Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.062716

ARTICLE

Priority-Aware Resource Allocation for VNF Deployment in Service Function Chains Based on Graph Reinforcement Learning

Seyha Ros[1,#], Seungwoo Kang[1,#], Taikuong Iv[1], Inseok Song[1], Prohim Tam[2] and Seokhoon Kim[1,3,*]

[1]Department of Software Convergence, Soonchunhyang University, Asan, 31538, Republic of Korea
[2]School of Digital Technologies, American University of Phnom Penh, Phnom Penh, 12106, Cambodia
[3]Department of Computer Software Engineering, Soonchunhyang University, Asan, 31538, Republic of Korea

*Corresponding Author: Seokhoon Kim. Email: seokhoon@sch.ac.kr

[#]These authors contributed equally to this work

Received: 25 December 2024; Accepted: 06 March 2025; Published: 16 April 2025

ABSTRACT: Recently, Network Functions Virtualization (NFV) has become a critical resource for optimizing capability utilization in the 5G/B5G era. NFV decomposes the network resource paradigm, demonstrating the efficient utilization of Network Functions (NFs) to enable configurable service priorities and resource demands. Telecommunications Service Providers (TSPs) face challenges in network utilization, as the vast amounts of data generated by the Internet of Things (IoT) overwhelm existing infrastructures. IoT applications, which generate massive volumes of diverse data and require real-time communication, contribute to bottlenecks and congestion. In this context, Multi-access Edge Computing (MEC) is employed to support resource and priority-aware IoT applications by implementing Virtual Network Function (VNF) sequences within Service Function Chaining (SFC). This paper proposes the use of Deep Reinforcement Learning (DRL) combined with Graph Neural Networks (GNN) to enhance network processing, performance, and resource pooling capabilities. GNN facilitates feature extraction through Message-Passing Neural Network (MPNN) mechanisms. Together with DRL, Deep Q-Networks (DQN) are utilized to dynamically allocate resources based on IoT network priorities and demands. Our focus is on minimizing delay times for VNF instance execution, ensuring effective resource placement, and allocation in SFC deployments, offering flexibility to adapt to real-time changes in priority and workload. Simulation results demonstrate that our proposed scheme outperforms reference models in terms of reward, delay, delivery, service drop ratios, and average completion ratios, proving its potential for IoT applications.

KEYWORDS: Deep reinforcement learning; graph neural network; multi-access edge computing; network functions virtualization; software-defined networking

1 Introduction

The Internet of Things (IoT) has increased rapidly in recent years, as it has birthed an immense chance to transition from a conventional economy to a digital economy. As in an intrinsically distributed data-centric system, IoT can gather and provide an enormous volume of data that can be processed, interpreted, and analyzed into useful information and acknowledgment. Hence, in the context of IoT-based applications, it involves creating huge and vast amounts of data that lead to resource constraints, increasingly vast resource capacity in network performance, and resource congestion in terms of the urgent business on time. However, Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) were proposed to provision treated as the key enabler technologies to address dynamic and agile IoT networks

[1–4]. SDN creates a programmable network by centralizing the control plan and data plan functionality from physical routing devices, while NFV decouples various network resources to compose the Network Functions (NFs) from physical network devices, running them as virtualization technologies [5,6]. By adopting these technologies, SDN and NFV-enabled IoT networks considerably simplify the deployment of NFs and improve network performance and service.

Upon closely providing the network resource management and orchestration of all data utilization, Service Function Chaining (SFC) is used to indicate an ordered or sequential collection of Virtual Network Functions (VNFs) to create service functions [6]. SFC propagates ordered NFs from ingress to egress (e.g., VNF1, VNF2, VNF3, and VNF-*n*); meanwhile, SFC requests must be rounded by specifying the typed and listed VNFs in the specified order. On the other hand, the more appropriate representation is the VNF-Forwarding Graph (VNF-FG), which is proposed by the European Telecommunications Standards Institute (ETSI) [7,8]. In fact, Multi-access Edge Computing (MEC) architecture provides benefits in conducting the proliferation of IoT networks. MEC provides efficient resource utilization and management mechanisms by eliminating computation intensively to enhance the end user of the resource capacities in terms of computation and communication on latency. However, placing and allocating virtual resources in network resource management in MEC servers is essential for responding and adapting to the satisfaction of service derived in complex network environments. Otherwise, VNFI employs resource computing techniques to optimize resource placement and allocation, offering scalable, adaptable, and flexible real-time solutions. However, efficient resource management is impacted by heterogeneous workloads to meet user demands for dynamic and flexible service. Due to limitation of availability of those resource orchestration and computation for diversity applications under MEC architecture, placement, and allocation of resources adaptively lead to many challenges. These challenges arise, including resource constraints, latency, and interoperability issues, during compute-intensive servers across heterogeneous IoT networks on applications. The distributed significant environment of multi-service demands, and multi-vendor applications adds complexity to allocating and placing the resource utilization.

This paper focuses on leveraging intelligent resource allocation and placement based on service priority demands-driven resource management in IoT networks. By integrating Deep Reinforcement Learning (DRL) based on Graph Neural Network (GNN) paradigms which use the graph structure of Message-Passing Neural Network (MPNN) for embedding the nodes and link states of VNF [9–11], GNN solves the scalability problem caused by MPNN variants in VNF-FG size. Moreover, DRL is implemented to determine the policy charging on SDNC for adaptively aligning the optimal decision service provisioning in the network fluctuation on IoT network. Fig. 1 leverages DRL based on GNN to increase near-optimal decision-making in the dynamic service provisioning process. SDN controller captures the traffic flow in distributed IoT networks in routing tables by using OpenFlow protocol for managing the flow entries. Meanwhile, NFVO handles orchestrating and managing the resource instant for computing workload of IoT networks. On the other hand, DRL-GNN framework ensures efficient techniques to align the controller for adaptive self-configuration on VNF placement and resource allocation. Our main contributions are summarized as follows:

- We propose DRL based on the GNN framework to support service requests simultaneously from MEC resources, maximizing the SFC acceptance ratio while minimizing the MEC workload. By addressing the complexities of resource allocation and VNF placement in SFC modules, our methods ensure priority-driven resource management in IoT networks, optimizing resource utilization based on service priorities. Once, NFVO abstracts the SFC placement in all the resource allocations to address and guarantee various network services' QoS requirements while offering minimum delay and resource usage.

- Leveraging a DRL-GNN algorithm, namely a priority-aware VNF deployment algorithm (PVD-GDR), to set the threshold outperformance and satisfy IoT applications. The PVD-GDR framework enables the capability to ensure resource preservation on VNF nodes and link embedding, which utilizes a message-passing variant to propagate the feature information to its neighbors until a stable equilibrium is reached optimally. We employ DQN algorithms to provide the capabilities to learn optimal policies in dynamic settings for efficient priority-aware resource orchestration and adaptive resource utilization in SFC deployment in assisted control data flow. Hence, DQN ensures that the update of value function in SFC deployment and make flexibility decision for SFC requests arriving at real-time in difference setting according to the optimal value function.
- Lastly, our scheme shows the performance matric, which compares the reference scheme in terms of reward, end-to-end task execution delay of requested service, delivery ratios of requested service, task drop ratios of requested service, and average completion ratios.

