has exponentially been increasing. To achieve sustainable networking, network service providers have shown considerable interest in employing automated network operations that integrate *network functions virtualization* (NFV), *software-defined networking* (SDN), and *machine learning* (ML). NFV can decouple network functions from dedicated hardware and execute them as *virtual network functions* (VNFs) on generic hardware [2], [3], [4]. As a result, it can deploy network services with agility and flexibility as well as reducing capital expenditure (CAPEX) and operating expenditure (OPEX). SDN separates the control plane from the data plane, thereby achieving programmable networking through the centralized control functionality [5]. As a result, it facilitates dynamic traffic steering and routing based on specific policy rules for each network service.

NFV and SDN are mutually complementary [6], [7], [8], [9], [10], [11]. NFV facilitates the virtualization of an SDN controller and SDN data forwarding rules (referred to as network functions), enabling dynamic and optimal lifecycle management of these components. SDN, on the other hand, provides programmable networking capabilities between VNFs, allowing for dynamic and optimal traffic steering and routing. The combined characteristics of NFV and SDN technologies foster the advancement of *service chaining* (SC) [2], which facilitates the directed flow of traffic through a predefined sequence of network functions. A certain network service can be expressed by a sequence of VNFs, called a *service (function) chain*. Given a *service chain request* (SCR), an SC orchestrator tries to solve an SC problem, which aims at establishing a special path (i.e., *service path*) from an origin node to a destination node, where the VNFs are executed at the intermediate nodes one by one under the resource constraints [2]. It is well known that the SC problem belongs to the complexity class NP-hard [12].

Several existing studies [13], [14] also pointed out the similarity between SC and shortest path tour problem (SPTP), which is a variant of shortest path problem and aims at calculating the shortest path from an origin node to a destination node while visiting at least one node from given disjoint node subsets, $\mathcal{T}_1, \ldots, \mathcal{T}_k$, in this order. Focusing on this similarity, Bhat and Rouskas proposed an algorithm called depth first tour search (DFTS) to efficiently find a service path as the shortest path tour [13]. The DFTS algorithm, however, does not consider the resource constraints. In our previous work [14], we modeled the SC problem as the capacitated SPTP (CSPTP) and formulated it as an integer linear program (ILP) for the CSPTP-based SC. CSPTP is an extension of SPTP with constraints on both node and link capacities with real values. We also proposed the Lagrangian heuristic algorithm to solve the online CSPTP-based SC, where the SC orchestrator immediately serves a new SCR arriving at the NFV network, by considering the balance between optimality and computational complexity [15]. This algorithm, however, may not sufficiently work under dynamical demand change and/or network dynamics (e.g., temporal link failures) because it requires environmentally dependent parameter tuning.

ML-based networking has been attracting many researchers to realize the automatic network operation by solving various network optimization problems under uncertain environments [16]. In particular, graph neural networks (GNNs) have a capability to explore hidden representation in networks from the complex relationship between network traffic and topologies [17], [18], [19], [20], [21], [22]. In recent years, there are several studies for the combination of reinforcement learning (RL) and SC [7], [9], [10], [23]. They, however, did not sufficiently consider the following issues of CSPTP-related SC: (1) permitting the use of identical links as many times as required, (2) meeting the service chain requirements, (3) holding resource constraints, and (4) achieving resource allocation adaptive to demand and topology changes.

To tackle these problems, in the conference paper [1], we proposed a deep RL (DRL) framework with the GNN for the online CSPTP-based SC and partly demonstrated the fundamental characteristics of the proposed framework through numerical results using the NSFNET topology [24]. In this paper, we comprehensively evaluate the proposed framework by revealing its generalization capabilities against changes of service demand trend and network topology (temporal link failures or different networks). For this purpose, we first evaluate the performance of the proposed framework under a different topology, i.e., the SPRINT topology [25], from the viewpoint of the learning speed, adaptability to changes of service demand and topological change with link failures. We further investigate the applicability of a model learned in a certain network to other networks through evaluations using the 243 real-world network topologies.

The main contributions of the manuscript are as follows:

1) The proposed framework is an initial step toward the realization of the automatic network operation for CSPTP-based SC, which aims at accepting as many SCRs as possible even under the changes of service demand trend and network topology.
2) Through numerical results, we demonstrate that (1) the proposed framework achieves nearly optimal SC with higher learning speed compared to the conventional deep Q-network (DQN) based approach, (2) the proposed framework, when trained under a certain service demand trend, also performs well when confronted with changes in service demand, and (3) the proposed framework, when trained under a certain network topology, exhibits competitive performance with the ILP solutions across the majority of 243 real-world topologies, benefiting from the generalization capabilities of both DRL and GNN.

The rest of the manuscript is organized as follows. Section II gives the related work. In Section III, we introduce the some preliminaries, i.e., CSPTP-based SC, DRL, and GNN. In Section IV, we propose the DRL framework with the GNN for CSPTP-based SC. Section V shows the fundamental characteristics of the proposal. Finally, Section VI gives the conclusion and future work.

II. Related Work

A. Service Chaining Problem

SC is one of the challenging resource allocation problems that maps the VNFs and virtual links connecting them into physical nodes and links [2]. It tries to calculate an appropriate service path from an origin node to a destination node while executing VNFs under both the resource constraints and service chain requirements. Under various scenarios (e.g., wide-area network, mobile network, data center network, and cloud), researchers have addressed SC problems in terms of diverse aspects such as minimizing total the total delay of the path [7], [8], [9], [10], [13], [14], [15], streamlining the resource utilization [10], [11], [26], [27], [28], [29], maximizing the acceptance rate [30], and reducing the management cost [27], [31], [32], [33]. It is well-known that the SC problems belong to NP-hard problems [12]. To address this issue, there have been many studies on efficiently solving the SC problems with the help of several types of special network models: graph transformation [34], layered graph [26], [27], expanded network [30], and augmented network [14]. These approaches formulated the SC problems as ILPs using the special network models and developed heuristic algorithms to overcome the computational complexity.

The graph transformation, layered graph, and expanded network construct their special networks in a similar manner. Basically, they build a hierarchical network with M_c+1 copies (layers) of the original physical networks where M_c denotes the number of functions required by an SCR c. The identical nodes between two successive layers are connected with each other. As a result, we can establish a service path, which can sequentially execute the M_c functions in the required order, by finding a path from an origin node at the bottom layer to a destination node at the top layer. They, however, have to build the special networks tailored for each SCR if the number and/or order of functions are different among SCRs. Different from these network models, the augmented network model can efficiently and agilely handle the SC problem for arbitrary SCRs [14]. Considering this advantage, we adopt the augmented network in the proposed framework.

Another important aspect of SC is its similarity with SPTP. The SPTP aims at finding the shortest path from an origin to a destination while visiting at least one intermediate node from given disjoint node subsets in required order [35]. Bhat and Rouskas first pointed out the similarity between SC and SPTP. They also proposed the DFTS algorithm to find the shortest path tour without consideration of resource constraints [13]. In [28], Gao and Rouskas applied the game-theoretic approach to SPTP-related traffic steering for SC. Focusing on this similarity and the resource constraints on physical network, we modeled the SC problem as the CSPTP-based SC, where the CSPTP is an extended version of SPTP supporting general constraints on node and link capacities with real values [14]. In addition, we formulated the CSPTP-based SC as an ILP using the augmented network (i.e., *CSPTP-based ILP*) and developed a Lagrangian heuristic algorithm based on the DFTS algorithm to overcome the computational complexity [15].

However, these approaches [13], [14], [15], [28] basically cope with the SCR one by one in a myopic manner, which results in the lack of the adaptability to demand/network dynamics. In particular, the Lagrangian heuristics requires environmentally dependent parameter tuning. In this paper, we propose an ML-empowered SC, which can achieve effective resource allocation in response to the demand trend and network dynamics.

Function placement plays a crucial role in enhancing resource efficiency. Numerous studies have been conducted to address the challenges of service chaining and function placement (SCFP) problems [7], [14], [15], [23], [26], [27], [29], [31], [32], [33], [36]. The SCFP problem involves establishing the service path while deploying VNFs on physical nodes, considering resource constraints and service chain requirements. In [14], [15], [26], [27], [29], the authors proposed linear-programming based approaches to tackle the SCFP problem. In [7], [23], [31], [32], [33], [36], the authors proposed the ML-based SCFP solutions. In this paper, our primary focus is on the SC problem. However, in Section IV-D, we will assess the feasibility of the proposed scheme for addressing the SCFP problem.

B. Machine Learning for Networking

ML techniques have been applied to various domains in networking and expected to realize automated network optimization even under uncertain environments [16]. Specifically, GNNs have been one of the promising approaches to explore the hidden representation of network traffic and topologies [9], [10], [37].

There have been many studies employing ML techniques to SC [7], [7], [8], [9], [10], [23]. Pei et al. proposed both supervised and unsupervised learning based two-phase VNF selection and chaining algorithms for networks with SDN and NFV support [7]. In [8], Heo et al. proposed GNN-based SC employing the encoder-decoder model with teacher forcing to establish a service path such that the total service path delay is minimized. The supervised learning and teacher forcing require a large amount of labeled data but it is quite challenging to obtain them from actual networks in a real-time manner. Artificial data generation using enormous simulations is an alternative approach at the cost of time and effort. Considering these points, we employ the RL approach that trains a model using the target score (i.e., reward) instead of the labeled data.

There have been several studies applying RL techniques to SC [7], [9], [10], [11], [23], [36]. These studies can be mainly categorized into two methods: (1) path generation [9], [23], [36] and (2) path prediction [10], [11], [37]. The path generation method aims at efficiently deriving an appropriate service path from all possible candidates. In other word, it finds an appropriate service path from the large solution space, and thus it may be hard to achieve SC in a real-time manner, due to the high computational complexity. Chen et al. proposed quality of service (QoS) and quality of experience (QoE) aware SC based on RL to select the VNF instances executed in SDN and NFV enabled slices [23]. In [9], Heo et al. extended their previous model [8] by applying RL algorithms.

On the contrary, the path prediction method first enumerates a moderate number of path candidates and then selects an appropriate service path from them. Therefore, it can suppress the computational complexity by squeezing the solution space at the risk of degrading the solution diversity. Rafiq et al. proposed GNN-based SC in SDN to predict the optimal path that can achieve the delay-aware traffic steering [10]. Ning et al. applied DRL to SC to optimize both end-to-end SC performance and overall network resource utilization by determining an appropriate service path from path candidates [11]. Almasan et al. applied message passing neural networks (MPNNs) to the DRL framework to solve the minimum cost flow problem in optical networks and showed the generalization capabilities of MPNN based GNN over different topologies [37]. Note that these approaches do not consider the possibility that an identical link would be used multiple times in a service path.

In this paper, to realize the real-time SC, we adopt the path prediction method. More specifically, we consider an RL model to select an appropriate service path from the path tour candidates with the solution optimality, which is derived by the extended version of the DFTS algorithm for finding the shortest path tour. In addition, inspired by the approach in [37], we propose a DRL with GNN framework to solve SC, which is more difficult than the conventional routing problem considered in [37]. The proposed framework aims at realizing (1) adaptive resource allocation based on the learning of demand trend and (2) generalization capabilities against temporal topology changes due to link failures and different physical topologies, thanks to both the DRL and GNN.

III. Preliminaries

In this section, we briefly introduce the preliminaries of the proposed framework from the viewpoint of system model, CSPTP-based SC, DRL, and GNN, respectively. Table I summarizes the notations used in this paper.

