FlagVNE: A Flexible and Generalizable Reinforcement Learning Framework for Network Resource Allocation

Tianfu Wang[1] , **Qilin Fan**[2] , **Chao Wang**[3,1*] ,
Long Yang[4] , **Leilei Ding**[1] , **Nicholas Jing Yuan**[5] and **Hui Xiong**[6*]
[1]School of Computer Science and Technology, University of Science and Technology of China
[2]School of Big Data and Software Engineering, Chongqing University
[3]Guangzhou HKUST Fok Ying Tung Research Institute
[4]School of Artificial Intelligence, Peking University
[5]Microsoft
[6]Thrust of Artificial Intelligence, The Hong Kong University of Science and Technology (Guangzhou)
{tianfuwang,dingleilei}@mail.ustc.edu.cn, fanqilin@cqu.edu.cn,chadwang2012@gmail.com
yanglong001@pku.edu.cn,nicholas.yuan@microsoft.com,xionghui@ust.hk

Abstract

Virtual network embedding (VNE) is an essential resource allocation task in network virtualization, aiming to map virtual network requests (VNRs) onto physical infrastructure. Reinforcement learning (RL) has recently emerged as a promising solution to this problem. However, existing RL-based VNE methods are limited by the unidirectional action design and one-size-fits-all training strategy, resulting in restricted searchability and generalizability. In this paper, we propose a FLexible And Generalizable RL framework for VNE, named FlagVNE. Specifically, we design a bidirectional action-based Markov decision process model that enables the joint selection of virtual and physical nodes, thus improving the exploration flexibility of solution space. To tackle the expansive and dynamic action space, we design a hierarchical decoder to generate adaptive action probability distributions and ensure high training efficiency. Furthermore, to overcome the generalization issue for varying VNR sizes, we propose a meta-RL-based training method with a curriculum scheduling strategy, facilitating specialized policy training for each VNR size. Finally, extensive experimental results show the effectiveness of FlagVNE across multiple key metrics. Our code is available at https://github.com/GeminiLight/flag-vne.

1 Introduction

Network virtualization (NV) emerges as a pioneering technology that facilitates dynamic management of Internet architecture, which finds applications in 5G networks and cloud computing [Zhuang *et al.*, 2020]. Through network slicing and shared infrastructure, NV enables the deployment of multiple user-submitted virtual network requests (VNRs) within the same physical network, thereby accommodating diverse network service requirements of users [Yang *et al.*, 2021; Chen *et al.*, 2022b]. The primary challenge in NV involves the embedding of VNRs within a physical network, known as virtual network embedding (VNE), an NP-hard combinatorial optimization problem [Rost and Schmid, 2020].

Effective resource allocation for VNRs is essential to improve the quality of service and the revenue of Internet service providers (ISPs) [Wang *et al.*, 2021b; Chen *et al.*, 2022a]. Regrettably, it is hard to address the VNE problem involving tackling combinatorial explosion and differentiated demands [Fischer *et al.*, 2013; Yang *et al.*, 2022b]. On the one hand, the solution space of VNE is extensive, encompassing vast permutations of VNRs within the underlying physical network. Consequently, a comprehensive exploration of this expansive solution space becomes imperative to determine superior solutions. On the other hand, due to specific requirements of user service, the integration of diverse VNR topologies and their associated resource demands is dynamic. VNRs of varying sizes manifest unique complexities, rendering a one-size-fits-all strategy inadequate to effectively manage the inherent variability in such circumstances.

Recently, reinforcement learning (RL) has shown promising potential for the VNE problem [Yan *et al.*, 2020; He *et al.*, 2023b; Zhang *et al.*, 2023]. RL approaches model the solution construction process of each VNR as Markov decision processes (MDPs), which can automatically build efficient solving policies. Unlike supervised learning relying on labeled data, RL facilitates the learning of effective heuristics through interactions with the environment. However, most existing RL-based VNE approaches are still plagued with some significant issues. Firstly, these approaches commonly adhere to a unidirectional action design within the MDP, i.e., presupposing a fixed decision sequence for virtual nodes, and subsequently designating a physical node to host each virtual node sequentially. Such unidirectional action schema significantly limits the available action space, consequently constraining the searchability of the agent and impeding the effi-

*Chao Wang and Hui Xiong are both corresponding authors cacy of exploring solution space. Secondly, conventional RL-based methods usually just train a single general policy, disregarding the distinctive complexities of VNRs with varying sizes in practice. Treating variable-sized VNRs equally poses challenges in achieving balanced learning of cross-size strategic knowledge and hinders the ability to generalize across VNRs of differing sizes. Thirdly, the direct training of multiple policies tailored to different VNR sizes slowly adapts to unseen distributions. In particular, training specific policies for large-sized VNRs from scratch tends to be stuck in the local optimum, due to the high complexity and challenges in exploring feasible solutions. These difficulties inevitably exert negative impacts on overall system performance. We conduct a preliminary study to highlight our motivations and latent challenges, which is detailed in Appendix A.