The remainder of the paper is aligned as follows: in Section 2, we illustrate the related work and the current work. Section 3 describes the network system formulation in terms of computation and network resource utilization. Section 4 presents DRL-GNN for service priority on IoT demands. Lastly, the proposed scheme indicates its performance compared to the reference scheme.

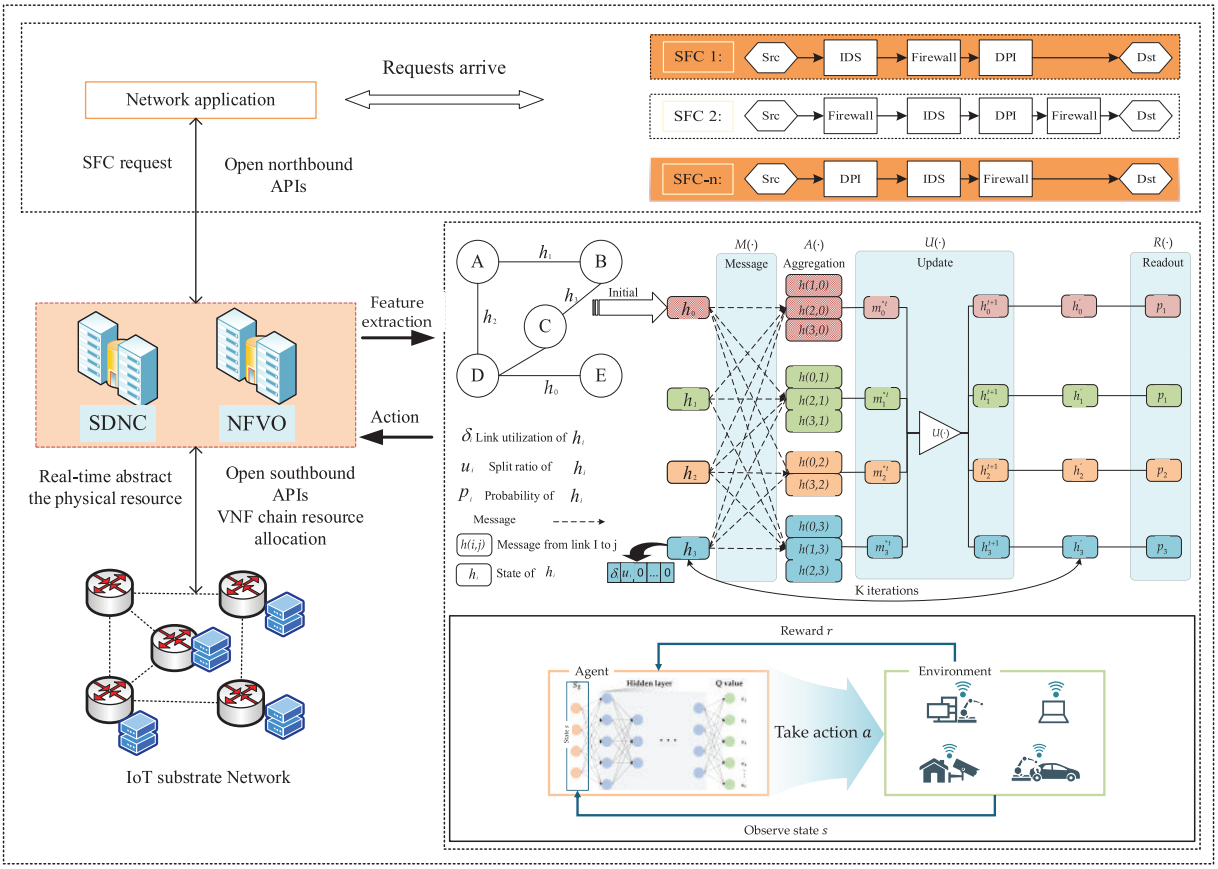

Figure 1: Message-passing neural network embedding with SDNC/NFVO architecture

2 Preliminary of Priority/Demanding in Resource Utilization

In this section, many studies have investigated resource placement and solved optimization problems. However, efficient allocation and resource placement of the limited resource is the most impact-critical component in MEC and its efficiency for orchestrating resource to satisfaction on different QoS. Hence, minimum delay and resource cost are priorities for the end goal. Additionally, we comprehensively gather previous studies on resource placement in MEC.

2.1 Enabling Resource Utilization Based on DRL

The utilization of resources plays a crucial role in demonstrating the efficiency of resource allocation and computation. SFC deployment directly impacts specific network functions to achieve network service targets. Moreover, compiling SFC formations for multiple sequential applications is a critical area that requires timely processing [12–15]. DRL has been leveraged to address the challenges of SFC placement, particularly in terms of service flexibility and diversity. However, existing DRL approaches struggle with handling a dynamic number of objectives and suffer from unreliable training time consumption. To overcome this, researchers have utilized divided sub-problem weight vectors to establish the relationship between solution positioning and placement. By introducing a related zone decomposition model, the weighted vectors are trained in parallel, leading to time savings and improved model quality. Furthermore, the dynamic objective placement method ensures efficient training for high-dimensional models. In [16] with multi-objective optimization problem was transformed into a single-objective Mixed Integer Linear Programming (MILP) problem, aiming to maximize the acceptable flow rate while minimizing energy costs for multiple service chains. A polynomial-time algorithm, based on linear relaxation and rounding, to approximate the optimal solution of MILP. This approach achieved a weakly Pareto-optimal solution resulting in balancing acceptable flow rate and energy cost, while ensuring easy implementation. Similarly, studies [17,18] utilized the DRL capability to address the challenges in SFC placement in terms of service flexibility and diversity. With the existing DRL lacking the ability in solving a dynamic number of objectives and unreliable training time consumption, the author takes the advantage of divided sub-problem's weight vectors to show the relation toward the solution position of placement. By proposing related zone decomposition (MQDRL), the weighted vectors are trained in parallel, resulting in time-saving and improving the quality of models. Then, the dynamic objective placement method ensures the training with the high-dimension model. As a result, the proposal achieved an improvement of the service acceptance ratio and the hyper-volume value in multi-objective SFC placement. Reference [19] categorized two sub-classes of the task according to their real-time (type-0) and non-real-time (type-1) requirements according to the priority level and latency demand. The author proposed two-phase solutions. First, the PARS scheme prioritizes type-0 users and allows network operators to assist the resource over the performance of SFC resource allocation. In the next phase, an adaptive algorithm, with two parts, was created; the first part identifies the best candidate node for deploying VNF, then the second part evaluates the best path connecting identified nodes using Dijkstra's minimum cost path-finding algorithm.

In [20], the authors addressed the resource competition of utilizing collaborative edges by proposing a prioritized queuing scheme with task dependency (PQTD) to allow the task with high priority to jump the queue, which addressed the resource competition in the edge with the drawback of implementing algorithms first come first served (FCFS) and other benchmark schemes. In addition, a joint DAG-queue delay model was proposed to address the complication in delay change due to queue-jumping by analyzing the chain reaction of delay changes. As a multi-task assignment problem, PQTD with Monte Carlo Tree Search algorithm is proposed to maximize the average satisfaction degree framework and avoid local optimization dilemmas. To address the resource constraints in IoT devices, the study in [21] introduces IRATS, a PPO-based intelligent algorithm for priority- and deadline-aware resource allocation and task scheduling, aimed

at enhancing computational workloads and communication capabilities. By modeling resource allocation as an MDP, IRATS minimizes task waiting time and delay, using a multi-level queue scheduler to prioritize tasks.