A. System Model

In this paper, we consider the NFV/SDN collaborative system model, as in [6], [7], [8], [9], [10], [11]. NFV consists of three main components as follows: (1) VNF, (2) NFV infrastructure (NFVI), and (3) NFV management and orchestrator (NFV MANO) [3], [4]. VNF represents the software

TABLE I
NOTATIONS

Symbol	Description
	Service chaining related notations
G	Physical network $G = (\mathcal{V}, \mathcal{E}, \mathcal{X})$
$\mathcal{V}$	Set of physical nodes, $V = \lvert\mathcal{V}\rvert$
$\mathcal{V}_{\mathrm{VNF}}$	Set of VNF-enabled nodes, $\mathcal{V}_{\mathrm{VNF}} \subseteq \mathcal{V}$, $V_{\mathrm{VNF}} = \lvert\mathcal{V}_{\mathrm{VNF}}\rvert$
$\mathcal{E}$	Set of physical links, $E = \lvert\mathcal{E}\rvert$
$\mathcal{X}$	Set of features of physical links
$\mathcal{F}$	Set of functions, $\mathcal{F} = \{f_1, \ldots, f_F\}$, $F = \lvert\mathcal{F}\rvert$
$\mathcal{F}_i$	Set of functions possessed by physical node $i \in \mathcal{V}$
$\mathcal{V}_f$	Set of physical nodes having function $f \in \mathcal{F}$ ($V_f = \lvert\mathcal{V}_f\rvert$)
c	Service chain request with origin o_c and destination d_c
r_c	Service chain requirements of request c
$\mathcal{R}_c$	Sequence of functions required by c, $(f_{c,1}, \ldots, f_{c,M_c})$
M_c	Number of functions required by c
w_c	Service path for c, $w_c = (w_{c,1}, \ldots, w_{c,M_c+1})$
$w_{c,m}$	m-th subpath of w_c
G^+	Augmented network $G^+ = (\mathcal{V}^+, \mathcal{E}^+, \mathcal{X}^+)$
$\mathcal{V}^+$	Set of nodes in augmented network, $\mathcal{V}^+ = \mathcal{V} \cup \hat{\mathcal{V}}$
$\mathcal{E}^+$	Set of links in augmented network, $\mathcal{E}^+ = \mathcal{E} \cup \hat{\mathcal{E}}^{\mathrm{in}} \cup \hat{\mathcal{E}}^{\mathrm{out}}$
$\mathcal{X}^+$	Set of link features in augmented network
$\hat{\mathcal{V}}$	Set of imaginary nodes, $\hat{\mathcal{V}} = \{\hat{v}_f\}_{f\in\mathcal{F}}$, where imaginary node $\hat{v}_f$ is responsible for function f
$\hat{\mathcal{E}}^{\mathrm{in}}$	Set of links incoming to imaginary nodes, $\hat{\mathcal{E}}^{\mathrm{in}} = \{(v_f, \hat{v}_f)\}_{f\in\mathcal{F}, v_f\in\mathcal{V}_f}$
$\hat{\mathcal{E}}^{\mathrm{out}}$	Set of links outgoing from imaginary nodes, $\hat{\mathcal{E}}^{\mathrm{out}} = \{(\hat{v}_f, v_f)\}_{f\in\mathcal{F}, v_f\in\mathcal{V}_f}$
$\mathcal{E}_{w_c}$	Multiset of physical links included in w_c
$\hat{\mathcal{E}}^{\mathrm{in}}_{w_c}$	Multiset of incoming virtual links included in w_c
$\hat{\mathcal{E}}^{\mathrm{out}}_{w_c}$	Multiset of outgoing virtual links included in w_c
b_c	Bandwidth requirement of c
$p_{c,f_{c,m}}$	Processing requirement of $f_{c,m} \in \mathcal{R}_c$
$B_{i,j}$	Residual capacity of link (i, j) at c's arrival
$P_{\hat{v}_f,i}$	Residual capacity of node i to execute f at c's arrival
$\hat{B}_{i,j}$	Original capacity of link (i, j)
$\hat{P}_{\hat{v}_f,i}$	Original capacity of node i to execute f
	DRL and GNN related notations
$a, \mathcal{A}$	Action and set of actions ($a \in \mathcal{A}$)
$s, \mathcal{S}$	Network state and set of network states
r, R	Reward and cumulative reward
π	Agent strategy, $\pi : \mathcal{S} \to \mathcal{A}$
L, T	Number of episodes and number of training iterations
I	Training interval of GNN
$\mathbf{A}$	Adjacency matrix
$\mathbf{D}$	Degree matrix
$\mathbf{I}$	Identity matrix
$\mathbf{X}$	Feature matrix, $\mathbf{X} = [\boldsymbol{x}_1 \ldots, \boldsymbol{x}_{\lvert\mathcal{E}^+\rvert}]^{\mathrm{T}}$
$\boldsymbol{x}_e$	Feature vector of link e, $\boldsymbol{x}_e = (x_{e,1}, \ldots, x_{e,D})$
D	Number of features
$\mathbf{X}^l$	Feature matrix at l-th layer in GCN
$\Theta^{(l)}$	Learnable weight matrix at lth layer in GCN
$\sigma(\cdot)$	General element-wise nonlinear activation function
$\mathbf{T}$	Transition matrix in GDC
η_n	Weighting coefficient for $\mathbf{T}^n$
$\mathbf{S}, \tilde{\mathbf{S}}$	Diffusion matrix and sparsified diffusion matrix
Q	Set of pairs of action and q-value
$Q(s, a)$	q-value of taking action a at state s
$\gamma,\ \alpha$	Discount rate and learning rate
	Proposed scheme related notations
$\mathcal{W}_c$	Set of service path candidates for c, $\mathcal{W}_c = \{w_c^1, \ldots w_c^K\}$
w_c^k	kth service path candidate for c
K	Number of service path candidates
$u_{c,e}$	Link e's utilization required to support service path w_c
$x_{e,1}$	Number of times that link e is used in w_c
$x_{e,2}$	Bandwidth/processing capacity requirement of link e in w_c
$x_{e,3}$	Link e's utilization required to support service path w_c, $u_{c,e}$
$x_{e,4}$	Betweenness centrality of link $e \in \mathcal{E}^+$
$x_{e,5}$	Residual capacity of link $e \in \mathcal{E}^+$
C_{accept}	Set of accepted SCRs, $C_{\mathrm{accept}} = \lvert C_{\mathrm{accept}}\rvert$
B_{accept}	Total amount of incoming traffic of accepted SCRs

functionality responsible for a specific network function, which is composed of one or more VNF components managed by an element management system (EMS). NFVI is the virtual resources logically partitioned from physical resources. NFV MANO is comprised of (1) a virtualized infrastructure manager (VIM), which controls, manages, and monitors NFVI resources, (2) a VNF manager (VNFM), which orchestrates and manages VNFs, and (3) an NFV orchestrator (NFVO)

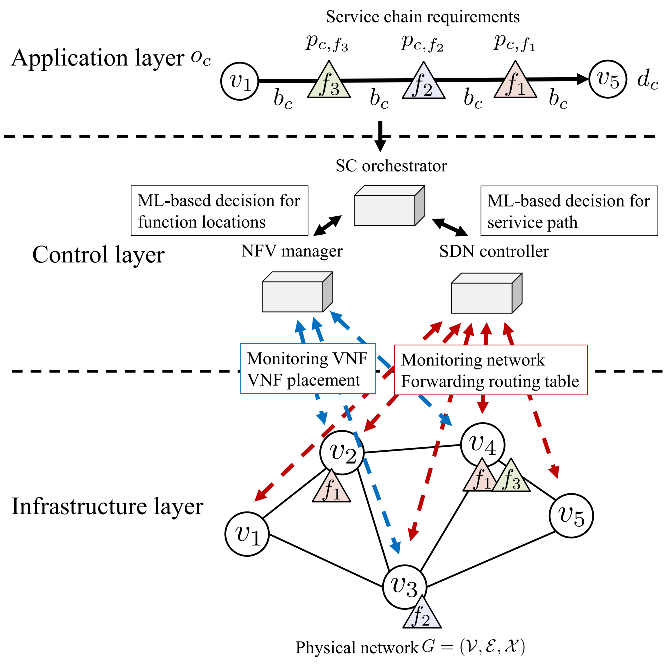


Fig. 1. System model.

responsible for the lifecycle management of network services. SDN consists of three layers as follows: (1) an application plane, (2) a control plane, and (3) a data plane [5]. The application plane handles network services and communicates with the SDN controller in the control plane through northbound interfaces. The control plane encompasses centralized controllers, i.e., SDN controllers, which control and manage the network devices in the data plane through southbound interfaces, following the requests from the application plane. In the data plane, network devices forward and steer traffic based on predefined rules installed by the SDN controllers.

Inspired by the system model presented in [6], we design a NFV/SDN collaborative system for SC, which consists of three layers: (1) an application layer, (2) a control layer, and (3) an infrastructure layer, as shown in Fig. 1. In the application layer, individual SCRs containing service chain requirements are generated by applications. The detail of the SCR will be explained in Section III-B1. Moving to the control layer, the SC orchestrator receives each SCR and makes an ML-based decision for an appropriate service path (and function locations) that should adhere to the defined service chain requirements and the resource constraints extracted by the NFV manager and the SDN controller. The NFV manager is responsible for NFVO and VNFM, thereby overseeing the lifecycle management of VNFs. It actively monitors the VNFs and orchestrates their deployment on the physical nodes. Meanwhile, the SDN controller functions as VIM, actively managing network resources. It collects network features and effectively routes traffic based on the service path determined by the SC orchestrator. Detailed information regarding the SC orchestrator will be presented in Section IV. Finally, in the infrastructure layer, physical nodes and links are located in the wide-area network. Further details regarding the physical network will be shown in Section III-B2.

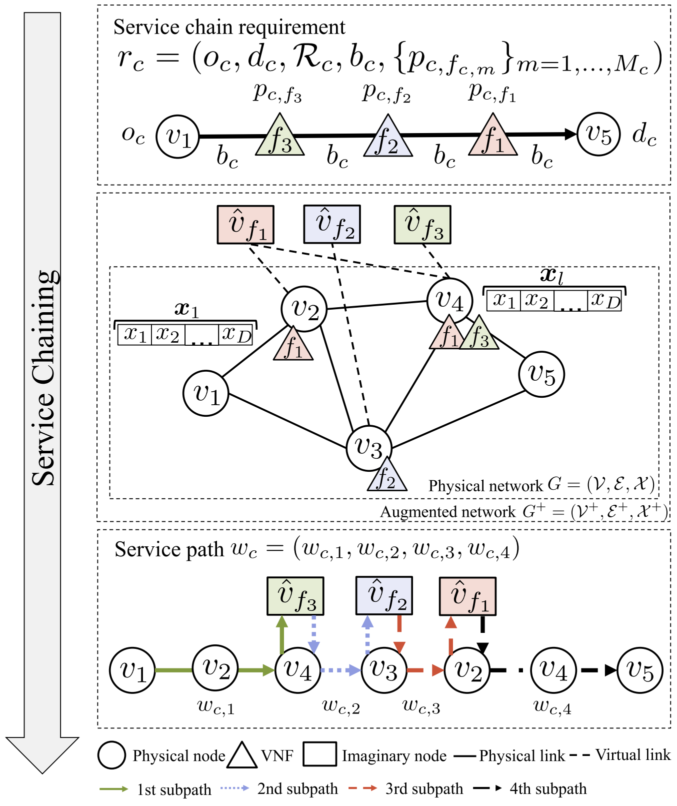

Service Chaining
Service chain requirement
$r_c = (o_c, d_c, \mathcal{R}_c, b_c, \{p_{c,f_{c,m}}\}_{m=1,\dots,M_c})$
p_{c,f_3} p_{c,f_2} p_{c,f_1}
o_c v_1 f_3 f_2 f_1 v_5 d_c
b_c b_c b_c b_c
$\hat{v}_{f_1}$ $\hat{v}_{f_2}$ $\hat{v}_{f_3}$
$\boldsymbol{x}_1$ x_1 x_2 ... x_D
$\boldsymbol{x}_l$ x_1 x_2 ... x_D
v_1 v_2 v_3 v_4 v_5 f_1 f_2 f_3
Physical network $G = (\mathcal{V}, \mathcal{E}, \mathcal{X})$
Augmented network $G^+ = (\mathcal{V}^+, \mathcal{E}^+, \mathcal{X}^+)$
Service path $w_c = (w_{c,1}, w_{c,2}, w_{c,3}, w_{c,4})$
$\hat{v}_{f_3}$ $\hat{v}_{f_2}$ $\hat{v}_{f_1}$
v_1 v_2 v_4 v_3 v_2 v_4 v_5
$w_{c,1}$ $w_{c,2}$ $w_{c,3}$ $w_{c,4}$
Physical node
VNF
Imaginary node
Physical link
Virtual link
1st subpath
2nd subpath
3rd subpath
4th subpath

Fig. 2. Overview of CSPTP-based SC.