In this paper, we propose a novel **FL**exible **A**nd **G**eneralizable RL framework for the **VNE** problem, named **FlagVNE**. Our framework aims to enhance the searchability and generalizability of RL-based VNE methods while achieving rapid adaption to the unseen distribution of VNR sizes. Specifically, our contributions are summarized as follows. (1) We propose a bidirectional action-based MDP modeling approach to enable the joint selection of virtual and physical nodes, enhancing the flexibility of agent exploration and exploitation. This method offers superior searchability and is proven theoretically. To handle the resulting large and changeable action space, we abstract it as two dependent aspects and design a hierarchical decoder with a bilevel policy, ensuring adaptive action probability distribution generation and high training efficiency. (2) We propose a meta-RL-based training method to enable efficient acquisition of multiple size-specific policies and quick adaptation to new sizes. A meta-policy is trained to grasp cross-size knowledge for different VNR sizes and then fastly fine-tuned to develop size-specific policies for each VNR size, even unseen sizes. Specially, due to difficult exploration and prone to suboptimal convergence, using large-sized VNRs for initial meta-learning yields inferior knowledge, impairing the meta-policy and generalization. Thus, we develop a curriculum scheduling strategy that gradually incorporates larger VNRs, alleviating suboptimal convergence. (3) Finally, we conduct experiments on the simulation platform to mimic various network systems and extensive results demonstrate the superiority of FlagVNE in terms of multiple key indicators, compared to state-of-the-art (SOTA) heuristics and RL-based methods.

2 Related Work

Traditional Methods for VNE. Initially, the VNE problem was tackled using exact methods such as integer linear programming [Shahriar *et al.*, 2018], which provides optimal solutions through exact solvers. However, these exact algorithms proved impractical for real-world scenarios due to their time-consuming nature. Thus, numerous heuristic algorithms have been proposed to find solutions in an acceptable time [Su *et al.*, 2014; Jin *et al.*, 2020; Fan *et al.*, 2023]. Among these approaches, node ranking is a prevalent strategy, which ranks virtual and physical nodes to determine the decision sequence and the matching priority, respectively. For example, [Zhang *et al.*, 2018] ranked nodes based on a node resource management (NRM) metric, and [Fan *et al.*, 2023] proposed a node essentiality assessment (NEA) metric considering topology connectivity. Additionally, [Dehury and Sahoo, 2019] designed VNE algorithms based on metaheuristics, such as particle swarm optimization (PSO). However, these algorithms heavily rely on manual designs and are usually tailored to specific scenarios, limiting their performance in general cases.

Learning-based Methods for VNE. Recently, machine learning techniques have been used to solve VNE, leading to faster and more efficient solutions [Blenk *et al.*, 2018; Geng *et al.*, 2023; He *et al.*, 2023b]. Particularly, RL has demonstrated significant potential as an intelligent decision-making framework [Liu *et al.*, 2023; Yang *et al.*, 2022a], which can effectively solve VNE with MDP modeling. In this paper, we unify most existing RL-based methods [Xiao *et al.*, 2019; Wang *et al.*, 2021c; Yan *et al.*, 2020; Yao *et al.*, 2020; Zhang *et al.*, 2022; Zhang *et al.*, 2023] into a general framework comprised of three key components: *MDP modeling*, *policy architecture*, and *training methods*. These methods model the process of VNE solution construction as unidirectional action-based MDPs, where a physical node is chosen to host a be-placing virtual node, and the decision sequence of virtual nodes is fixed. Then they build policy models with various neural networks and train a single general policy to deal with VNRs of varying sizes. For instance, [Xiao *et al.*, 2019] used multilayer perception (MLP) as a policy model and trained it with policy gradient (PG) algorithm, [Zhang *et al.*, 2023] designed a policy model with MLP and graph convolutional network (GCN) [Kipf and Welling, 2017] and trained it with asynchronous advantage actor-critic (A3C) [Mnih *et al.*, 2016]. However, existing RL-based VNE methods suffer from limited searchability and generalizability due to their unidirectional action design and one-size-fits-all training policy, ultimately affecting overall performance.

3 Preliminaries

3.1 Problem Definition

As shown in Fig. 1, in a practical network system, users' service requests arriving continuously are represented as VNRs. We collect all VNRs with a set $\mathcal{V}$. Mapping these VNRs onto physical networks managed by ISPs is known as VNE, crucial in managing the quality of various network services [Chen *et al.*, 2020; Wang *et al.*, 2023a; Chen *et al.*, 2022c].

System Modeling. *Physical network* is formulated as a weighted undirected graph $\mathcal{G}^p = (\mathcal{N}^p, \mathcal{L}^p)$, where $\mathcal{N}^p$ is the set of physical nodes, $\mathcal{L}^p$ is the set of physical links. Each physical node $n^p \in \mathcal{N}^p$ is equipped with multiple resource capacities $\{C(n^p), \forall C \in \mathcal{C}\}$, where $\mathcal{C}$ is the set of node resource types, and each physical link $l^p \in \mathcal{L}^p$ has bandwidth capacity $B(l^p)$. In this paper, we consider multidimensional node resources, including the central processing unit (CPU), storage resource, and graphics processing unit (GPU). Similarly, each *VNR* is modeled as a weighted undirected graph $G^v = (\mathcal{N}^v, \mathcal{L}^v, d^v)$, where $\mathcal{N}^v$ is the set of virtual nodes and $\mathcal{L}^v$ is the set of virtual links, and d^v denotes the lifetime of VNR. Once the VNR is accepted, it will be maintained for d^v