2.2 DRL-GNN Method-Based Resource Utilization in IoT

Although deep learning (DL) can effectively capture patterns in Euclidean data, the complexity of network topology and data generation in IoT networks presents significant challenges, as conventional DL algorithms struggle to handle non-Euclidean data effectively. For that reason, researchers found that GNN is one of the up-to-date methods to extract network properties in network environments and utilize in SFC deployments according to superiority of GNN in graph features extraction.

For instance, Zhang et al. [22] employed GNNs alongside a multitask regression layer to analyze VNF-FG topologies and predict future resource requirements. Similarly, reference [23] explored resource allocation and task scheduling for edge machine learning, integrating heuristic algorithms, GNNs, and Multi-Agent Reinforcement Learning (MARL). While heuristic approaches lack scalability in dynamic environments, GNN-based models effectively capture task-device dependencies. Furthermore, MARL enhances decentralized decision-making, allowing multiple agents to collaboratively optimize resource usage. Building on these advancements, our work combines GNNs and MARL to overcome limitations in scalability and adaptability, achieving enhanced task placement and resource efficiency. Additionally, reference [24] proposed DRL-based on GNN approach to optimize resource consumption in elastic optical data center. This reinforcement learning method integrated Graph Convolutional Networks (GCNs) with temporal-difference learning. This method leverages GCNs to capture and provide detailed insights into the network state, aiming to maximize the long-term average revenue while enabling real-time deployment decisions for SFC requests.

In summary, existing research has primarily focused on resource utilization and placement. While previous studies have made significant contributions to their respective domains [25–29], challenges remain in addressing the diverse critical service priorities of real-time computation in IoT networks. Specifically, there is a need to achieve convergence between autonomous resource allocation, placement, and the mapping of service criticality.

In this paper, we investigate how to handle priority-demanding resource management using an effective solution which utilizes the GNN model for effective extraction of network in IoT network. MPNN executes through three steps: hidden layer, aggregation, and readout in network topologies. Subsequently, our model integrates with DQN-enabled NFV architecture to ensure efficient priority-aware resource orchestration and adaptive resource utilization in SFC deployment. This work has two objectives minimizing short delay time on executing VNF instances and ensuring resource utilization properly for resource placement in SFC deployment.

3 System Model and Problem Formulation

At this stage, NFV architecture integrated with DRL-GNN are given to address DQN with agent policy formulation for determining service pooling and orchestrating resource. Therefore, forwarding decisions and rules obey the optimal SFC orchestration policy. In VNF-FG, descriptor is cooperated with leveraging the Topology and Orchestration Specifications for Cloud Applications (TOSCA) template in NFV [30]. We introduce the system model, including the substrate physical resource in the MEC server of the network model, SFC request model, and resource allocation. Then, we aim to offer a formulation method to represent our proposed scheme, including constraints, and minimize the execution cost and delay in system architecture. The key notation of our system model is in Table 1.

Table 1: Notation of system models

Notations	Descriptions
$G = V, E$	Physical substrate topology
V	Set of the number of MEC nodes $V = \{1, 2, \ldots, v\}, \forall v \in V$
E	Set of the number of links $E = \{1, 2, \ldots, e\}, \forall e \in E$
BW	Bandwidth capacity of links between nodes $(e \in E)$
PD	Propagation delay of $(e \in E)$
C_i^v	The capacity of type resource of node $v_i \in V$
C_v^{max}	Upper-bound capacities of physical node (CPU, memory, disk)
C_j^e	The unit cost of bandwidth in links (i, j)
T	Number of timeslots $t \in T$
	Service request
F	Set of the number of service request $f \in F$
$\overline{V}$	Set of nodes of VNFs $\{1, 2, \ldots, \overline{v}\}, \forall \overline{v} \in \overline{V}$
$\overline{E}$	Set of virtual links $\{1, 2, \ldots, \overline{e}\}, \forall \overline{e} \in \overline{E}$
$\overline{v}_i \in \overline{V}$	Set of the number $\overline{V}$
$\overline{e}_j \in \overline{E}$	Set of the number link of $\overline{E}$
$Q_{\overline{v}_i}^r$	Required of $r - type$ resource of VNF $\overline{v}_f$
$C_{\overline{v}}^{f,t}$	The request of the resource capacity at $time - t$ (CPU, memory, disk)
$B_{\overline{e}_{f,j}}$	Required of bandwidth of the link $\overline{e}_f$
	Decision variable
$x_{\overline{v}_{f,i}}^v$ (t)	Equals 1 if VNF $\overline{v}_f$ of service request f is deployed at node v at *time – t*
$y_{\overline{e}_{f,j}}^e$ (t)	Equals 1 if link *i, j* is used to build the path of virtual link $\overline{e}_f$ of service request at $time - t$

3.1 System Architecture

We use the undirected graph to represent the physical resource. $G = (V, E)$, represents a network with NFV, where V is the set of MEC server nodes and $E$is the set of the physical edge, with v denotes the v – th server and e denotes the number of the u – th physical edge number. Each MEC server can instantiate the virtual machines (VMs) to support the multiple type VNFs. Each server has a maximum computing resource capacity, i.e., $C_v^{max} = [cpu_v^{max}, m_v^{max}, d_v^{max}]$ represented the amount of CPU, memory, and disk resources, respectively. Therefore, the physical edge, $e_j \in E$, connects two servers. The physical edge e_j represented by the cluster of quaternions (source, destination, bandwidth, and propagation delay), where v_{src} and $v_{des} \in V$ are denoted the node of source and destination of e_j, respectively. Propagation delay is the term of delay between physical edge denoted by P_e.

3.2 SFC Request Model

We assume the set of service requests is denoted by $SF = \{f, C_i^{\overline{v}}\}$ representing SFC request in the network, where f denotes the chain of the network service functions. $C_i^{\overline{v}}$ denotes the type of network resource capacity or node attributed (e.g., CPU, memory, and disk).

Decision Variables

In this case, optimized time is divided into multiple sequential discrete time slots. It provides optimal VNF placement and virtual link mapping, minimizing cost, resource usage, and delay.

- Binary variable $x^{v}_{\overline{v}_{f,i}}(t)$ equals to 1 if the VNF $\overline{v}_i$ from service f is deployed on a physical node v at $time-t$, and 0 otherwise. By determining the specific placement of VNFs, $x^{v}_{\overline{v}_{f,i}}(t)$ facilitates efficient utilization of the computational resources, including CPU, memory, and storage, available under the physical resource nodes.
- Binary variable $y^{e}_{\overline{e}_{f,j}}(t)$, equals to 1 if the physical link is leveraged to establish the path for the virtual link $\overline{e}_j$ at *time-t* and 0 otherwise.