B. CSPTP-Based Service Chaining

In this paper, we consider the system model used in [14]. Fig. 2 illustrates the overview of the CSPTP-based SC.

1) Service Chain Request: We assume the online SC where the SC orchestrator serves a newly incoming SCR c immediately after its arrival. As shown in the top layer of Fig. 2, each SCR c has the service chain requirements $r_c = (o_c, d_c, \mathcal{R}_c, b_c, \{p_{c,f_{c,m}}\}_{k=1,\dots,M_c})$ where o_c and d_c denote an origin node and a destination node, respectively. $\mathcal{R}_c$ represents a sequence $(f_{c,1}, \dots, f_{c,M_c})$ of M_c *functions* in required order. Let b_c and $p_{c,f_{c,m}}$ be the required bit rate and the processing resources required for executing the mth function $f_{c,m}$ at a physical node, respectively.

2) Physical Network: A *physical network* is defined as a directed graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, where $\mathcal{V}$ (resp. $\mathcal{E}$) is a set of *physical nodes (resp. links)*. Let $\mathcal{X}$ denote a set of *features* of physical links, i.e., $\mathcal{X} = \{\boldsymbol{x}_e\}_{\forall e \in \mathcal{E}}$, where $\boldsymbol{x}_e$ represents a vector of $D > 0$ features of physical link e, i.e., $\boldsymbol{x}_e = (x_{e,1}, \dots, x_{e,D})$. The NFV/SDN collaborative system supports a set of F distinct functions, $\mathcal{F} = \{f_1, \dots, f_F\}$, and consists of two types of the physical nodes: *VNF-enabled nodes* $\mathcal{V}_{\text{VNF}}$ and forwarding devices (i.e., routers and switches). Each VNF-enabled node $i \in \mathcal{V}_{\text{VNF}}$ is a commodity server and accommodates one or more virtual machines corresponding to functions $\mathcal{F}_i \subseteq \mathcal{F}$. Each function $f \in \mathcal{F}$ can be installed at part of VNF-enabled nodes, i.e., $\mathcal{V}_f \subseteq \mathcal{V}_{\text{VNF}}$.

3) Augmented Network: To handle CSPTP-based SC, the *augmented network* $G^+ = (\mathcal{V}^+, \mathcal{E}^+, \mathcal{X}^+)$ is constructed by extending the physical network G with *imaginary nodes* $\hat{\mathcal{V}}$ and *virtual links* $\hat{\mathcal{E}}^{\text{in}} \cup \hat{\mathcal{E}}^{\text{out}}$ where $\mathcal{V}^+ = \mathcal{V} \cup \hat{\mathcal{V}}$ and $\mathcal{E}^+ = \mathcal{E} \cup \hat{\mathcal{E}}^{\text{in}} \cup \hat{\mathcal{E}}^{\text{out}}$. $\mathcal{X}^+$ denotes a set of features on physical and virtual links, i.e., $\mathcal{X}^+ = \{\boldsymbol{x}_e\}_{e \in \mathcal{E}^+}$. An imaginary node $\hat{v}_{c,f_{c,m}} \in \hat{\mathcal{V}}$ is responsible for function $f_{c,m}$ and is connected to VNF-enabled node(s) supporting $f_{c,m}$. Links incoming to (resp. outgoing from) imaginary node $\hat{v}_f$, called virtual links, are defined as $\hat{\mathcal{E}}^{\text{in}}$ (resp. $\hat{\mathcal{E}}^{\text{out}}$). Note that $\hat{\mathcal{E}}^{\text{in}} = \{(v_f, \hat{v}_f)\}_{f \in \mathcal{F}, v_f \in \mathcal{V}_f}$ (resp. $\hat{\mathcal{E}}^{\text{out}} = \{(\hat{v}_f, v_f)\}_{f \in \mathcal{F}, v_f \in \mathcal{V}_f}$). The virtual link $(\hat{v}_f, v) \in \hat{\mathcal{E}}^{\text{out}}$ indicates that the VNF-enabled node $v \in \mathcal{V}_f$ supports the function f. Each virtual link $(\hat{v}_f, v)$ (resp. physical link (i,j)) has the residual processing capacity $P_{\hat{v}_f, v}$ of physical node v for executing function f (resp. residual link capacity $B_{i,j}$) at the arrival of SCR c. The middle layer of Fig. 2 illustrates an example of the augmented network.

4) Service Path: Thanks to the augmented network, the *service path* w_c with origin o_c, destination d_c, and required functions $\mathcal{R}_c$ can be decomposed into a sequence of $M_c + 1$ subpaths, i.e., $w_c = (w_{c,1}, \dots, w_{c,M_c+1})$. The pair $(o_{c,m}, d_{c,m})$ of origin and destination nodes of the mth subpath $w_{c,m}$ is given by $(o_c, \hat{v}_{f_{c,1}})$ for $m = 1$, $(\hat{v}_{f_{c,m-1}}, \hat{v}_{f_{c,m}})$ for $m = 2, \dots, M_c$, and $(\hat{v}_{f_{c,M_c}}, d_c)$ for $m = M_c + 1$. Note that selecting the virtual link in the service path determines the physical node on which the corresponding function is conducted. Each subpath does not contain any loop while the entire service path may have loop(s). As a result, a certain link may be used more than once in the service path. We define $\mathcal{E}^+_{w_c}$ as a multiset of links included in w_c, i.e., $\mathcal{E}^+_{w_c} = \mathcal{E}_{w_c} \cup \hat{\mathcal{E}}^{\text{in}}_{w_c} \cup \hat{\mathcal{E}}^{\text{out}}_{w_c}$, where $\mathcal{E}_{w_c}$ is a multiset of physical links included in w_c and $\hat{\mathcal{E}}^{\text{in}}_{w_c}$ (resp. $\hat{\mathcal{E}}^{\text{out}}_{w_c}$) is a multiset of incoming (resp. outgoing) virtual links included in w_c. Here, a multiset is a set that allows multiple instances for each of its elements. The bottom layer of Fig. 2 shows an example of the service path.

C. Deep Reinforcement Learning

RL aims at learning a long-term strategy (i.e., policy) to solve an optimization problem under a certain environment, which is defined by a set $\mathcal{S}$ of states [38]. Given a state $s \in \mathcal{S}$, an agent takes an action $a \in \mathcal{A}$ at the state s according to the current policy $\pi : \mathcal{S} \to \mathcal{A}$ learned so far. After taking an action a at the state s, the agent will move to a next state s' and obtain a reward r with a probability $\Pr(s', r|s, a)$. The agent aims at acquiring a strategy that maximizes the cumulative reward R at the end of an episode (trial), which starts from an arbitrary initial state followed by multiple state transitions and ends with a certain stop condition, e.g., reaching a predefined number of steps. Finding the optimal strategy can be modeled as a Markov decision process (MDP) [39].

Q-learning is an RL algorithm to solve MDP by making the agent learn an optimal policy $\pi : \mathcal{S} \to \mathcal{A}$. It maintains a table with the size of $|\mathcal{S}| \times |\mathcal{A}|$, where (s, a)th element is initialized as zero or a random value and updated with a q-value for the combination of state s and action a. If the agent takes an action a at a state s according to the current policy π, it will update the value of (s, a)th element by $Q(s, a)$, which is the expected cumulative reward after performing the action

a at the state s under the assumption that the agent will follow the current policy π in the rest of episode. Here, the q-value $Q(s, a)$ is updated according to the rule based on the Bellman equation [40]:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(r + \gamma \max_{a' \in \mathcal{A}} Q(s',a')\right),$$

where γ $(0 \leq \gamma \leq 1)$ denotes a discount rate indicating the importance of the future reward and α $(0 < \alpha \leq 1)$ is a learning rate.

One of the potential drawbacks of Q-learning is the scalability against the size of state and action space. If the solution space, i.e., $\mathcal{S} \times \mathcal{A}$, becomes large, it is difficult for the agent to explore all the possible combinations of states and actions, which would degrade the optimality of learned policy. DQN can solve this potential drawback by approximating the q-value function using a deep neural network (DNN) and learning it through observed states and actions [41]. In DQN, the q-values for unobserved states and actions are estimated by the DNN learned through observed states and actions, with the help of its generalization capabilities. By taking advantage of DNN, the DRL agent is expected to take an appropriate action even at a state that it has not experienced yet. The state transition information $\{s, a, r, s'\}$ is stored in a memory called an experience replay buffer, which is used for training DNNs. The DRL agent trains neural networks by randomly sampling from the experience replay buffer to cope with the time-dependency problem.

D. Graph Neural Network

GNNs are deep learning based methods to operate the graph domain [17], [18], [19], [20], [21], [22]. Given the graph structure and node feature information as inputs, a GNN outputs the node, edge, or graph-level representation by graph convolution operation in the spectral or spatial domain. Message passing neural networks (MPNNs) are a well-known type of GNNs, which is a unified framework for the graph convolution operations (i.e., aggregation, update, and readout) in the spatial domain [22]. In MPNNs, each node in the graph initially has its own features. Then, it collects the features from the neighbors and aggregates them into a message. It further combines the message with its own features and updates its features as the hidden embedding. These operations are repeated along with multiple layers of MPNNs. The output of the final layer defines the node-level representation, i.e., embedding of each node, and it can generate a graph-level representation through the readout operation.

Graph convolutional networks (GCNs) are one of the most popular baseline GNN models and employ the first-order neighboring aggregation and the self-loop update [18]. GCN with the renormalization trick can be defined as the following layer-wise aggregation and update operations:

$$\mathbf{X}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}^{(l)}\mathbf{\Theta}^{(l)}\right). \quad (1)$$

Here, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I} \in \mathbb{R}^{N \times N}$ is the adjacent matrix with self loops where $\mathbf{A} \in \{0,1\}^{N \times N}$ is the original adjacent matrix of the undirected graph G with N nodes and $\mathbf{I}$ is the identity matrix. $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$ is the degree matrix of $\tilde{\mathbf{A}}$, where $\mathbf{D}$ denotes the degree matrix of $\mathbf{A}$. $\mathbf{X}^{(l)} \in \mathbb{R}^{N \times D}$ represents a feature matrix at the lth GCN layer, where $\mathbf{X}^{(0)} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\top$ indicates an original feature matrix. $\mathbf{\Theta}^{(l)}$ indicates a learnable weight matrix for the lth layer, and $\sigma(\cdot)$ is a general element-wise nonlinear activation function, e.g., rectified linear unit function (ReLU) [42].