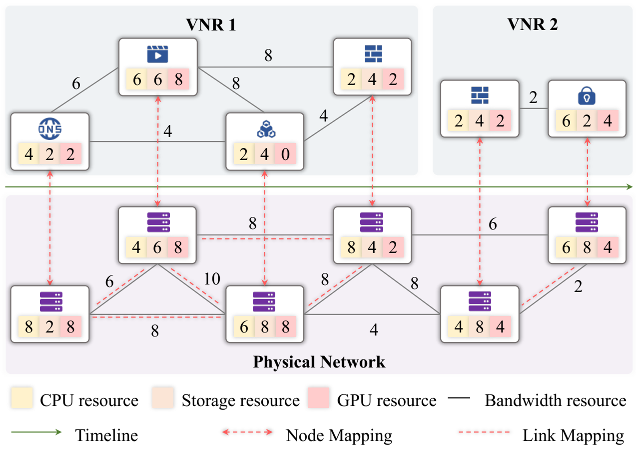


Figure 1: An example of the VNE problem with multidimensional resources. The numbers denote the unit counts of resources.

time slots. Each virtual node $n^v \in \mathcal{N}^v$ represents a virtual machine with resource demands $\{C(n^v), \forall C \in \mathcal{C}\}$ and each virtual link $l^v \in \mathcal{L}^v$ indicates the bandwidth demand $B(l^v)$.

Objective. Acknowledging the stochastic nature of online networking, most existing methods and this work aim to minimize the embedding cost of each VNR onto the physical network, which facilitates long-term performance. The quality of solutions is assessed using the revenue-to-cost ratio (R2C):

$$\text{R2C}\left(\mathcal{G}^v\right) = \left(\Psi \cdot \text{REV}\left(\mathcal{G}^v\right)\right) / \text{COST}\left(\mathcal{G}^v\right). \tag{1}$$

Here, REV$(\mathcal{G}^v)$ denotes the revenue of the VNR G^v (i.e., the sum of VNR's resource requirements) and COST$(\mathcal{G}^v)$ denotes the embedding cost resulting from the solution (i.e., the sum of ISP' resource consumption). Ψ is the binary variable that indicates the feasibility of a solution.

Constraints. The VNR embedding consists of two subprocesses. (1) *Node mapping* entails assigning each virtual node to a physical node with adequate resources, i.e., $C(n^p) \geq C(n^v), \forall C \in \mathcal{C}$, while ensuring one-to-one placement and mutual exclusivity. (2) *Link mapping* involves finding a physical path for each virtual link, ensuring that the path connects the physical nodes hosting the virtual link endpoints and that each physical link l^p in the path has sufficient bandwidth, i.e., $B(l^p) \geq B(l^v)$. A solution is deemed feasible ($\Psi = 1$) only when all these constraints are satisfied.

Due to the space limit, we place detailed formulations of VNE's objective and constraints in Appendix B.

3.2 Motivations and Challenges

We conduct a preliminary study placed in Appendix A, and motivate our framework from the following two aspects.

Flexibility of Action Space. Most existing RL-based VNE approaches employ a unidirectional action design, assuming that the decision sequence of virtual nodes is predetermined. However, our analysis in Appendix A.1 reveals that varying the decision sequences of virtual nodes significantly impacts performance. This underscores the necessity of exploring different decision sequences for optimal solutions. Moreover, the fixed decision sequence of virtual nodes lacks the flexibility needed to adapt to the dynamic nature of exploration process. Thus, to enhance the flexibility of exploration and exploitation, we aim to achieve a joint selection of both physical and virtual nodes to eliminate the fixed decision sequence. Nevertheless, it will pose some challenges, such as the difficulty of variable action probability distribution generation and the training efficiency issue caused by large action space.

Generalization of Solving Policy. VNRs of different sizes exhibit distinct complexities, necessitating varied solving strategies. Existing RL-based methods typically use a one-size-fits-all policy to tackle VNRs of varying sizes, leading to generalization issues. To address this, an intuitive approach might be to develop size-specific policies for different VNR sizes. Yet, as observed in Appendix A.2, specific policies for large-sized VNRs trained from scratch often get stuck in local optima due to their high complexity and the difficulty in exploring viable solutions. Their performance is even inferior to that of the general policy for all sizes. Furthermore, this strategy lacks the quick adaptability to handle previously unseen VNR sizes, since it requires extensive data demand.

4 FlagVNE Framework

In this section, we present the proposed RL-based framework for VNE, **FlagVNE**. As illustrated in Fig. 2, FlagVNE is designed to improve searchability and generalizability while achieving rapid adaptation to unseen distribution.