3.3 Resource Allocation

3.3.1 Node Resource

The node ensures the maximum capacity of a physical MEC server is not eclipsed by the resources in VNFs. Every server possesses limited computational resources, such as CPU, memory, and disk components. The resource demands of the VNFs are added up for all service requests that are assigned to servers. This constraint prevents overloaded physical nodes from receiving too many requests while protecting the quality of services and the system's overall stability. Resource demand of a specific VNF $\overline{v}_i$ denotes as $Q^{r}_{\overline{v}_i}$, while considered in terms of physical node's capacity denoted by C^{max}_{v}.

$$\sum_{f\in F}\sum_{\overline{v}_{f,i}} Q^{r}_{\overline{v}_{f,i}} x^{v}_{\overline{v}_{f,i}}(t) \le C^{max}_{v}, r \in \{CPU, memory, disk\}, \forall v \in V, t \in T \tag{1}$$

3.3.2 Link Bandwidth

In a particular defined bandwidth constraint, a set of links denoted as $\overline{E} = \{1, 2, \ldots, \overline{e}\}$. The cost required of bandwidth of link denote by $\overline{e}_f$ as in Eq. (2) to ensure that the sum of bandwidth allocated to virtual links mapped onto a single physical link does not exceed its available bandwidth.

$$\sum_{f\in F}\sum_{\overline{e}_{f,j}} B_{\overline{e}_{f,j}} y^{e}_{\overline{e}_{f,j}}(t) \le BW, \forall e \in E, t \in T \tag{2}$$

3.4 Delay Constraint

3.4.1 Communication Delay

Communication captures the delay caused by data transmission over physical links. Hence, communication delay for virtual links relies on the propagation delay from ingress to egress by chaining the VNF nodes. Otherwise, we need to make sure in ensuring that the service request meets the latency requirement.

$$PD = \sum_{e\in E} P_e y^{e}_{\overline{e}_{f,j}}(t) \tag{3}$$

3.4.2 Computation Delay

Now, we need to make sure that computation delay for processing a VNF on a MEC server ensures that VNFs are executed within reasonable time limits. Eq. (4) expresses the computation allocation in MEC for executing $VNF-i$. The CPU demand of each VNFs denote as $Q^{cpu}_{\overline{v}_i}$ and CPU capacity of the MEC servers denotes as Q^{cpu}_{v}.

$$D_{comp} = \frac{Q^{CPU}_{\overline{v}_i}}{Q^{CPU}_{v}} \tag{4}$$

Minimizing the total execution delay is a key objective, which achieved by optimally allocating sufficient computational resources, ensuring adequate bandwidth for the NFV-enabled $node-i$, and accurately predicting the performance of VNFs along the ordered path. Eq. (5) expresses the execution delay, which includes modification delay to reconfigure the chain descriptor by following the agent policy, computation, and communication delays, representing the total time required to process a service request. This constraint ensures that the delay is minimized for optimal service performance in any required interval:

$$D_f(t) = \sum_{\overline{v}_i} \left(D_{comp}\left(\overline{v}_i, v\right) + PD \right). \tag{5}$$

The objective is to minimize the total cost and delay in embedding service requests. The total cost includes computing resources, bandwidth usage, and propagation delay. The delay encompasses computation and communication delays for VNFs.

$$min \sum_{t \in T} \left[\sum_{f \in F} \left(\sum_{\overline{v}_i} \sum_{v \in V} C_i^v x_{\overline{v}_{f,i}}^v (t) + \sum_{\overline{e}_j \in \overline{E}} \sum_{e \in E} C_j^e y_{\overline{e}_{f,j}}^e \right) + \sum_{f \in F} D_f(t) \right]. \tag{6}$$

s.t. (1)–(5).

4 Graph Reinforcement Learning-Based Solution in SFC

In this section, we present the design of DRL-GNN algorithms to address the proper handling of resource allocation and VNF placement for provisioning resource priority on demands. Firstly, we introduce the GNN framework that interacts with network environments. GNN embedded the service functionality in VNF nodes as a directed graph by leveraging Message Passing Neural Network (MPNN) variant. Meanwhile, the DRL framework is feasibly defined as the agent policy to select resource utilization regarding the VNF-FG placement problem. It significantly decomposes the creation of a realistic networking environment to make a policy charging for network configuration. GNN gathers network states and readout as node embedding, link level, and MEC graph. After gaining output from the GNN operation, DRL will define the action, and DQN will decompose the target q-network and online network as in Fig. 2.

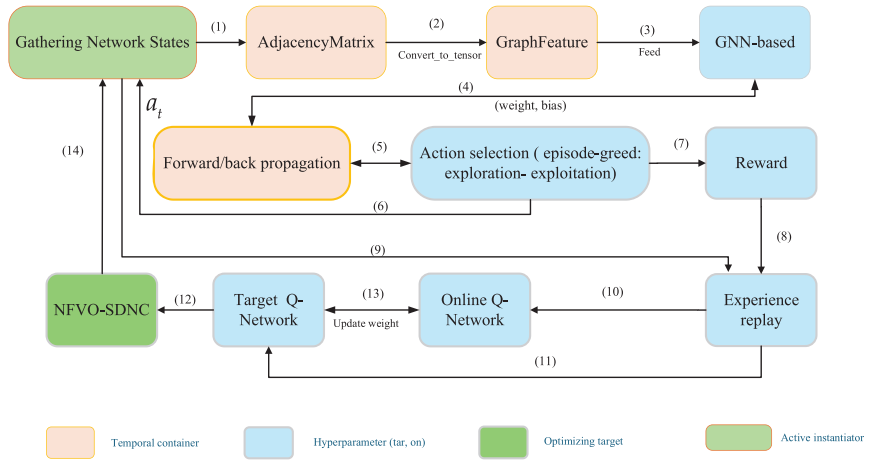

Figure 2: An architecture of DRL-GNN embedding in SDNC-NFVO for service priority

4.1 Priority-Awareness-Based Approach

We study the significance of resource management MEC servers-assisted in SDN/NFV controller that enhances the orchestration and management flow entries of IoT workload. SDN/NFV-assisted MEC environment, where it allocates the resource cost, link, bandwidth, and required resource to instantiate per VNFs. We propose leveraging a GNN-enabled DRL algorithm, namely a priority-aware VNF deployment algorithm (PVD-GDR).

4.1.1 State

State spaces indicate the significance of gathering which consists of C_v^{rem} as the remaining resources of a physical MEC server node $v \in V$ at $time - t$. It is a vector that tracks the amount of CPU, memory, and disk resources that are still available at $node - v$. In service mode, a set of service requests (VNFs and VLs) that have not yet been deployed or fully embedded into the substrate network denotes as PR at $time - t$.

$$s_t = \left\{ C_v^{rem}, BW_v^{rem}, B_{\overline{e}_{f,j}}, PR \right\} \tag{7}$$

4.1.2 Action

The decision-maker and flow selection, $x^v_{\overline{v}_{f,i}}(t)$, defines the status of deploying VNF in service requests, which can be a variable decision, adjustable between 0 and 1, to determine placement. It considers key factors such as service workloads, application criticality, and network congestion. Similarly, $y^e_{\overline{e}_{f,j}}(t)$ is also defined as a decision variable for virtual link between VNFs, which outputs 0 and 1.

$$a_t = \left\{ x^v_{\overline{v}_{f,i}}(\mathrm{t}), y^e_{\overline{e}_{f,j}}(\mathrm{t}) \right\} \tag{8}$$

4.1.3 Reward

The evaluation of agent and SDNC/NFVO control policy from overall open flow that maximize the long-term reward expectation by experience from each iteration. In our proposed method, the reward, denoted as r_t, is a compete formulation for resource utilization on computational resource, delay, allocation resource, and placement virtual links.