Klicpera et al. proposed graph diffusion convolution (GDC) to generalize the graph convolution by considering the impact of both direct and indirect neighbors [19]. GDC replaces the adjacency matrix $\mathbf{A}$ with the following diffusion matrix $\mathbf{S}$:

$$\mathbf{S} = \sum_{n=0}^{\infty} \eta_n \mathbf{T}^n, \quad (2)$$

where $\mathbf{T} \in \mathbb{R}^{N \times N}$ is a transition matrix whose (i,j)th element means the transition probability from node i to node j. $\mathbf{T}^n$ gives the n-step transition probabilities and $\eta_n > 0$ is the weighting coefficient for $\mathbf{T}^n$. In [19], the authors showed some special cases of graph diffusion, i.e., personalized PageRank [43], heat kernel [44], and GCN [18]. If the diffusion matrix $\mathbf{S}$ is dense, the sparsified diffusion matrix $\tilde{\mathbf{S}}$ was used to obtain the spatial locality by removing links with small values of $\mathbf{S}$ in a simple manner, e.g., top-k-based sparsification or threshold-based sparsification.

IV. Proposed Scheme

A. Overview

In this paper, inspired by the DRL with GNN architecture for network routing problems [37], we propose the DRL based framework with a GNN for the CSPTP-based SC. The SC problem as CSPTP is more challenging than the conventional routing problem as the shortest path problem. The agent, i.e., SDN controller, aims at accepting as many SCRs as possible, which will be achieved by the minimization of the overall physical and virtual link utilization in the physical network. The proposed DRL agent is realized by the double-DQN algorithm [45], where the q-value function is modeled by a GNN. (We will give the DRL agent design in Section IV-C and the GNN architecture in Section IV-C3.)

Fig. 3 illustrates the overview of the proposed DRL based framework with a GNN for the CSPTP-based SC. At each time step, the agent (i.e., SDN controller) monitors the environment (i.e., physical network) and obtains both a network state and an SCR c as inputs from the environment (Step 1 in Fig. 3). Here, the network state is represented by the features of each link in the augmented network, which will be described in Section IV-B. Next, the agent enumerates the service path candidates $\mathcal{W}_c$, i.e., an action set $\mathcal{A}$, using K-DFTS algorithm (Step 2 in Fig. 3). We will show the details of the action set in Section IV-C2. For each service path candidate (i.e., action), it generates an SC-embedded state from the current state $s \in \mathcal{S}$ by concatenating the network-related features and SC-related ones (Step 3 in Fig. 3) and generates a sparsified diffusion matrix $\tilde{\mathbf{S}}$ with the help of GDC (Step 4 in Fig. 3). It computes the q-value of the SC-embedded (action-embedded) state (Step 5 in Fig. 3). The details of the agent operation will be described in Section IV-C1. Note that the existing

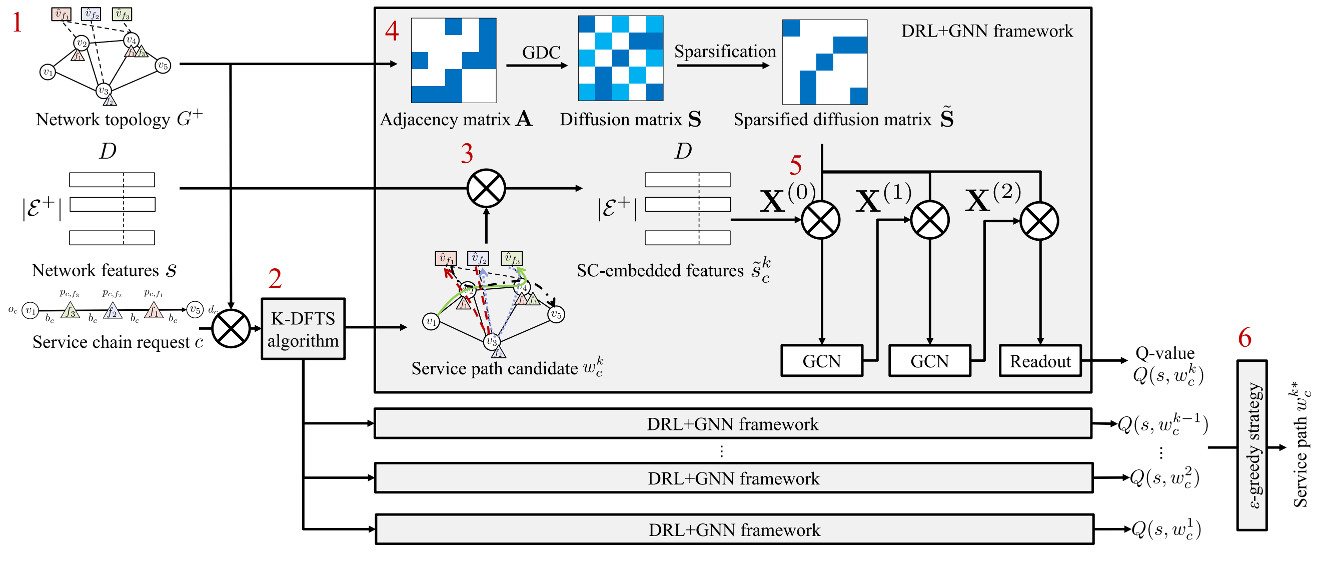

1
Network topology G^+
D
$|\mathcal{E}^+|$
Network features $\mathcal{S}$
Service chain request c
2
K-DFTS algorithm
3
4
Adjacency matrix $\mathbf{A}$
GDC
Diffusion matrix $\mathbf{S}$
Sparsification
Sparsified diffusion matrix $\tilde{\mathbf{S}}$
DRL+GNN framework
SC-embedded features $\tilde{s}_c^k$
Service path candidate w_c^k
5
$\mathbf{X}^{(0)}$
$\mathbf{X}^{(1)}$
$\mathbf{X}^{(2)}$
GCN
GCN
Readout
Q-value $Q(s, w_c^k)$
$Q(s, w_c^{k-1})$
$Q(s, w_c^2)$
$Q(s, w_c^1)$
6
ϵ-greedy strategy
Service path w_c^{k*}

Fig. 3. DRL framework with GNN for the CSPTP-based SC.

work for the conventional routing problem in [37] adopts a different GNN approach, i.e., MPNN. Finally it performs an appropriate action $a \in \mathcal{A}$, i.e., selecting an appropriate service path, according to the policy π (Step 6 in Fig. 3), and then obtains the reward r, the next SCR c', and the next state $s' \in \mathcal{S}$ from the environment (back to Step 1 in Fig. 3).

B. Environment

In this paper, we consider the environment as the augmented network with link features, as shown in the middle layer of Fig. 2. More specifically, the network state s is defined as the feature matrix $\mathbf{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_{|\mathcal{E}^+|}]^\mathsf{T}$ where $\boldsymbol{x}_e$ is a $D = 5$ dimensional feature vector of physical/virtual link $e \in \mathcal{E}^+$, i.e., $\boldsymbol{x}_e = (x_{e,1}, \dots, x_{e,5})$. Note that the features of physical (resp. virtual) link are associated with the network (resp. computing) resources. The feature vector $\boldsymbol{x}_e$ is composed of the SC-related features (i.e., $x_{e,1}$, $x_{e,2}$, and $x_{e,3}$) and the network-related features (i.e., $x_{e,4}$ and $x_{e,5}$). The SC-related features are calculated per service path candidate $w_c \in \mathcal{W}_c$ to evaluate its deployment cost in terms of resource usage. $x_{e,1}$ is the number of times that the link e is used in w_c. Please note that $x_{e,1}$ can be more than one if the service path candidate w_c has loop(s), which makes the problem more difficult than the conventional routing problem [37]. $x_{e,2}$ is SCR c's bandwidth requirement b_c (resp. processing capacity requirement $p_{c,f_{c,m}}$) for the physical (resp. virtual) link e, which is demanded by the SCR c. $x_{e,3}$ is the link e's utilization $u_{c,e}$ resulting from the establishment of w_c, which considers the possibility that an identical link would be used multiple times in a service path. Focusing on the SC-related features only for the links used in w_c, we set $x_{e,1}$, $x_{e,2}$, and $x_{e,3}$ to be zero for the links unused in w_c. (Similar assumption is also used in [37].)

The network-related features are used to evaluate the overall utilization of physical network, which will contribute to saving the network resources for future requests. For this purpose, we apply the link betweenness centrality [46] and the residual capacity of link e as the network-related features $x_{e,4}$ and $x_{e,5}$, respectively. Note that these features are also used in [37].

C. Agent Design

1) Agent Operation: The agent operates through the interactions with the environment. We assume that the agent learns the optimal policy through $T \geq 1$ *training iterations*, each of which consists of $L \geq 1$ *episodes*. Algorithm 1 presents a pseudocode describing the proposed agent behavior in one episode of the τth training iteration ($\tau = 1, \dots, T$). At the beginning of the episode, the environment *env* is initialized by calling the INIT() function, which also generates a new SCR c with the service chain requirements r_c (line 1). At the same time, the cumulative reward R is set to be zero (line 2).

Algorithm 1 executes the following procedures as long as the agent succeeds in allocating a service path to a new SCR c, i.e., the corresponding binary flag *allocated* is true (lines 3–16). Since considering all possible service path candidates will result in a highly dimensional action space, the action set is limited to K service path candidates as in [37]. The agent calculates the set of K service path candidates, $\mathcal{W}_c = \{w_c^1, \dots, w_c^k\}$, by calling the K-DFTS() function (line 4). We will describe the details of K-DFTS() function in Section IV-C2. Note that symbols $\mathcal{A}$ and $\mathcal{W}_c$ will be used interchangeably. We also initialize a set $\boldsymbol{Q}$ of each pair of action a and its yielding q-value $Q(s, a)$ to an empty set (line 5).

For each service path candidate $w_c^k \in \mathcal{W}_c$, the agent computes the corresponding q-value using the GNN (lines 6–8). More specifically, the agent first generates the SC-embedded state $\tilde{s}_c^k$ using the ALLOCATE-SCR() function (line 7). As described in Section IV-B, $\tilde{s}_c^k$ is represented by the feature vector $\boldsymbol{x}_e$ of each link e. Given the state $\tilde{s}_c^k$ as the input, the agent then computes the corresponding q-value $Q(\tilde{s}_c^k, w_c^k)$ by calling the GET-Q-VAL-FROM-GNN() function and adds the new element $(w_c^k, Q(\tilde{s}_c^k, w_c^k))$ to $\boldsymbol{Q}$ (line 8). The details of

Algorithm 1 Agent Operation

```
Require: Agent $agent$, environment $env$, augmented network $G^+$, the number $K$ of actions, training iteration id $\tau$, training interval $M$.
1:  $s, c, r_c \leftarrow \text{INIT}(env)$
2:  $R \leftarrow 0$
3:  do
4:      $\mathcal{W}_c \leftarrow \text{K-DFTS}(K, r_c)$
5:      $\mathcal{Q} \leftarrow \emptyset$
6:      for $w_c^k \in \mathcal{W}_c$ do
7:          $\tilde{s}_c^k \leftarrow \text{ALLOCATE-SCR}(env, c, r_c, w_c^k)$
8:          $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(w_c^k, \text{GET-Q-VAL-FROM-GNN}(\tilde{s}_c^k, w_c^k)\}$
9:      $w_c^{k'} \leftarrow \text{EPSILON-GREEDY}(\mathcal{Q}, \varepsilon)$
10:     $r, allocated, s', c', r'_c \leftarrow \text{STEP}(env, s, w_c^{k'})$
11:     $R \leftarrow R + r$
12:     $\text{MEMORIZE}(agent, \{s, a, r, s'\})$
13:     if $\tau \mod I = 0$ then
14:         $\text{TRAIN-GNN-USING-REPLAY-BUFFER}(agent)$
15:     $s, c, r_c \leftarrow s', c', r'_c$
16: while $allocated$ = true
```

the GET-Q-VAL-FROM-GNN() function will be explained in Section IV-C3.