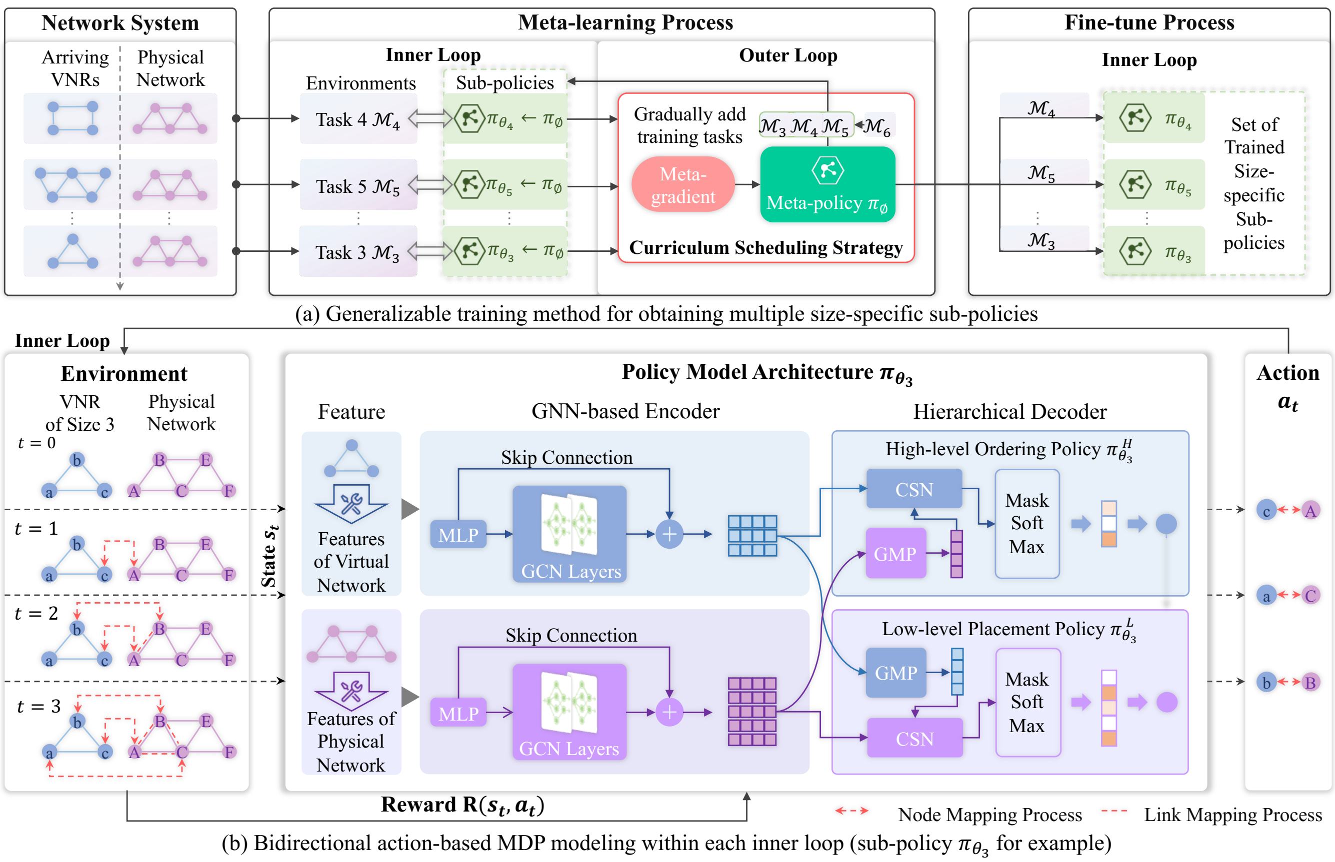
4.1 Bidirectional Action-based MDP

We formulate the solution construction process of each VNR as a bidirectional action-based MDP, allowing joint selection of virtual nodes and physical nodes. Specifically, at each decision timestep t, observing the state s_t of the environment, the agent takes an action $a_t \sim \pi(\cdot|s_t)$ according to the policy π. Then, the environment will feedback a reward $R(s_t, a_t)$ and transit to a new state $s_{t+1} \sim P(s_t, a_t)$ following the transition probability function. During interactions, a trajectory memory $\mathcal{D} = \{s_1, a_1, s_2, a_2, \cdots\}$ collects state-action pairs. We present these notations in VNE as follows.

State represents the status of the network system at a specific decision timestep t, consisting of the current situation of VNR s_t^v and physical network s_t^p, i.e., $s_t = (s_t^v, s_t^p), s_t \in \mathcal{S}$, where $\mathcal{S}$ is the state space.

Action is defined as a pair of a virtual node to be placed and a physical node to host, denoted $a_t = (n^v, n^p), a_t \in \mathcal{A}$, where $n^v \in \mathcal{N}^v$, $n^p \in \mathcal{N}^p$, and $\mathcal{A}$ is the action space.

Transition $P(s_{t+1} \mid s_t, a_t)$ refers to the process of placing the virtual node n^v to the physical node n^p and routing the virtual links, resulting in the changes of state from s_t to s_{t+1}. Based on the selected bidirectional action, the environment attempts to place the virtual node n^v to the physical node n^p. If the node placement is successful, the link routing is executed based on the breadth-first search algorithm that finds the shortest physical paths meeting bandwidth demands from n^p to other physical nodes hosting the virtual node neighbors of n^v. If node placement and link routing are successful, the available resource of the physical network is updated with the VNR requirement. Otherwise, the current VNR is rejected.



(a) Generalizable training method for obtaining multiple size-specific sub-policies

(b) Bidirectional action-based MDP modeling within each inner loop (sub-policy π_{θ_3} for example)

Figure 2: The overview of the FlagVNE framework. (a) For vary-sized VNRs that continuously arrive at the network system, we consider them as different tasks $\mathcal{M}_i \sim p(\mathcal{M})$ based on their size. We first train a meta-policy π_ϕ with cross-task knowledge in the meta-learning process, using a curriculum scheduling strategy. Then, we fine-tune it to obtain a set of size-specific sub-policies π_{θ_i}. This generalizable training method effectively obtains refined solving policies for each VNR size. (b) Within each inner loop, we formulate the solution construction process of each VNR as a bidirectional action-based MDP, which enables the joint selection of virtual and physical nodes. We also design a hierarchical encoder with a bilevel policy to adaptively generate action probability distributions and ensure high training efficiency.