$$r_t = \left\{ \sum_{f \in F} \left(\sum_{\overline{v}_i \in V} \sum_{v \in V} C_i^v x^v_{\overline{v}_{f,i}}(t) + \sum_{\overline{e}_j \in \overline{E}} \sum_{e \in E} C_j^e y^e_{\overline{e}_{f,j}} \right) + \sum_{f \in F} D_f(t) \right\} \tag{9}$$

4.1.4 Optimal Policy Selection

Q-value and loss optimization concentrate on improving long-term goals and adjusting parameters for deep neural networks that act as function approximators. The target network utilizes replay batch parameters throughout the training process. Nevertheless, the loss values can vary during each training session, sometimes higher or lower than the intended level. The loss is calculated using the mean squared error (MSE), which assesses the squared difference between the target Q-value $Q(s, a)$ and the predicted Q-value. By attaining an optimal approximation of the Q-value, the agent can make choices that boost the performance of real-time services while maintaining long-term efficiency. Optimal $\pi(t)^*$ maximizes the expected cumulative reward over the $time - t$, which is presented in Eq. (10). The value function defines the expected cumulative reward obtained by following Eq. (11), which calculated by the bellman equation.

$$\pi(t)^* = \underset{\pi}{argmax}\, \mathbb{E}_\pi \left[\sum_{t \in T} \gamma_t r_t \right] \tag{10}$$

$$\mathcal{Q}^*(s,a) = \mathbb{E}_{s'} \left[r_t + \gamma \underset{a'}{argmax}\, \mathcal{Q}^*(s',a') \right] \tag{11}$$

4.2 GNN-Based Resource Embedding

In primary GNN, an input feature was received from SDN/NFV-assisted MEC entities. GNN model defines three phases: message passing, aggregation and update, and readout. Resource capacities (e.g., create the sequential of VNFs and VLs), which appended into hidden state h_i^t, h_j^t. It includes the feature of the link e_{ij}^t, which provides a combined message into an aggregated message m_i^t. After that, $M(.)$ encodes information with connected nodes in graph. The function updates the node states, $U(.)$ produces state vector h_i^{t+1}, as the transition from the connection between VNF nodes is updated. After GNN proceeds, message-passing iterations are used to propagate information from $node-1$ to $node{-}i$. In the readout of the last processing in the GNN phrase, GNN produces output prediction features two output types: node-level and link-level.

4.3 DQN-Based MEC Resource Pooling

For network environments and agent initiatives, frameworks are leveraged to design and implement including the primary algorithm of DRL-GNN, which uses DQN in online networks and target network optimization. Algorithm 1 presents the PVD-GDR function in approximating $\mathcal{Q}_{value}$ and selects the optimal action in the policy setting. The current state gathered from MEC servers' information through the environment interface is represented as $[s_t^i, s_t^j]$. The initial GNN weight and architecture, which sets up the GNN model by initializing its weight and defining its architecture, including the number of types, which consist of layers, activation functions, and other parameters. Through exploration and exploitation (epsilon greedy), the environment is observed as graph topology and state information. Within this loop, action is selected randomly after obtaining (s_{t+1}, r_t), which leads to a new state, and the reward is stored in the replay memory. It regularly appends a mini-batch size of experiences from buffer memory to enhance the online Q-network by applying gradient descent on the difference between the predicted and target $\mathcal{Q}_{value}$. The target Q-network is updated periodically to align with the online network and ensure stable learning. This cycle is carried out over several episodes, as it will continue fine-tuning the policy and identifying the optimal action a_t^*, thereby ensuring that the algorithm aligns with the development of an effective resource allocation and VL placement. It enables the MEC server to develop and improve resource allocation methods, ultimately providing the best action for each time slot according to the acquired policy.

Algorithm 1: DRL-GNN optimizes policy in priority resource

Require: MEC server resources $[s_t^i, s_t^j]$
Output: Optimal a_t action selection and update $\varnothing^{online}$, $\varnothing^{target}$
1: **For** each episode do
2: Observe state of MEC servers $[s_t^t, s_t^t]$ using *env_int*
3: ***while*** *not done* do
4: Construct GNN from s_t
5: Pass GNN
6: Approximate $\mathcal{Q}_{value}$

(Continued)

Algorithm 1 (continued)

```
 7:        If exploration
 8:            each a_t do
 9:            Observe (s_{t+1}, r_t) after executing a_t to environment
10:            Store (s_t, a_t, r_t, s_{t+1}) in replay memory B
11:            Update GNN with s_{t+1}
12:            Construct GNN from the state (t + 1)
13:            GNN passes (t + 1)
14:            Q_value ← comput.Q_value (s_t, r_t)
15:        else if exploitation do
16:            Select action a_t = argmax_a Q(s_t, a_t)a_t, ∅^online
17:            Update weight ∅^target and ∅^online
18:            Optimal action a_t^* = argmax_a Q(s_t, a_t)a_t, ∅^online
19:        end if
20:    end while
21: end
```

5 Simulation and Discussion

5.1 Simulation Parameter and Specification

Simulation parameters and specifications in network scenarios are used to define the operation in terms of resource utilization and optimization for intended resource management to achieve satisfaction based on resource priority and demands. Hence, the parameter is conducted with primary parameters, and hyperparameters used in this softwarization and virtualization framework are described in Table 2. Specifically, Mininet and TensorFlow are used to conduct our proposed.

- Mininet and mini-NFV: it creates realistic network topologies and resource management for network environments [31,32]. Ryu creates the flow manager over OVS [33]. The number of edge devices is 300 connected to 5 MEC servers and the maximum delay on a link is allowable, which ranges from 5–15 ms per ISF (inference service function). MEC servers are utilized to handle computation in closed-to IoT devices which allocate the resource for computing tasks to address the latency and avoid the traffic congestion. Hence, the placement of resources in instantiating VNF is 4 in each service request. Setting upper-bound bandwidth is 20 MHz [34].
- TensorFlow setup: it provides the design for relevance for running GNN and DRL models by defining the functionality to initialize the environment [35,36]. The episode is set to 500, the replay memory size is set to 50,000, and the batch size is set to (64, 128, 256). The learning rate, discount factor, hidden layer, and exploration rate are set to 0.001, 0.95, (32–256), and 0.5, respectively.

With the above given simulation scenarios, we dive to address our management approach and reference approach in the controller to evaluate the performance. Network performance is conducted in several key performance indicators such as the cumulative reward evaluation on metrics, end-to-end task execution delay of requested service, delivery ratio tasks of requested service, and task drop ratio of requested service.