Next, the agent selects a service path candidate $w_c^{k'}$ from $\mathcal{W}_c$ according to $\mathcal{Q}$ and the ε-greedy exploration strategy [38] by calling the EPSILON-GREEDY() function (line 9). In the STEP() function, the agent tries to apply the service path candidate $w_c^{k'}$ to the physical network and then obtains the reward r, the binary flag *allocated*, the next state s', and the next SCR c' with r'_c from the environment (line 10). Here, we design the reward r after selecting $w_c^{k'}$ such that it should be nonnegative and becomes large in case of low utilization of network resources:

$$r = \omega_1 \exp\left(-\sum_{e \in \mathcal{E}_{w_c^{k'}}} u_{c,e}\right) + \omega_2 \exp\left(-\sum_{\hat{e} \in \hat{\mathcal{E}}^{\text{out}}_{w_c^{k'}}} u_{c,\hat{e}}\right),$$

where the first (resp. second) term is related to the usage degree of physical links (resp. virtual links) in $w_c^{k'}$, $\omega_1 > 0$ (resp. $\omega_2 > 0$) is the corresponding weighting parameter. Recall that $\mathcal{E}_{w_c^{k'}}$ (resp. $\hat{\mathcal{E}}^{\text{out}}_{w_c^{k'}}$) is a multiset of physical links (resp. outgoing virtual links) in $w_c^{k'}$ since an identical link would be used multiple times in the service path $w_c^{k'}$. Note that the cumulative reward R is defined as the sum of reward r during one episode.

The agent updates the cumulative reward R (line 11) and stores the transition (experience), i.e., $\{s, a, r, s'\}$, into the experience replay buffer (line 12). The stored transition will be used to train the GNN by executing the TRAIN-GNN-USING-REPLAY-BUFFER() function every $I \geq 1$ training iterations (lines 13–14). The GNN model is trained such that a loss function $L(\boldsymbol{\Theta})$ with the learnable weight matrix

Algorithm 2 K-DFTS Algorithm

```
Require: Augmented network $G^+$, the number $K$ of path candidates, service chain requirement $r_c$.
Ensure: Service path candidates $\mathcal{W}_c$.
1: $\mathcal{W}_c \leftarrow \emptyset$
2: for $k = 1$ to $K$ do
3:     $w_{c,k} \leftarrow \text{DFTS}(G^+, r_c)$
4:     if $w_{c,k} = \emptyset$ then return $\mathcal{W}_c$
5:     $\mathcal{W}_c \leftarrow \mathcal{W}_c \cup w_{c,k}$
6:     $G^+ \leftarrow \text{REMOVE-LINK}(G^+, w_{c,k})$
   return $\mathcal{W}_c$
```

$\boldsymbol{\Theta}$ approaches to zero by using the samples $\mathcal{Z}$ randomly chosen from the experience reply buffer. $L(\boldsymbol{\Theta})$ is defined as follows:

$$L(\boldsymbol{\Theta}) = \sum_{\{s,a,r,s'\} \in \mathcal{Z}} \left(Q(s, a|\boldsymbol{\Theta}) - \left(r + \gamma \max_{a' \in \mathcal{A}} Q\left(s', a' \mid \boldsymbol{\Theta}\right)\right)\right)^2 + \rho E_{\text{L1}}(\boldsymbol{\Theta}),$$

where the first term is the mean squared error between the estimated q-value and observed one. The second term indicates L1 regularization penalty to prevent overfitting, where $E_{\text{L1}}(\Theta)$ is the L1 regularization and $\rho > 0$ is a weighting parameter.

2) Action Set: To obtain the action set, i.e., K service path candidates $\mathcal{W}_c$, we propose a K-DFTS algorithm by extending the DFTS algorithm [13]. K is expected to be a moderate value, e.g., 5, to hold the balance between computational complexity and flexibility of steering traffic. In addition, the service path candidates are expected to be as exclusive as possible with each other to avoid making specific physical nodes/links highly congested. Algorithm 2 presents the K-DFTS algorithm. The agent first initializes the service path candidates $\mathcal{W}_c$ with the empty set (line 1) and then repeats the following procedures (lines 2–6). It calculates the first service path candidate $w_{c,k}$ under G^+ and r_c by calling DFTS() function [13] (line 3). Since saving the network resources leads to accepting more SCRs in future, we design the cost of each link (i,j) as $b_c/B_{i,j}$, $p_{c,f}/P_{i,j}$, and zero in case of $(i,j) \in \mathcal{E}$, $(i,j) \in \hat{\mathcal{E}}^{\text{out}}$, and $(i,j) \in \hat{\mathcal{E}}^{\text{in}}$, respectively. If the service path $w_{c,k}$ cannot be found, it returns the current service path candidates $\mathcal{W}_c$ (line 4). Otherwise, the service path $w_{c,k}$ is added into $\mathcal{W}_c$ (line 5). In addition, it updates the augmented network G^+ by removing a physical/virtual link with the highest utilization in the service path candidates selected so far by calling REMOVE-LINK() function (line 6), and calculates the next service path candidate in the same way. It continues this procedure to obtain at most K service path candidates.

3) GNN Architecture: Given the graph structure and link feature information as inputs, the GNN model outputs the q-value by the following procedures. To deal with link features and neighborhood links, we first transform the augmented network by treating links as nodes. Note that two nodes in the transformed augmented network are connected if the corresponding two links in the original augmented network are connected to the same node. Let $\mathbf{A}$ denote the adjacency matrix of the transformed augmented network. As a result, we can interpret the link features of the original augmented network as the node features of the transformed one.

Next, to extract the hidden representation in the graph domain, we apply the topological augmentation to $\mathbf{A}$ and obtain the diffusion matrix $\mathbf{S}$, which is given by Eq. (2), according to the extended version of GDC [19]. The extended version of GDC applies the weighted PageRank [47] into GDC to derive the transition matrix $\mathbf{T}$, where the weight of a link (e_i, e_j) in the transformed augmented network is defined as the minimum of the normalized residual capacities of links e_i and e_j in the original augmented network. We further calculate the sparsified diffusion matrix $\tilde{\mathbf{S}}$ by using the threshold-based sparsification. Then, a two-layer GCN is applied to the sparsified diffusion matrix $\tilde{\mathbf{S}}$ and link feature matrix $\mathbf{X}$ to derive the hidden representation $\mathbf{X}^{(l)}$ according to Eq. (1). Next, the graph-level features $\mathbf{X}_G^{(l)} \in \mathbb{R}^D$ are obtained by applying the sum-pooling to the feature matrix $\mathbf{X}^{(l)} \in \mathbb{R}^{N\times D}$ across nodes. Finally, the readout function modeled by DNNs computes the q-value from $\mathbf{X}_G^{(l)}$.

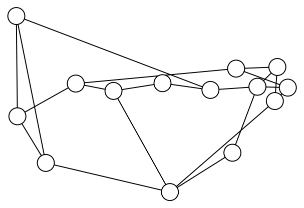
D. Applicability to Service Chaining and Function Placement Problem

Finally, we discuss the applicability of the proposed SC approach to the SCFP problem, which will be realized in the similar manner used in our previous ILP-based approaches [14], [15]. SC tries to find a service path under the predefined function locations by using the augmented network. The SC problem can be extended to the SCFP problem, which incorporates both the service chaining and function placement in the following manner. It is important to note that selecting a virtual link $(\hat{v}_f, v)$ from an imaginary node $\hat{v}_f$ to a physical node v in the mth subpath indicates the execution of VNF f at physical node v, as mentioned in Section III-B4. The virtual link $(\hat{v}_f, v)$ between imaginary and physical nodes in the augmented network signifies the deployment of VNF f on physical node v. All possible function placements can be considered by connecting each imaginary node to all physical nodes through virtual links. Specifically, we construct the augmented network for SCFP by connecting each imaginary node to all physical nodes. Finding a service path on this augmented network realizes both service chaining and function placement.

V. Numerical Results

A. Evaluation Settings

We first use two kinds of real-world network topologies: the NSFNET topology with 14 nodes and 21 links and the SPRINT topology with 11 nodes and 18 links, as shown in Figs. 4a and 4b, respectively. The topological data is available at the Internet topology zoo [49]. The original capacity of each physical link (i, j) is set to be identical, $\hat{B}_{i,j} = 1$ Gbps. As for each virtual link $(\hat{v}_f, v)$, the original capacity $\hat{P}_{\hat{v}_f,v}$ is set to be $2/|\mathcal{F}_v|$ such that the physical node v equally divides


(a) NSFNET topology (14 nodes and 21 links).

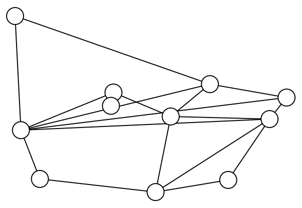

(b) SPRINT topology (11 nodes and 18 links).

Fig. 4. Network topologies used in the evaluation.

TABLE II
Service Chain Demand and Requirements (NAT: Network Address Translator, FW: Firewall, TM: Traffic Monitor, WOC: WAN Optimization Controller, IDPS: Intrusion Detection Prevention System, and VOC: Video Optimization Controller)

Service	Sequence of functions	Demand	b_c
Web service	NAT-FW-TM-WOC-IDPS	18.2%	1 Mbps
VoIP	NAT-FW-TM-FW-NAT	11.8%	4 Mbps
Video streaming	NAT-FW-TM-VOC-IDPS	69.9%	16 Mbps
Online gaming	NAT-FW-VOC-WOC-IDPS	0.1%	32 Mbps

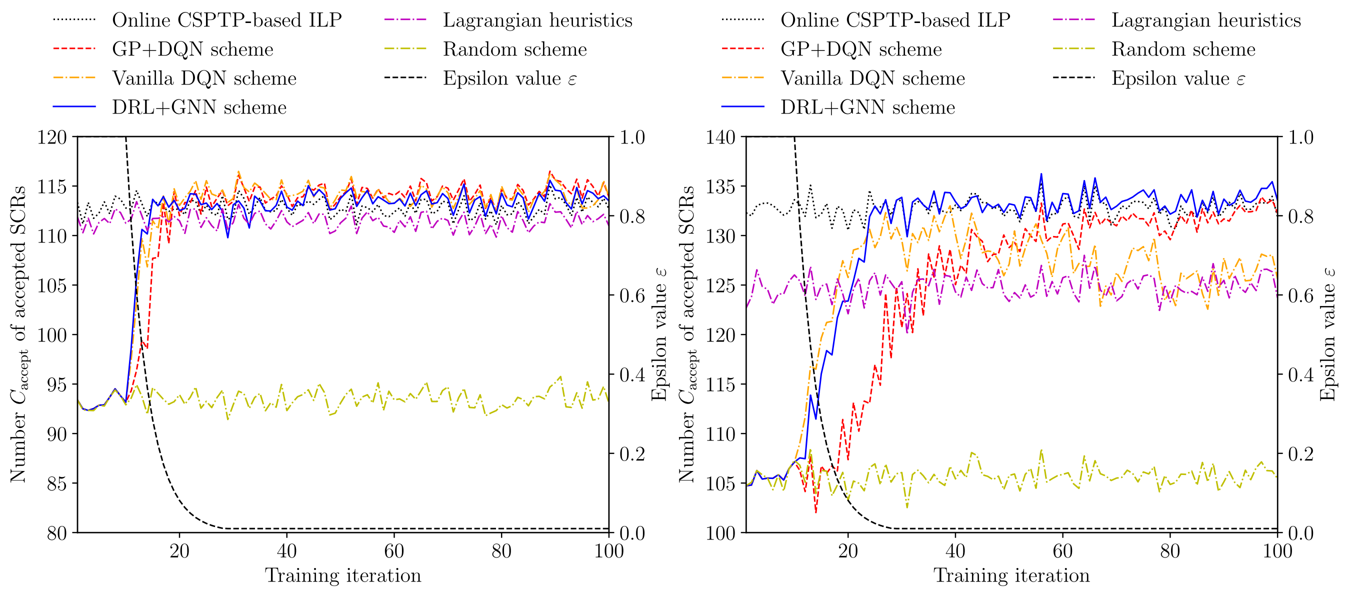
TABLE III
Relationship Between Function Type and the Number of CPU Cores for Executing the Corresponding Function Per SCR [48]

Function type	NAT	FW	TM	IDPS	VOC	WOC
$p_{c,f_{c,m}}$	0.00092	0.0009	0.0133	0.0107	0.0054	0.0054

and distributes the processing resource of two CPUs to its supporting functions $\mathcal{F}_v$. Each function $f \in \mathcal{F}$ is assigned to two VNF-enabled nodes randomly selected ($V_f = 2$). Table III gives $p_{c,f_{c,m}}$ as the number of CPU cores required for executing each function $f \in \mathcal{F}$ per SCR [48].