Reward R measures the quality of agent's action at a given state. We define the reward function R as follows:

$$R(s_t, a_t) = \begin{cases} \text{R2C}(\mathcal{G}^v), & \text{if } \mathcal{G}^v \text{ is accepted at } t, \\ -1/|\mathcal{N}^v|, & \text{if } \mathcal{G}^v \text{ is rejected at } t, \\ 1/|\mathcal{N}^v|, & \text{otherwise.} \end{cases} \quad (2)$$

We design implicit rewards to encourage successful placement with $1/|\mathcal{N}^v|$ and punish failure with $-1/|\mathcal{N}^v|$. Once the $\mathcal{G}^v$ is completed embedding, we return R2C($\mathcal{G}^v$).

Policy is parameterized by θ, which denotes the distributions over the action space under a given state s_t:

$$\pi_\theta(a_t|s_t) = P(a_t|s_t). \quad (3)$$

Discount factor $\lambda \in (0, 1)$ balances the importance of immediate rewards versus future rewards. Overall, the optimization objective of RL is to maximize the expected return, i.e., cumulative discounted rewards over timesteps T:

$$J_\pi = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}[\sum_{t=0}^{T} \lambda^t R(s_t, a_t)]. \quad (4)$$

If $J_\pi \geq J_{\pi'}$, then we denote it as $\pi \succeq \pi'$.

Theorem 1. *Given two MDPs with bidirectional and unidirectional action, $\mathcal{M}^b = \langle \mathcal{S}^b, \mathcal{A}^b, P^b, R, \lambda \rangle$ and $\mathcal{M}^u = \langle \mathcal{S}^u, \mathcal{A}^u, P^u, R, \lambda \rangle$, and their optimal policies denoted as $\pi^{\star,b}$ and $\pi^{\star,u}$, respectively, we have $\pi^{\star,b} \succeq \pi^{\star,u}$.*

See Appendix C for its proof [Sutton and Barto, 2018]. Our bidirectional action enhances flexibility and expands the search space, allowing for a more comprehensive exploration of possible solutions, which offers superior MDP Optimality.

4.2 Hierarchical Policy Architecture

We construct raw features, encode them with a GCN-based encoder, and design a hierarchical decision module to ensure adaptive probability output and training efficiency.

Feature Constructor. We build the feature input for the subsequent encoder from the current state s_t, which includes the processing status of VNR s_t^v and the current physical network situation s_t^p. With comprehensive information on the current state, the agent gains deeper environmental insight, resulting in better decisions. For the VNR G_t^v, the feature constructor takes into account not only various node resource requirements denoted $X_{t,N}^v$, but also aggregates bandwidth resource requirements into the node features, represented as $X^v_{t,L}$. These features include essential bandwidth metrics, such as maximum, mean, and sum of bandwidth requirements of virtual links adjacent to one node. To indicate the embedding status, a placement flag $X^v_{t,P}$ is designed for virtual nodes, with a value of 1 indicating that the virtual node has been placed and 0 otherwise. The VNR features X^v_t are organized as follows: $X^v_t = (X^v_{t,N}, X^v_{t,L}, X^v_{t,P}) \in \mathbb{R}^{|N^v| \times 7}$.

Similarly, the physical network features X^p_t are constructed as follows: $X^p_t = (X^p_{t,N}, X^p_{t,L}, X^p_{t,S}) \in \mathbb{R}^{|N^p| \times 7}$, where $X^p_{t,N}$ denotes available resources of physical nodes, $X^p_{t,L}$ similar to $X^v_{t,L}$ denotes aggregated bandwidth availability, and $X^p_{t,S}$ is a selection flag indicating the status of physical nodes. A selection flag value of 1 indicates that a physical node has been selected to host a virtual node and 0 otherwise.

GNN-based Encoder. To encode the features of the virtual network X^v_t and the physical network X^p_t, into latent representations, Z^v_t and Z^p_t, respectively, we adopt a graph neural network (GNN) encoder. First, both X^v_t and X^p_t undergo the MLP to obtain the initial node representations, denoted I^v_t and I^p_t, respectively: $I^v_t = \mathrm{MLP}(X^v_t), I^p_t = \mathrm{MLP}(X^p_t)$.

Then, we consider multiple GCN [Kipf and Welling, 2017] layers as the GNN modules to obtain the latent representations of virtual nodes $\tilde{Z}^v_t$ and physical nodes $\tilde{Z}^p_t$: $\tilde{Z}^v_t = \mathrm{GNN}(I^v_t, A^v), \tilde{Z}^p_t = \mathrm{GNN}(I^p_t, A^p)$, where A^v and A^p is adjacency matrixes of virtual and physical networks.

Furthermore, to enhance the feature representation ability, we also employ the residual connection method to combine the output of the GNN module with the initial representation: $Z^v_t = \tilde{Z}^v_t + I^v_t, Z^p_t = \tilde{Z}^p_t + I^p_t$. Finally, we obtain the representation of each virtual and physical node.

Hierarchical Decoder with Bilevel Policy. In our bidirectional action-based MDP, the action space is represented by the matrix size $|\mathcal{N}^v| \times |\mathcal{N}^p|$, reflecting the number of virtual and physical nodes. The variable and often large size of VNRs contribute to the expansive and dynamic nature of the space. To effectively manage this, we develop a hierarchical decoder with a bilevel policy, ensuring high training efficiency and adaptive action probability generation. Specifically, we abstract this task into two dependent aspects: virtual node ordering and physical node placement. Our bilevel policy, $\pi(a_t|s_t) = \pi^H(n^v|s_t) \cdot \pi^L(n^p|s_t, n^v)$, consist of a high-level ordering policy $\pi^H(n^v|s_t)$ and a low-level placement policy $\pi^L(n^p|s_t, n^v)$. This hierarchical approach reduces the size of policy distribution from $|\mathcal{N}^v| \times |\mathcal{N}^p|$ to $|\mathcal{N}^v| + |\mathcal{N}^p|$, thus significantly enhancing training efficiency.