Table 2: Determine the simulation setting and hyperparameter

Parameters	Specification	Parameters	Specification
MEC servers	5	Edge devices	300
Service priority	3	Network conditions	10
Payload size	1024 bytes	Mininet/Mini-NFV	RYU/TOSCA NFV template
VNF length per SFC	(4–10 VNFs)	Episode	500
Upper-bound delay on the link	5–15 ms per ISF	Upper-bound bandwidth	20 MHz
Traffic rate	1000–5000 pkt/s	Simulation times	500 s
Replay memory sizes	50,000	Batch size	(64, 128, 256)
Hidden layer	(32–256)	Learning rate	0.001
Discount factor	0.95	Exploration rate	0.5

In our proposed scheme, the suggested approach collects state features and the experienced path to dynamically modify critical weight to address the unpredictability of congestion and performance fluctuations. The designated simulation parameters are conducted with simulation scenarios randomly from normal to heavy congestion. To demonstrate the performance efficiency of the proposed approach, three reference schemes are chosen for experimentation and comparison as follows:

- Load-balancing shows about dedicating network selection which calculates the target network selection and workloads evenly across available resources to optimize system performance and prevent resource overloading. This approach assigns VNFs to SFC deployment based on their priority levels, ensuring critical functions are handled with higher preference while balancing the overall system load. However, it cannot overcome the priorities constraints of VNFs in priority demands.
- Greedy approach is a conventional approach that leads to finding the optimal policy in terms of cost or resource allocation. While greedy approach can be effective for smaller networks or scenarios with less complexity, it may struggle in large-scale, dynamic environments.
- Deep Q-learning (DQL) indicates the sole reinforcement learning-based SDNC method, where a single Q-learning algorithm is utilized to manage the MEC network resource. The agent acquires knowledge via trial and error, refining its policy by repeatedly querying through the Q-learning process (for instance, by employing q-table).

5.2 Performance Evaluation and Discussion

This sub-section presents the results of both the proposed and reference schemes based on five key performance metrics. Each scheme adheres to the simulation configuration for the overall topologies. Fig. 3a demonstrates how the agent explored and exploited. The evaluation illustrates how each algorithm performed throughout the episodes in terms of effectively exploring and exploiting resources. In the initial episodes, the greedy algorithm recorded a negative reward of −20.51, considerably lower than DQL and load-balancing which achieved −10.01 and −15.12, respectively. Furthermore, the PVD-GDR algorithm exhibited a positive reward of 1.29, showcasing its ability to adapt to resource exploration and exploitation early on. At the final episode, PVD-GDR reaches a cumulative reward of 40.08, indicating optimal exploitation and effective prioritization of SFC deployments under varying resource constraints. PVD-GDR reaches a reward of 40.08, which is 33.07, 27.69, and 18.74 of load-balancing, greedy, and DQL, respectively. The PVD-GDR algorithm successfully balances the trade-offs among costs, latency, and resource utilization while adapting to congestion states. Its multi-faceted weighted reward guarantees that priority-aware deployment of SFCs can manage complex, real-time IoT service requests with different priority classes. The algorithm reduces

non-optimal orchestrations, ensuring reliable and scalable SFC deployments for various use cases, especially in settings that require dynamic adaptability and efficient resource allocation. In Fig. 3b, in the initial stages, the load-balancing algorithm achieved a delay of 12.02 ms, which was slightly higher than greedy of 11.19 ms and DQL of 11.16 ms. PVD-GDR algorithm shown delay of 10.12 ms, reflecting its ability to manage task execution more efficiently from the outset. As the simulation progressed, PVD-GDR continued to maintain relatively stable and lower delays. At the final simulation timeslot, PVD-GDR algorithm excels in reducing the task execution delay by effectively balancing resource utilization and delay optimization. This makes PVD-GDR the most effective and reliable choice for managing real-time SFC in a dynamic and large-scale network environment. Fig. 3c presents in processing for real-time applications on IoT devices, QCI establishes the necessary upper-limit tolerable delays lower than those for lower critical application. In the initial stages, PVD-GDR achieved a delivery ratio of 99.9985% at 50 ms, PVD-GDR maintained a high delivery ratio, reaching 99.9001% at 500 ms. This result is greater than load-balancing, greedy, and DQL of 0.1675%, 0.0804%, and 0.2802%, respectively. The proposed PVD-GDR algorithm excels in delivering high success rates for service tasks due to its proactive resource management, ensuring that VNFs are appropriately positioned and efficiently allocated. Fig. 3d shows the task drop ratio of requested service where network states are configured in succession based on increasing congestion levels and the complexity of application tasks. We assess the flexibility of the agent by examining the variations in results in relation to the diversity of intense tasks and congestion conditions. Results demonstrate that PVD-GDR significantly outperforms load-balancing, greedy, and DQL in minimizing the task drop ratio over increasing simulation times. While greedy struggles with consistently high task drops and DQL shows moderate improvement, PVD-GDR ensures minimal task drops even under increasing workloads. This highlights its robustness and reliability for efficient task allocation in dynamic and resource-constrained environments.

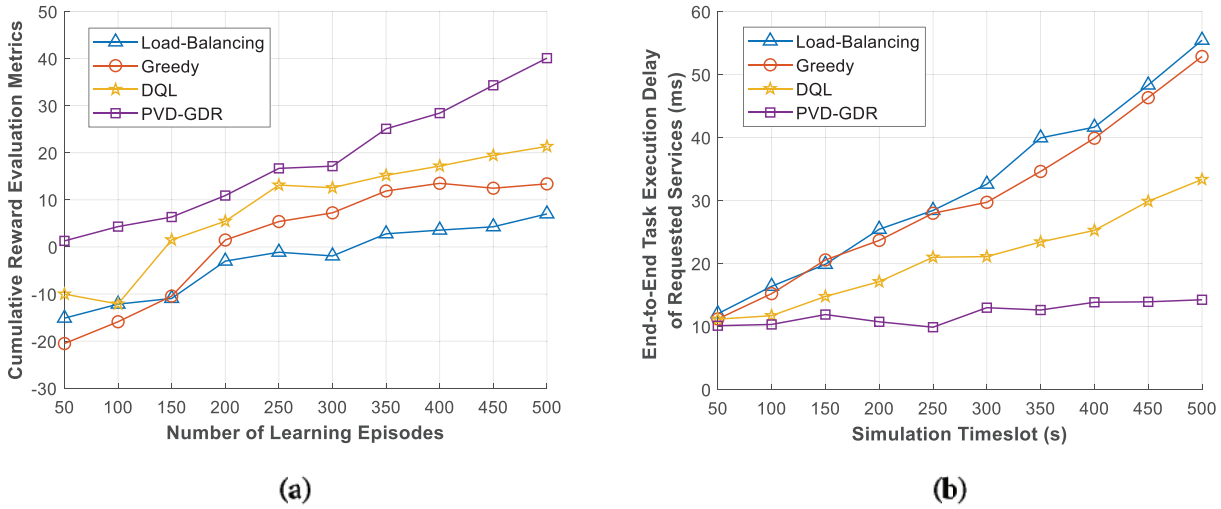

Figure 3: (Continued)