An event-driven simulator is implemented according to Algorithm 1. One episode of the SC scenario is as follows. A new SCR c with a random o–d pair occurs in the physical network (i.e., environment) according to the demand distribution in Table II. Next, the SDN controller (i.e., agent) allocates the resources to the SCR c according to the ε-greedy exploration strategy. To examine how many SCRs the SDN controller can simultaneously support, we assume that each established service path holds until the end of simulation. If the SDN controller fails to allocate resources to the SCR c, the simulation is terminated. The set of accepted SCRs is defined as C_{accept}. These procedures are repeated TL times where T and L are the number of iterations and that of episodes in one iteration, respectively. (T, L) = (100, 50) is used for both the training and testing phases.

The DRL+GNN agent is implemented by using Pytorch and Pytorch geometric libraries [50], [51]. In the training phase, we use the Adam optimizer [52] with the initial learning rate of 10^{-4} and the discount rate $\gamma = 0.95$. We train the model every $I = 2$ training iterations by using 5 batches with 32 samples randomly chosen from the experience replay buffer. The experience replay buffer has the size of 5000 samples with the first-in first-out (FIFO) updating policy.



(a) NSFNET topology. (b) SPRINT topology.

Fig. 5. Evolution of number C_{accept} of accepted SCRs on SPRINT topology in training phase.

As for the ε-greedy exploration strategy, in the training phase, ε initially takes one, keeps the value during the first ten iterations, then exponentially decays with the base of 0.99 every two episodes, and approaches asymptotically to 0.01. On the other hand, in the testing phase, ε is fixed to be zero in order to apply the learned GNN model and lines 12–14 in Algorithm 1 are skipped.

As for the evaluation metric, we use the average number of SCRs that are successfully allocated per episode, i.e., $C_{\text{accept}} = |\mathcal{C}_{\text{accept}}|$. Note that we have confirmed that the cumulative reward R shows the similar tendency to C_{accept} in the following results. In terms of the efficient resource allocation, we also use the total amount of incoming traffic among accepted SCRs, i.e., $B_{\text{accept}} = \sum_{c\in\mathcal{C}_{\text{accept}}} b_c$. From the viewpoint of the computational complexity, we adopt the *computation time*, which is the average time required to calculate a service path under the same scenario used in the training phase except $\varepsilon = 0$.

We compare the DRL+GNN scheme with the following five schemes: (a) a random scheme where the agent randomly selects an action regardless of the state, (b) a vanilla DQN scheme where the agent computes the q-value based on the DNN only. (c) a graph pooling (GP) +DQN scheme where the agent computes the q-value based on the DNN with graph pooling, (d) the Lagrangian heuristics for CSPTP-based SC [15], and (e) the online CSPTP-based ILP [14] where the objective function is modified to minimize the overall physical and virtual link utilization.

The vanilla DQN scheme cannot be applied to different networks from the learned network because it does not consider the node permutation invariance and equivariance [17]. To cope with this problem, the GP+DQN scheme first obtains the graph-level features $\mathbf{X}_G \in \mathbb{R}^D$ by applying the sum-pooling to the feature matrix $\mathbf{X} \in \mathbb{R}^{N\times D}$ across nodes and then computes the q-value by applying the two-layer neural networks to $\mathbf{X}_G$. As for the Lagrangian heuristics, we use it with parameters that are appropriately tuned for the initial condition. To solve the online CSPTP-based ILP, we use the existing solver CPLEX 12.8 [53] with the parallel optimization parameter (i.e., the number of threads) of 32. Note that the online CSPTP-based ILP gives the optimal solution per SCR but does not guarantee the optimality in the long-term perspective, due to the lack of prediction of future SCRs. As a result, there is a possibility that the ML-based approaches outperform the online CSPTP-based ILP in the long-term perspective by learning the demand trend.

In the calculation, we use the server with 16-core Intel Xeon Gold 6226R, 196 GB memory, and an NVIDIA GeForce RTX 3090 GPU.

B. Fundamental Characteristics

1) Training Result: We train the DRL+GNN scheme, the GP+DQN scheme, and the vanilla DQN scheme under the NSFNET and SPRINT topologies, respectively. Figs. 5(a) and 5(b) illustrate the evolution of the number C_{accept} of accepted SCRs averaged over $L = 50$ episodes per iteration during the training phase under the NSFNET and SPRINT topologies, respectively.

Since all the schemes except both CSPTP-based ILP and Lagrangian heuristics randomly adopt a service path candidate per SCR during the first 10 iterations, due to the ε-greedy exploration strategy with $\varepsilon = 1$, they show almost the same behavior, regardless of the topologies. On the other hand, they show different behavior after the 11th iteration. Since the random scheme continues the random selection, it cannot improve C_{accept}. On the contrary, the DRL+GNN, GP+DQN, and vanilla DQN schemes increase C_{accept} with iteration, which is confirmed as the learning effect with the decay of ε. In particular, the DRL+GNN scheme becomes

TABLE IV
COMPARISON OF COMPUTATION TIME UNDER THE NSFNET TOPOLOGY

Scheme	Computation time per SCR [ms]	
	Avg.	Std.
CSPTP-based ILP	105.0	9.38
DRL+GNN scheme	57.2	4.46
Lagrangian heuristics	21.9	0.29
GP+DQN scheme	50.0	3.61
Vanilla-DQN scheme	50.2	3.55
Random scheme	33.5	0.64

TABLE V
SERVICE DEMAND DISTRIBUTION OF EACH DEMAND TREND SCENARIO FOR THE TESTING PHASE

Demand trend	Web service	VoIP	Video streaming	Online gaming	Cosine similarity θ
Base	18.2%	11.8%	69.9%	0.1%	1
Different 1	24.2%	17.8%	51.9%	6.1%	0.96
Different 2	30.2%	23.8%	33.9%	12.1%	0.82
Different 3	36.2%	29.8%	15.9%	18.1%	0.54

competitive with the online CSPTP-based ILP under both topologies. Someone might wonder why the DRL+GNN scheme sometimes overcomes the online CSPTP-based ILP. This is because the online CSPTP-based ILP gives the optimal solution per SCR but does not guarantee the optimality in the long-term perspective.

Comparing the results between NSFNET and SPRINT, we confirm that the GP+DQN and vanilla DQN schemes exhibits similar performance compared with the DRL+GNN scheme under the NSFNET topologies in Fig. 5(a) while their performance is smaller than that of DRL+GNN scheme under the SPRINT topology in Fig. 5(b). This result indicates that the DRL+GNN scheme has the learning effect regardless of the network topologies, which comes from the representation capabilities of GNNs. In addition, the DRL+GNN scheme has the faster learning convergence rate than the GP+DQN scheme under both topologies.

2) Computation Time: Table IV presents the average and standard deviation of computation time for the six schemes under the NSFNET scenario. We observe that the CSPTP-based ILP and Lagrangian heuristics show the largest and smallest computation time, respectively. The DRL+GNN scheme shows the similar tendency to other ML-based schemes (i.e., vanilla-DQN and GP+DQN). More specifically, it requires larger computation time than the Lagrangian heuristics but can almost halve the computation time compared with the CSPTP-based ILP.

C. Adaptability to Different Service Demand Trend

Next, we evaluate the trained models in terms of the adaptability to demand trend through the evaluations under the following four scenarios. We first prepare the *base* (demand trend) scenario, which is the same environment in the training phase except for the random seed value. Then, we prepare the three different demand trend scenarios (i.e., *different 1*, *different 2*, and *different 3*) in descending order of its cosine similarity θ to the base service demand trend. More specifically, we make these scenarios by modifying the base scenario as follows: We reduce a certain amount of the service demand of video streaming and equally dividing it among the others. Table V shows the service demand distribution in each scenario with its cosine similarity θ to the base scenario.

Fig. 6 (resp. Fig. 7) depicts the box-and-whisker plot of C_{accept} (resp. B_{accept}) for each scheme in the base and different demand trend scenarios under the NSFNET and SPRINT topologies. The box-and-whisker plot consists of three parts, i.e., box, two whisker lines, and outliers. The box has the height ranging in $[Q_1, Q_3]$ where Q_1 (resp. Q_3) is the first (resp. third) quartile and includes a horizontal line as the median. The upper (resp. lower) whisker line is connected between Q_3 (resp. Q_1) and the upper (resp. lower) bound, over (resp. under) which the data samples are regarded as outliers, denoted by points. The length of whisker line is given by $1.5(Q_3 - Q_1)$.

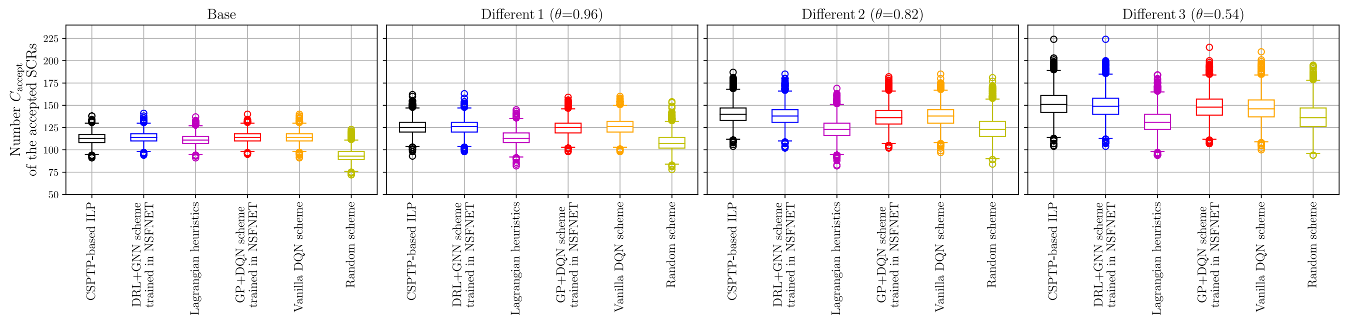
We first focus on C_{accept} of each scheme in the base demand trend scenario under both network topologies. As we expect, regardless of the network topologies, the performance of each scheme has almost the same as that achieved at the end of the training phase in Fig. 5(a) and 5(b), respectively. As a result, the DRL+GNN and GP+DQN schemes exhibit the competitive performance with the online CSPTP-based ILP under both topologies. B_{accept} in Fig. 5(a) shows the same tendency as C_{accept} in Fig. 5(b), regardless of the network topologies.