High-level ordering policy selects the appropriate virtual node n^v_t for placement. Concretely, we use an MLP-based compatibility scoring network (CSN) to calculate the fitness between each virtual node representation and the graph-level representation of the physical network $G^p_t = \mathrm{GMP}(Z^p_t)$. Here, $\mathrm{GMP}(Z) = \frac{1}{|Z|}\sum_{z \in Z} z$ denotes graph mean pooling (GMP), averaging all node representations. Then an MLP is applied to generate compatibility scores for each virtual node:

$$\tilde{Y}^H = \mathrm{MLP}(Z^v_t + G^p_t) \in \mathbb{R}^{1 \times |\mathcal{N}^v|}. \tag{5}$$

Although the VNR's sizes are variable, this layer adaptively generates scores with the shape of $(1, |\mathcal{N}^v|)$. After masking virtual nodes already placed (i.e., setting their scores to $-\infty$ on $\tilde{Y}^H$), we apply a softmax function to the resultant score Y^H to produce the high-level action probability distribution.

$$\pi^H(n^v_t|s_t) = \mathrm{softmax}(Y_H). \tag{6}$$

Low-level placement policy identifies a suitable physical node n^p_t for accommodating the to-be-placed virtual node n^v_t, which is selected by π^H. Similarly, we adopt an MLP-based compatibility scoring network to calculate the fitness between the representation of each physical node and the current context representation of virtual network, including the graph-level representation of virtual network $G^v_t = \mathrm{GMP}(Z^v_t)$ and to-be-placed virtual node's representation $z_{n^v_t}$:

$$\tilde{Y}^L = \mathrm{MLP}(Z^p_t + G^v_t + z_{n^v_t}) \in \mathbb{R}^{1 \times |\mathcal{N}^p|}. \tag{7}$$

To avoid unnecessary exploration, we mask the physical nodes that do not have enough resources or have been selected to obtain the final scores Y^L. Then, the low-level action probability distribution is generated:

$$\pi^L(n^p_t|s_t, n^v_t) = \mathrm{softmax}(Y_L). \tag{8}$$

For both two-level probability distributions, we employ the sampling and greedy strategy to select actions during the training and inference phases, respectively.

4.3 Generalizable Training Method

Training a general policy for VNRs of varying sizes leads to imbalanced learning of cross-size strategy and generalization issues. Conversely, individualized training of multiple policies for each size is slow to adapt to new sizes, in which policies for large-sized VNRs are prone to suboptimal. To address this, we develop a meta-RL-based training method with a curriculum scheduling strategy. As illustrated in Algorithm 1 (see Appendix D), our method enables efficient training of multiple size-specific policies and quick adaptation to new sizes, while balancing the learning process across tasks of varying difficulty and avoiding suboptimal convergence.

Meta-RL for VNE. We treat VNRs of different sizes as distinct tasks and formulate them as multiple MDPs following a distribution $\mathcal{M}_i \sim p(\mathcal{M})$. Note that this distribution of VNR size is bounded and always obviously smaller than the number of physical nodes, following the network service orchestration standards [Zhuang *et al.*, 2020]. We adopt model-agnostic meta-learning (MAML) as the basic training method [Finn *et al.*, 2017]. MAML facilitates the learning of a meta-policy that can be swiftly fine-tuned on new tasks with only a few training samples, which improves generalizability and adaptability. This training process comprises two stages as follows. Firstly, during the meta-learning process, we iteratively execute the inner loops and outer loops to derive a well-trained meta-policy π_ϕ with cross-task knowledge. Secondly, in the fine-tuning process, we leverage task-specific experiences to fine-tune the meta-policy to a set of size-specific policies π_{θ_i} solely through inner loops.

Concretely, in the inner loop, the meta-policy π_ϕ is updated to accommodate a specific task $\mathcal{M}_i$ by performing gradient descents with the learning rate α and task-specific data $\mathcal{D}_i$:

$$\theta_i = f(\phi, \mathcal{D}_i) = \phi - \alpha \nabla_\phi \mathcal{L}_{\mathcal{D}_i}(\phi). \tag{9}$$

Here, $\mathcal{L}(\cdot)$ follows the objective of proximal policy optimization (PPO) algorithm [Schulman *et al.*, 2017]:

$$\mathcal{L}_{\mathcal{D}_i}(\phi) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}_i}\left[\min\left(r_\phi \hat{A}, \text{clip}\left(r_\phi, \epsilon\right)\hat{A}\right)\right], \quad (10)$$

where $\hat{A}$ denotes the estimated advantage of taking an action. $r_\phi = \frac{\pi_\phi(a_t|s_t)}{\pi_{\phi_{\text{old}}}(a_t|s_t)}$ denotes the ratio between the current policy π_ϕ and the last updated policy $\pi_{\phi_{\text{old}}}$. The clip function with a hyperparameter ϵ is used to limit r_ϕ within the range of $[1-\epsilon, 1+\epsilon]$, improving the stability of policy updates. In PPO, the critic uses a GNN-based encoder and GMPs, then inputs concatenated virtual and physical graph representations into an MLP-based decoder to estimate value.