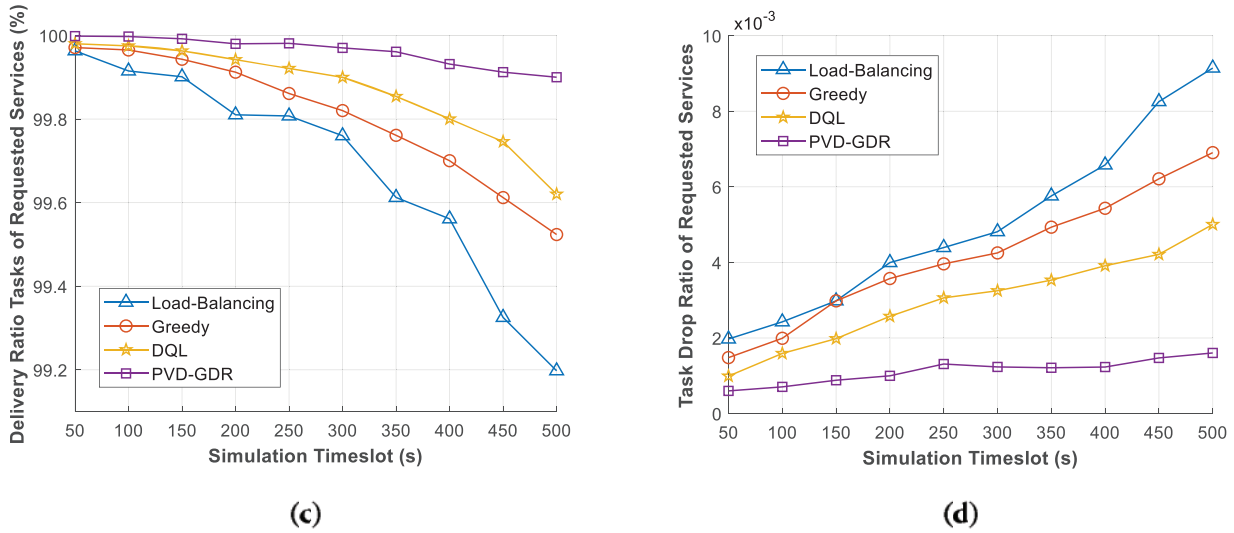

Figure 3: Result of proposed and reference schemes on (**a**) cumulative reward evaluation metric, (**b**) end-to-end execution delay, (**c**) delivery ratio, and (**d**) task drop ratio of requested service

Fig. 4 presents the average completion ratios of the requested service tasks throughout the simulation timeslots. PVD-GDR continues to demonstrate exceptional performance, reaching 99.88% at the final timeslot, which is higher than load-balancing, greedy, and DQL, thereby maintaining a steady and high completion ratio throughout. The superior performance of PVD-GDR can be attributed to its efficient handling of resource allocation, ensuring that tasks are completed successfully even under dynamic network conditions. PVD-GDR optimizes the completion of service tasks, making it an ideal choice for environments that require high reliability and low-latency performance.

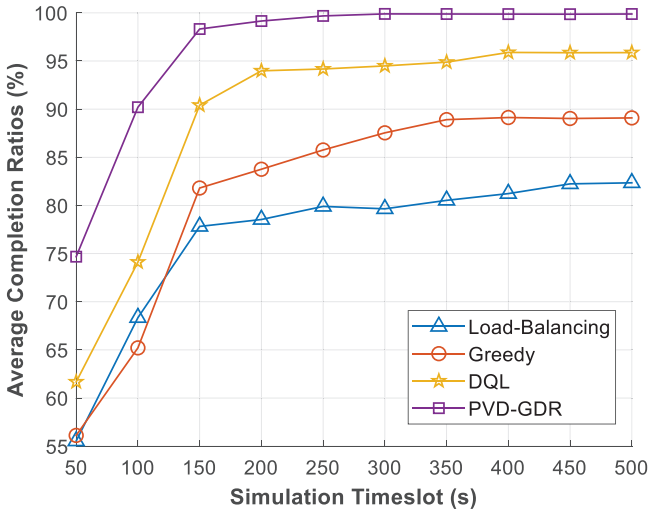

Figure 4: Average completion ratio of requested services

6 Conclusion and Future Works

This paper presented a priority-aware resource allocation for VNF deployment in MEC environments for IoT devices. The proposed scheme utilized MEC servers and SDN/NFV to capture traffic flow control and manage the resource orchestration to achieve scalable resource computation. We proposed DRL-GNN algorithm called the Priority-Aware VNF Deployment Algorithm (PVD-GDR) integrated with SDN/NFV architecture. Additionally, GNN is leveraged to execute the network environment in three phases: hidden layer, aggregation, and readout. On the other hand, we designed resource allocation and placement in SFC deployment to execute massive resource constraints in IoT services aligned with their priority level based on DQN. This PVD-GDR approach effectively prioritizes high-priority tasks while minimizing the impact on lower-priority tasks, ensuring QoS for critical applications. Finally, we conduct a simulation to evaluate efficiency of PVD-GDR, demonstrating its superior performance compared to reference schemes. Our proposed enhances efficient resource utilization and optimizes for diversity service priority on IoT device demands. In future work, we aim to extend simulation to an MEC-enabled O-RAN system, supporting diversity platforms. Additionally, we explore MARL to handle different criticalities in IoT networks.

Acknowledgement: Not applicable.

Funding Statement: This work was supported by Institute of Information & Communications Technology Planning and Evaluation (IITP) grant funded by the Korean government (MSIT) (No. RS-2022-00167197, Development of Intelligent 5G/6G Infrastructure Technology for the Smart City); in part by the National Research Foundation of Korea (NRF), Ministry of Education, through the Basic Science Research Program under Grant NRF-2020R1I1A3066543; in part by BK21 FOUR (Fostering Outstanding Universities for Research) under Grant 5199990914048; and in part by the Soonchunhyang University Research Fund.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Seyha Ros, Seungwoo Kang, Seokhoon Kim; data collection: Seyha Ros, Seungwoo Kang, Taikuong Iv; analysis and interpretation of results: Seyha Ros, Prohim Tam, Inseok Song; draft manuscript preparation: Seyha Ros, Taikuong Iv, Seungwoo Kang, Inseok Song, Prohim Tam, Seokhoon Kim. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Mostafavi S, Hakami V, Sanaei M. Quality of service provisioning in network function virtualization: a survey. Computing. 2021;103(5):917–91. doi:10.1007/s00607-021-00925-x.
2. Yi B, Wang X, Li K, Das SK, Huang M. A comprehensive survey of network function virtualization. Comput Netw. 2018;133(2):212–62. doi:10.1016/j.comnet.2018.01.021.
3. Shahraki A, Taherkordi A, Haugen Ø, Eliassen F. A survey and future directions on clustering: from WSNs to IoT and modern networking paradigms. IEEE Trans Netw Serv Manag. 2021;18(2):2242–74. doi:10.1109/TNSM.2020.3035315.
4. Bizanis N, Kuipers FA. SDN and virtualization solutions for the Internet of Things: a survey. IEEE Access. 2016;4:5591–606. doi:10.1109/ACCESS.2016.2607786.
5. Siddiqui S, Hameed S, Shah SA, Ahmad I, Aneiba A, Draheim D, et al. Toward software-defined networking-based IoT frameworks: a systematic literature review, taxonomy, open challenges and prospects. IEEE Access. 2022;10(20):70850–901. doi:10.1109/ACCESS.2022.3188311.