Next, we compare the results among the four scenarios. At first, someone might wonder why C_{accept} of each scheme increases with decrease of the cosine similarity θ (from the left to right in Fig. 7). This is because the different bandwidth requirement b_c among services as shown in Table II. More specifically, in the preparation of the three different scenarios, we reduce $\beta\%$ service demand of video streaming and add $\beta/3\%$ service demand to each remaining service, which reduces the bandwidth requirement in proportion to $16\beta - (1 + 4 + 32)\beta/3 \simeq 3.67\beta$. Since the amount of network resource is identical among all scenarios, such increasing trend does not arise in terms of B_{accept}, as shown in Fig. 5. This tendency can be confirmed from the evaluations in both network topologies.

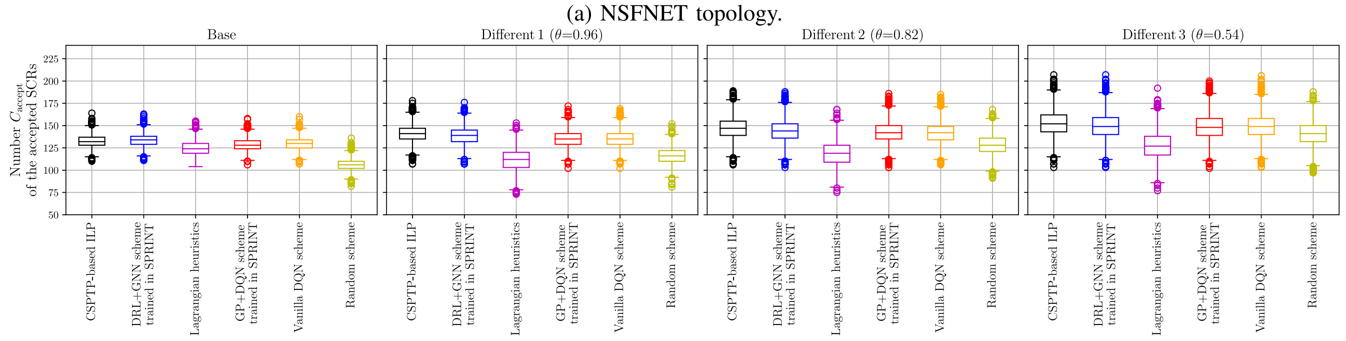
We observe from Figs. 7 and 6 that the DRL+GNN, GP+DQN, and vanilla DQN schemes have competitive C_{accepts} and B_{accepts} with the online CSPTP-based ILP among all scenarios, thanks to their generalization capabilities. Note that the GP+DQN and vanilla DQN schemes require more training iterations as shown in Fig. 5. On the other hand, the Lagrangian heuristics gradually degrades the performance with decrease of θ and consequently exhibits almost the same performance as the random scheme. This indicates that the Lagrangian heuristics fine-tuned for the base scenario cannot adapt to the demand change.

D. Adaptability to Topology Change With Link Failures

In actual systems, some of the links may be temporarily down, due to equipment failures, which changes the network topology. In this section, we evaluate how much the

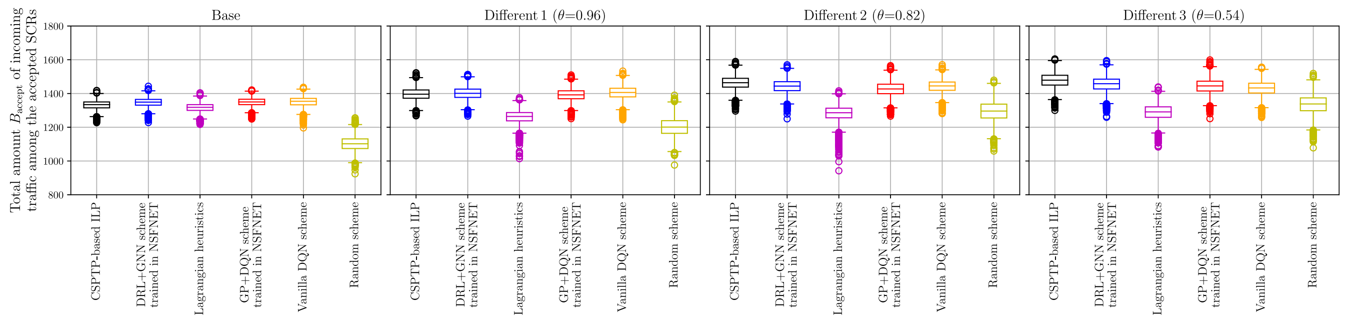


(a) NSFNET topology.

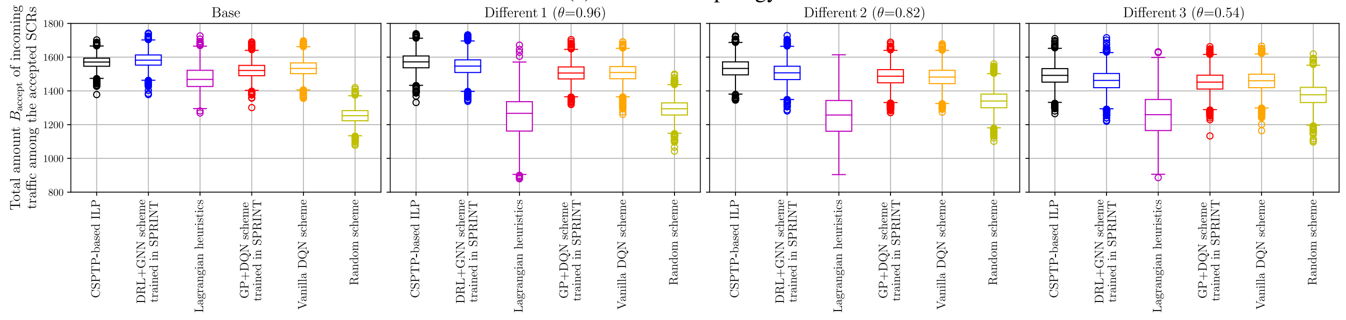


(b) SPRINT topology.

Fig. 6. The number C_{accept} of the accepted SCRs for five schemes in the testing phase.



(a) NSFNET topology.

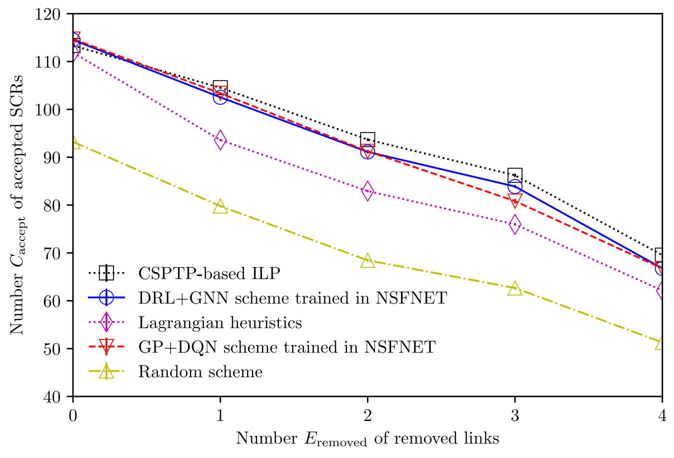


(b) SPRINT topology.

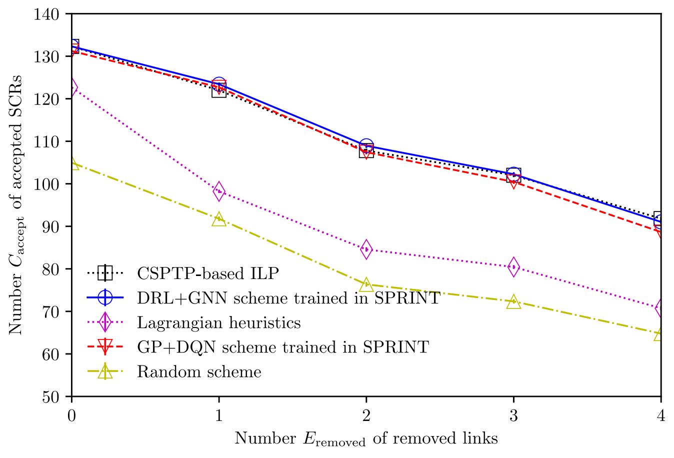
Fig. 7. Total amount B_{accept} of incoming traffic among accepted SCRs for five schemes in the testing phase.

DRL+GNN scheme learned through the original NSFNET (resp. SPRINT) topology can work well even under the NSFNET (resp. SPRINT) topology with link failure(s). In the testing phase, we prepare link failure scenarios by changing the number E_{removed} of links removed from the NSFNET and SPRINT topologies from 1 to 4, respectively. More specifically, for each E_{removed}, we randomly remove E_{removed} link(s) from the original NSFNET and SPRINT topologies at the beginning of an episode, respectively. We should note that the vanilla DQN scheme cannot be applied to this link failure scenario because the input size of the vanilla DQN scheme depends on the number of links and it changes before and after the link failure(s).

Figs. 8(a) and 8(b) depict the relationship between E_{removed} and C_{accept} for the five schemes under the NSFNET and SPRINT topologies, respectively. In these figures, we show



(a) NSFNET topology.



(b) SPRINT topology.

Fig. 8. Impact of the number E_{removed} of removed links on the number C_{accept} of accepted SCRs.

TABLE VI
CHARACTERISTICS OF 243 TOPOLOGIES IN THE INTERNET TOPOLOGY ZOO [49]

	mean	standard deviation	min	max
# of nodes	42.0	28.4	10	203
# of links	56.3	35.3	16	255

the average with 95% confidence interval. We observe that all the schemes decrease C_{accept} with E_{removed}, regardless of the network topologies. Comparing the results of the DRL+GNN scheme, GP+DQN scheme, Lagrangian heuristics, and random scheme with those of the online CSPTP-based ILP, we confirm that the maximum performance degradation becomes 3.9%, 6.3%, 11.9%, and 27.3% (resp. 0.8%, 3.5%, 23.0%, and 29.4%) under the NSFNET (resp. SPRINT) topology, respectively. The smaller performance degradation of the DRL+GNN scheme can be regarded as the generalization capabilities of the GNN and the similarity between the original topology and modified one. The Lagrangian heuristics decreases its performance due to the same reason explained in Section V-C.

E. Applicability to Other Real-World Topologies

Finally, we assess the generalization capabilities of the proposed DRL+GNN scheme using 243 real-world network topologies available from the Internet topology zoo [49]. Table VI depicts the characteristics of the 243 topologies in terms of the numbers of nodes and links. Fig. 9 illustrates the complementary cumulative distribution of the relative number of accepted SCRs compared to the online CSPTP-based ILP result. (As mentioned in Section V-A, the Vanilla DQN scheme cannot apply the learned model in a certain network to other networks, due to lack of properties of node permutation invariance and equivariance.) As the values of C_{accept} and B_{accept} vary based on network topologies, we focus on their relative performance to the online CSPTP-based ILP.