In the outer loop, our objective is to find a meta-policy π_ϕ that learns balanced strategy knowledge required by VNRs of different sizes and exhibits superior generalizability, enabling it to quickly learn optimal task-specific policies:

$$J_\phi = \mathbb{E}_{\mathcal{M}_i\sim p(\mathcal{M})}\left[\mathbb{E}\left[\sum_{t=0}^{T}\lambda^t R(s_t,a_t)|\theta_i,\mathcal{D}_i\right]\right]. \quad (11)$$

We update ϕ with a meta-learning rate β according to average second-order meta-gradient over task-specific policies:

$$\phi \leftarrow \phi - \beta\nabla_\phi\left(\frac{1}{|\mathcal{M}|}\sum_{i=1}^{|\mathcal{M}|}\mathcal{L}(\theta_i)\right). \quad (12)$$

Curriculum Scheduling Strategy. In our preliminary study (see Appendix A.2), we observed that training specific policies for large VNRs often leads to suboptimal convergence. This issue stems from the complexity of large-sized VNRs and the challenges of exploring the solution space to find feasible solutions. This tendency also towards local optima adversely impacts the meta-learning process. Specifically, using large-sized VNRs in the initial stages of meta-learning results in low-quality gradients, which negatively affects the convergence and generalizability of the meta-policy.

To address this challenge, we draw inspiration from curriculum learning [Wang *et al.*, 2021d] and propose a curriculum scheduling strategy to gradually integrate larger VNRs into the meta-learning process. This strategy enables high-quality initializations for sub-policies of large-sized VNRs, alleviating the problems of suboptimal convergence and compromising meta-policy. We implement this by maintaining a training task list $\mathcal{I}$, initially containing the smallest VNR size. The meta-learning process begins by focusing on tasks with smaller VNR sizes, which are inherently easier and provide beneficial foundational knowledge for tackling more complex tasks. Policies adeptly trained on these smaller tasks serve as effective initializations for larger VNR tasks, facilitating to mitigating local optima issues.

To achieve a gradual increase in task complexity, we use the entropy metric $H(\pi)$ to evaluate the stability of policy. For our bilevel policy, we approximate it with $H(\pi) = H(\pi^H) + H(\pi^L)$. A lower entropy suggests that the policy is making more confident decisions. When the policy entropy $H(\pi_{\theta_k})$ for the largest size $k = \max(\mathcal{I})$ currently on the training task list falls below a specified threshold δ, we consider the policy ready to handle more complex tasks. At this point, we introduce the next larger VNR size to the training task list $\mathcal{I}$. This progressive approach allows the meta-policy to adapt and generalize effectively to larger VNRs.

5 Performance Evaluation

In this section, we evaluate the effectiveness of FlagVNE.

5.1 Experiment Setup

Simulations. Following the latest works [He *et al.*, 2023b; Wang *et al.*, 2023b], we conduct experiments on the simulation platform to mimic various realistic network systems. We adopt two topologies, GEANT (40 nodes and 61 links) and WX100 (100 nodes and 500 links) [Waxman, 1988], as physical networks. See Appendix E.1 for these topologies' descriptions. The multiple-type resources (i.e., CPU, storage, GPU) of physical nodes and bandwidth resources of physical links are uniformly generated within the range of [50, 100] units. In each simulation run, we randomly generate 1000 VNRs with varying sizes ranging from 2 to 10. The virtual nodes within each VNR are randomly interconnected with a probability of 50%. Additionally, resource demands of each VNR's node and link requirements are uniformly generated within the range of [0, 20] and [0, 50] units, respectively. The lifetime of each VNR is exponentially distributed with an average of 500 time units. The arrival of these VNRs follows a Poisson process with an average rate η, wherein η VNRs are received per unit of time. In subsequent experiments, we first train models with $\eta = 0.001$ on GEANT and $\eta = 0.08$ on WX100, due to their different capacities of physical resources. Then we manipulate the value of η to emulate network systems with different traffic throughputs and infer with trained models to study the sensitivity of algorithms.

Implementations. During training, we first conduct meta-learning in the initial 20 simulations and then focus on fine-tuning in the subsequent 10 simulations. We set the policy entropy threshold δ to 2. We implement neural network models with PyTorch and decide reasonable values for hyperparameters following the guide of related studies [Huang *et al.*, 2022; Zhou *et al.*, 2023; Wang *et al.*, 2021a; He *et al.*, 2023a; Kingma and Ba, 2014; Joshi *et al.*, 2022]. See Appendix E.2 for hyperparameter settings on neural networks and meta-RL.