6. Bera S, Misra S, Vasilakos AV. Software-defined networking for Internet of Things: a survey. IEEE Internet Things J. 2017;4(6):1994–2008. doi:10.1109/JIOT.2017.2746186.
7. ETSI GS NFV 002 V1.2.1 (2014-12). Network functions virtualisation (NFV). In: Architectural framework. Nice, France: ETSI; 2014.
8. Abderrahmane A, Drid H, Behaz A. A survey of controller placement problem in SDN-IoT network. Int J Networked Distrib Comput. 2024;12(2):170–84. doi:10.1007/s44227-024-00035-y.
9. Alam I, Sharif K, Li F, Latif Z, Karim MM, Biswas S, et al. A survey of network virtualization techniques for Internet of Things using SDN and NFV. ACM Comput Surv. 2021;53(2):1–40. doi:10.1145/3379444.
10. Cao B, Zhang J, Liu X, Sun Z, Cao W, Nowak RM, et al. Edge-cloud resource scheduling in space-air–ground-integrated networks for Internet of vehicles. IEEE Internet Things J. 2022;9(8):5765–72. doi:10.1109/JIOT.2021.3065583.
11. Hantouti H, Benamar N, Taleb T. Service function chaining in 5G & beyond networks: challenges and open research issues. IEEE Netw. 2020;34(4):320–7. doi:10.1109/MNET.001.1900554.
12. Yang S, Li F, Trajanovski S, Yahyapour R, Fu X. Recent advances of resource allocation in network function virtualization. IEEE Trans Parallel Distrib Syst. 2021;32(2):295–314. doi:10.1109/TPDS.2020.3017001.
13. Ros S, Tam P, Song I, Kang S, Kim S. Handling efficient VNF placement with graph-based reinforcement learning for SFC fault tolerance. Electronics. 2024;13(13):2552. doi:10.3390/electronics13132552.
14. Tam P, Math S, Kim S. Priority-aware resource management for adaptive service function chaining in real-time intelligent IoT services. Electronics. 2022;11(19):2976. doi:10.3390/electronics11192976.
15. Farkiani B, Bakhshi B, Ali MirHassani S, Wauters T, Volckaert B, De Turck F. Prioritized deployment of dynamic service function chains. IEEE/ACM Trans Netw. 2021;29(3):979–93. doi:10.1109/TNET.2021.3055074.
16. Jang I, Suh D, Pack S, Dán G. Joint optimization of service function placement and flow distribution for service function chaining. IEEE J Sel Areas Commun. 2017;35(11):2532–41. doi:10.1109/JSAC.2017.2760162.
17. Chen Z, Xiong B, Chen X, Min G, Li J. Joint computation offloading and resource allocation in multi-edge smart communities with personalized federated deep reinforcement learning. IEEE Trans Mob Comput. 2024;23(12):11604–19. doi:10.1109/TMC.2024.3396511.
18. Zhou C, Zhao B, Tang F, Han B, Wang B. Dynamic multi-objective service function chain placement based on deep reinforcement learning. IEEE Trans Netw Serv Manag. 2024;PP(99):1. doi:10.1109/TNSM.2024.3446248.
19. Pati PS, Sonu M, Datta R, Krishnaswamy D, Singhal C. PARS: a priority-aware resource sharing scheme for services with differential QoS requirements in B5G edge network. IEEE Netw Lett. 2023;5(2):105–9. doi:10.1109/LNET.2023.3256087.
20. Cai Q, Zhou Y, Liu L, Qi Y, Shi J. Prioritized assignment with task dependency in collaborative mobile edge computing. IEEE Trans Mob Comput. 2024;23(12):13505–21. doi:10.1109/TMC.2024.3427380.
21. Jamil B, Ijaz H, Shojafar M, Munir K. IRATS: a DRL-based intelligent priority and deadline-aware online resource allocation and task scheduling algorithm in a vehicular fog network. Ad Hoc Netw. 2023;141(7):103090. doi:10.1016/j.adhoc.2023.103090.
22. Zhang J, Liu Y, Li Z, Lu Y. Forecast-assisted service function chain dynamic deployment for SDN/NFV-enabled cloud management systems. IEEE Syst J. 2023;17(3):4371–82. doi:10.1109/JSYST.2023.3263865.
23. Li Y, Zhang X, Zeng T, Duan J, Wu C, Wu D, et al. Task placement and resource allocation for edge machine learning: a GNN-based multi-agent reinforcement learning paradigm. IEEE Trans Parallel Distrib Syst. 2023;34(12):3073–89. doi:10.1109/TPDS.2023.3313779.
24. Li B, Zhu Z. GNN-based hierarchical deep reinforcement learning for NFV-oriented online resource orchestration in elastic optical DCIs. J Light Technol. 2022;40(4):935–46. doi:10.1109/JLT.2021.3125974.
25. Pan P, Fan Q, Wang S, Li X, Li J, Shi W. GCN-TD: a learning-based approach for service function chain deployment on the fly. In: Proceedings of the IEEE Global Communications Conference (GLOBECOM); 2020 Dec 7–11; Taipei, Taiwan. p. 1–6. doi:10.1109/globecom42002.2020.9322359.
26. Huang W, Li S, Wang S, Li H. An improved adaptive service function chain mapping method based on deep reinforcement learning. Electronics. 2023;12(6):1307. doi:10.3390/electronics12061307.

27. Shokrnezhad M, Taleb T. ORIENT: a priority-aware energy-efficient approach for latency-sensitive applications in 6G. arXiv:2402.06931. 2024.
28. Khemani B, Patil S, Kotecha K, Tanwar S. A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. J Big Data. 2024;11(1):18. doi:10.1186/s40537-023-00876-4.
29. Tam P, Song I, Kang S, Ros S, Kim S. Graph neural networks for intelligent modelling in network management and orchestration: a survey on communications. Electronics. 2022;11(20):3371. doi:10.3390/electronics11203371.
30. Di Martino B, Cretella G, Esposito A. Defining cloud services workflow: a comparison between *TOSCA* and OpenStack hot. In: Proceedings of the 2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems; 2015 Jul 8–10; Santa Catarina, Brazil. p. 541–6. doi:10.1109/CISIS.2015.80.
31. Lantz B, O'Connor B. A mininet-based virtual testbed for distributed SDN development. SIGCOMM Comput Commun Rev. 2015;45(4):365–6. doi:10.1145/2829988.2790030.
32. Castillo-Lema J, Venancio Neto A, de Oliveira F, Takeo Kofuji S. Mininet-nfv: evolving Mininet with oasis *Tosca* nvf profiles Towards Reproducible nfv Prototyping. In: Proceedings of the IEEE Conference on Network Softwarization (NetSoft); 2019 Jun 24–28; Paris, France. p. 506–12. doi:10.1109/netsoft.2019.8806686.
33. Asadollahi S, Goswami B, Sameer M. Ryu controller's scalability experiment on software defined networks. In: Proceedings of the IEEE International Conference on Current Trends in Advanced Computing (ICCTAC); 2018 Feb 1–2; Bangalore, India. p. 1–5. doi:10.1109/ICCTAC.2018.8370397.
34. ETSI TS 138 101-1 V15.2.0 (2018-07). 5G; NR; User Equipment (UE) radio transmission and reception. In: Part 1: range 1 standalone (3GPP TS 38.101-1 version 15.2.0 Release 15). Nice, France: ETSI; 2018.
35. Ferludin O, Eigenwillig A, Blais M, Zelle D, Pfeifer J, Sanchez-Gonzalez A, et al. TF-GNN: graph neural networks in TensorFlow. arXiv:2207.03522. 2022.
36. Ajay Rao P, Navaneesh Kumar B, Cadabam S, Praveena T. Distributed deep reinforcement learning using TensorFlow. In: Proceedings of the 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC); 2017 Sep 8–9; Mysore, India. p. 171–4. doi:10.1109/CTCEEC.2017.8455196.