Our observations reveal the proposed DRL+GNN scheme trained in the NSFNET (resp. SPRINT) topology can achieve

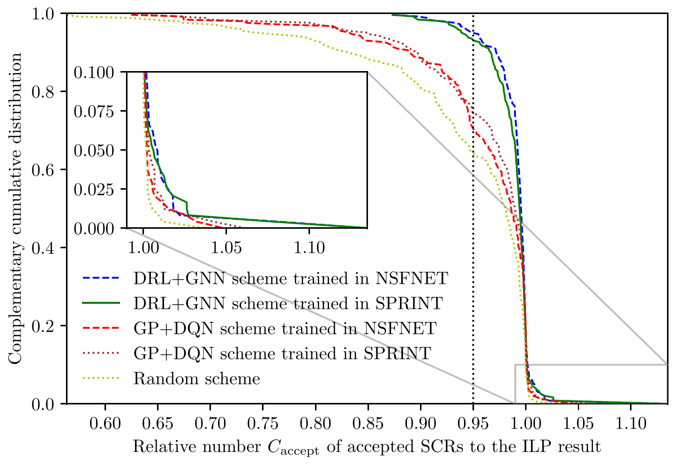


Fig. 9. Complementary cumulative distribution of the relative number of accepted SCRs to the online CSPTP-based ILP result.

over 95% relative performance to the online CSPTP-based ILP for 95.1% (resp. 93.4%) of the total real-world network topologies, with the help of DRL, GNN, and path candidates. In comparison, the GP+DQN scheme trained in the NSFNET (resp. SPRINT) topology and the random scheme support 70.8% (resp. 74.5%) and 64.6% of the total real-world topologies in the same case. This outcome demonstrates the DRL+GNN scheme trained in the NSFNET/SPRINT topology can support most real-world topologies while mitigating performance degradation. As mentioned in Section V-A, the online CSPTP provides an optimal solution per SCR but does not guarantee optimality in the long-term perspective. As a result, the DRL+GNN (resp. GP+DQN) scheme trained in NSFNET and SPRINT topologies can achieve over 100% relative performance compared to the online CSPTP-based ILP across 21.4% and 18.9% (resp. 19.7% and 19.3%) of the topologies.

Additionally, we observe a similar tendency in the total amount B_{accept} of incoming traffic among the accepted SCRs, mirroring C_{accept}. The enhanced performance of the

DRL+GNN scheme can come from the benefit of the generalization capabilities by graph diffusion. By employing graph diffusion, the proposed agent identifies critical physical links for SC in terms of resource efficiency and aggregates their features into the graph feature during candidate path evaluation for the q-value determination. Conversely, the GP+DQN scheme utilizes graph-pooling to aggregate the features of all physical links in the network topology into the graph feature, making it challenging to identify the crucial features of the bottleneck links for SC.

VI. Conclusion

In this paper, we have proposed the deep reinforcement learning (DRL) framework with the graph neural network (GNN) for addressing the service chaining (SC) problem based on the capacitated shortest path tour problem (CSPTP) in the context of network functions virtualization (NFV) and software defined networking (SDN). The proposed framework adopts the GNN architecture for computing the q-values, which consists of the graph convolutional network and graph diffusion convolution. Through the numerical results, we have shown that the proposed framework achieves both optimality and adaptability (generalization capabilities). More specifically, as for the optimality, the proposed framework is competitive with the online CSPTP-based ILP. As for the adaptability, the proposed framework trained under a base demand distribution (resp. a certain network topology) can also work well under different demand distributions (resp. changes of network topologies, due to link failures or different networks). Specifically, the proposed framework demonstrates competitive performance compared to the CSPTP-based ILP in the majority of real-world topologies, thanks to the graph diffusion.

References

[1] T. Hara and M. Sasabe, "Deep reinforcement learning with graph neural networks for capacitated shortest path tour based service chaining," in *Proc. Int. Conf. Netw. Service Manag. (CNSM)*, Oct. 2022, pp. 1–9.

[2] S. Yang, F. Li, S. Trajanovski, R. Yahyapour, and X. Fu, "Recent advances of resource allocation in network function virtualization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 295–314, Feb. 2021.

[3] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on service function chaining," *J. Netw. Comput. Appl.*, vol. 75, pp. 138–155, Nov. 2016.

[4] B. Yi, X. Wang, K. Li, S. K. Das, and M. Huang, "A comprehensive survey of network function virtualization," *Comput. Netw.*, vol. 133, pp. 212–262, Mar. 2018.

[5] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[6] J. Zhang, Z. Wang, N. Ma, T. Huang, and Y. Liu, "Enabling efficient service function chaining by integrating NFV and SDN: Architecture, challenges and opportunities," *IEEE Netw.*, vol. 32, no. 6, pp. 152–159, Nov./Dec. 2018.

[7] J. Pei, P. Hong, K. Xue, D. Li, D. S. L. Wei, and F. Wu, "Two-phase virtual network function selection and chaining algorithm based on deep learning in SDN/NFV-enabled networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1102–1117, Jun. 2020.

[8] D. Heo, S. Lange, H.-G. Kim, and H. Choi, "Graph neural network based service function chaining for automatic network control," in *Proc. Asia–Pacific Netw. Oper. Manag. Symp. (APNOMS)*, Sep. 2020, pp. 7–12.

[9] D. Heo, D. Lee, H.-G. Kim, S. Park, and H. Choi, "Reinforcement learning of graph neural networks for service function chaining," Nov. 2020, *arXiv:2011.08406*.

[10] A. Rafiq, T. A. Khan, M. Afaq, and W.-C. Song, "Service function chaining and traffic steering in SDN using graph neural network," in *Proc. Int. Conf. Inf. Commun. Technol. Converg.*, Oct. 2020, pp. 500–505.

[11] Z. Ning, N. Wang, and R. Tafazolli, "Deep reinforcement learning for NFV-based service function chaining in multi-service networks: Invited paper," in *Proc. IEEE Int. Conf. High Perform. Switch. Routing (HPSR)*, May 2020, pp. 1–6.

[12] B. Awerbuch, Y. Azar, and A. Epstein, "The price of routing unsplittable flow," in *Proc. ACM Symp. Theory Comput.*, May 2005, pp. 57–66.

[13] S. Bhat and G. N. Rouskas, "Service-concatenation routing with applications to network functions virtualization," in *Proc. Int. Conf. Comput. Commun. Netw.*, Jul. 2017, pp. 1–9.

[14] M. Sasabe and T. Hara, "Capacitated shortest path tour problem based integer linear programming for service chaining and function placement in NFV networks," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 104–117, Mar. 2021.

[15] T. Hara and M. Sasabe, "Speedy and efficient service chaining and function placement based on Lagrangian heuristics for capacitated shortest path tour problem," *J. Netw. Syst. Manag.*, vol. 31, no. 1, pp. 1–34, Dec. 2022.

[16] R. Boutaba et al., "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *J. Internet Services Appl.*, vol. 9, no. 16, pp. 1–99, Jun. 2018.

[17] W. L. Hamilton, *Graph Representation Learning* (Synthesis Lectures on Artificial Intelligence and Machine Learning), vol. 14. San Rafael, CA, USA: Morgan Claypool, Sep. 2020.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," Feb. 2017, *arXiv:1609.02907*.

[19] J. Klicpera, S. Weiß enberger, and S. Günnemann, "Diffusion improves graph learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, Dec. 2019, pp. 13333–13345.

[20] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," Mar. 2020, *arXiv:1812.04202*.

[21] J. Zhou et al., "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Jan. 2020.

[22] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," Jun. 2017, *arXiv:1704.01212*.

[23] X. Chen et al., "Reinforcement learning–based QoS/QoE-aware service function chaining in software-driven 5G slices," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 11, pp. 1–18, Jul. 2018.

[24] D. L. Mills and H. Braun, "The NSFNET backbone network," in *Proc. ACM Workshop Front. Comput. Commun. Technol.*, 1987, pp. 191–196.

[25] C. Fraleigh et al., "Packet-level traffic measurements from the sprint IP backbone," *IEEE Netw.*, vol. 17, no. 6, pp. 6–16, Nov./Dec. 2003.

[26] N. Huin, B. Jaumard, and F. Giroire, "Optimal network service chain provisioning," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1320–1333, Jun. 2018.

[27] N. Hyodo, T. Sato, R. Shinkuma, and E. Oki, "Virtual network function placement for service chaining by relaxing visit order and non-loop constraints," *IEEE Access*, vol. 7, pp. 165399–165410, 2019.

[28] L. Gao and G. N. Rouskas, "Congestion minimization for service chain routing problems with path length considerations," *IEEE/ACM Trans. Netw.*, vol. 28, no. 6, pp. 2643–2656, Dec. 2020.

[29] D. Li, P. Hong, K. Xue, and J. Pei, "Virtual network function placement and resource optimization in NFV and edge computing enabled networks," *Comput. Netw.*, vol. 152, pp. 12–24, Apr. 2019.

[30] T. Nguyen, A. Girard, C. Rosenberg, and S. Fdida, "Routing via functions in virtual networks: The curse of choices," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1192–1205, Jun. 2019.

[31] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco, and F. Liberal, "Virtual network function placement optimization with deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 292–303, Feb. 2020.

[32] S. M. A. Araújo, F. S. H. de Souza, and G. R. Mateus, "A hybrid optimization-machine learning approach for the VNF placement and chaining problem," *Comput. Netw.*, vol. 199, Nov. 2021, Art. no. 108474.

[33] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou, "Optimal VNF placement via deep reinforcement learning in SDN/NFV-enabled networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 263–278, Feb. 2020.

[34] G. Sallam, G. R. Gupta, B. Li, and B. Ji, "Shortest path and maximum flow problems under service function chaining constraints," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 2132–2140.

[35] P. Festa, "The shortest path tour problem: Problem definition, modeling, and optimization," in *Proc. INOC*, Apr. 2009, pp. 1–7.

[36] P. Sun, J. Lan, J. Li, Z. Guo, and Y. Hu, "Combining deep reinforcement learning with graph neural networks for optimal VNF placement," *IEEE Commun. Lett.*, vol. 25, no. 1, pp. 176–180, Jan. 2021.

[37] P. Almasan, J. Suárez-Varela, A. Badia-Sampera, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, "Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case," Feb. 2020, *arXiv:1910.07421*.

[38] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, Nov. 2018.

[39] R. Bellman, "A Markovian decision process," *J. Math. Mech.*, vol. 6, no. 5, pp. 679–684, Apr. 1957.

[40] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, Jul. 1966.

[41] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[42] M. Zeiler et al., "On rectified linear units for speech processing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 3517–3521.

[43] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the Web," Stanford InfoLab, Stanford, CA, USA, Rep. 1999-66, Nov. 1999.

[44] F. Chung, "The heat kernel as the pagerank of a graph," *Proc. Nat. Acad. Sci.*, vol. 104, no. 50, pp. 19735–19740, Dec. 2007.

[45] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, Feb. 2016, pp. 2094–2100.

[46] U. Brandes, "On variants of shortest-path betweenness centrality and their generic computation," *Social Netw.*, vol. 30, no. 2, pp. 136–145, May 2008.

[47] W. Xing and A. Ghorbani, "Weighted PageRank algorithm," in *Proc. 2nd Annu. Conf. Commun. Netw. Services Res.*, May 2004, pp. 305–314.

[48] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing-resource sharing on the placement of chained virtual network functions," *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1479–1492, Oct.–Dec. 2021.

[49] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

[50] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2019, pp. 8024–8035.

[51] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch geometric," in *Proc. ICLR Workshop Represent. Learn. Graphs Manifolds*, May 2019, pp. 1–9.

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Jan. 2017, *arXiv:1412.6980*.

[53] "IBM ILOG CPLEX optimizer." Accessed: Jul. 24, 2023. [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio

Takanori Hara (Member, IEEE) received the B.Eng. degree from the National Institution for Academic Degrees and Quality Enhancement of Higher Education in 2016, and the M.Eng. and Ph.D. degrees from the Nara Institute of Science and Technology, Japan, in 2018 and 2021, respectively, where he is currently an Assistant Professor with the Division of Information Science, Graduate School of Science and Technology. His research interests include NFV/SDN, machine learning for networking, eBPF/XDP, and game-theoretic approaches.

Masahiro Sasabe (Member, IEEE) received the B.S., M.E., and Ph.D. degrees from Osaka University, Japan, in 2001, 2003, and 2006, respectively. He is currently a Professor with the Faculty of Informatics, Kansai University, Japan. His research interests include P2P/NFV networking, game-theoretic approaches, human-harmonized network systems, and network optimization. He is a member of IEICE.