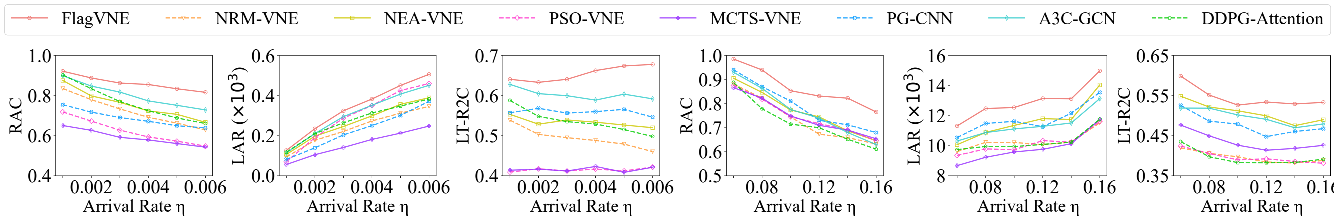
Baselines. To validate the effectiveness of FlagVNE, we compare it with the following SOTA heuristics (NRM-VNE [Zhang *et al.*, 2018]; NEA-VNE [Fan *et al.*, 2023]; PSO-VNE [Jiang and Zhang, 2021]) and RL-based baselines (MCTS-VNE [Haeri and Trajković, 2017]; PG-CNN [Zhang *et al.*, 2022]; A3C-GCN [Zhang *et al.*, 2023]; DDPG-Attention [He *et al.*, 2023b]). See Appendix E.3 for their descriptions.

Metrics. The following metrics are widely used to evaluate the long-term operational status of network systems over a period $\mathcal{T}$ [Fischer *et al.*, 2013]: *request acceptance rate* (RAC), *long-term average revenue* (LAR) and *long-term revenue-to-cost* (LT-R2C). See Appendix E.3 for their definitions.

5.2 Results and Analysis

Overall Performance. To simulate diverse and complex scenarios with varying traffic throughputs, we manipulate the arrival rate of VNRs in two settings due to the difference in physical resource capacity: in GEANT, we explore a range of [0.001, 0.006] with a step of 0.001, and in WX100, we investigate a range of [0.08, 0.18] stepped by 0.02.

Fig. 3(a)(b)(c) and (d)(e)(f) illustrate the performance of all algorithms in GEANT and WX100, respectively. As the



(a) RAC on GEANT (b) LAR on GEANT (c) LT-R2C on GEANT (d) RAC on WX100 (e) LAR on WX100 (f) LT-R2C on WX100

Figure 3: Experimental results in traffic throughput test.

	GEANT			WX100		
	RAC ↑	LAR ↑	LT-R2C ↑	RAC ↑	LAR ↑	LT-R2C ↑
FlagVNE-UniActionNEA	0.781	475.335	0.637	0.724	14334.671	0.493
FlagVNE-MetaFree-SinglePolicy	0.758	472.455	0.614	0.712	14170.514	0.501
FlagVNE-MetaFree-MultiPolicy	0.746	435.502	0.593	0.685	14069.938	0.472
FlagVNE-MetaPolicy	0.773	478.646	0.634	0.717	14292.962	0.485
FlagVNE-NoCurriculum	0.787	485.267	0.643	0.708	14144.234	0.509
FlagVNE	**0.804**	**499.303**	**0.668**	**0.754**	**14769.080**	**0.526**

Table 1: Results on ablation study. ($\eta = 0.006$ on GEANT and $\eta = 0.18$ on WX100).

arrival rate η increases, all algorithms experience a decline in RAC on both topologies, attributed to heightened competition for limited physical resources among VNRs. Despite the variability in algorithm performance across different network topologies, influenced by the varying abundance of physical bandwidth resources, FlagVNE consistently achieves the best performance in all scenarios. We observe that the improvements of FlagVNE are more pronounced at higher values of η, corresponding to heightened resource competition. This underscores the importance of searchability and generalizability in network environments with limited resources. Specifically, at $\eta = 0.006$ on GEANT, FlagVNE surpasses A3C-GCN, NEA-VNE and NRM-VNE by margins of 10.4%, 20.7% and 27.9% on RAC, 10.5%, 28.1%, and 44.2% on LAR, and 12.8%, 28.4%, and 45.1% on LT-R2C. On WX100, compared to A3C-GCN, NEA-VNE and NRM-VNE, FlagVNE shows average improvements over different η of 12.4%, 12.5% and 17.4% in RAC, 12.8%, 10.4% and 24.3% on LAR, and 9.1%, 6.7% and 36.7% on LT-R2C, respectively. Overall, FlagVNE demonstrates exceptional performance across various network system conditions.

Ablation Study. To verify the effectiveness of each proposed component, we build several variations of FlagVNE: (1) *FlagVNE-UniActionNEA* replaces the bidirectional action with the unidirectional one and sorts the decision sequence of virtual nodes with NEA [Fan *et al.*, 2023]. (2) *FlagVNE-MetaFree-SinglePolicy* trains a single general policy with valina PPO, without the help of Meta-RL. (3) *FlagVNE-MetaFree-MultiPolicy* directly trains a set of subpolicies from scratch, without using Meta-RL. (4) *FlagVNE-MetaPolicy* only uses the meta-policy to handle variable-sized VNRs. (5) *FlagVNE-NoCurriculum* discards the curriculum scheduling strategy during the meta-learning process.

We examine their performance under arrival rate settings of $\eta = 0.006$ on GEANT and $\eta = 0.18$ on WX100. These cases exhibit more intense competition for resources, accentuating the performance differentials stemming from the algorithms' searchability and generalizability. As shown in Table 1, FlagVNE outperforms all variations on three metrics, demonstrating that each component of FlagVNE contributes to the improvement in the final performance. Notably, we observe significant performance declines in FlagVNE-MetaFree-MultiPolicy and FlagVNE-MetaFree-SinglePolicy compared to FlagVNE, which shows the effectiveness of our meta-RL training method with a curriculum scheduling strategy in achieving generalization.

Additional Evaluation. Due to the space limit, We place more experiments and analysis in Appendix F, including: the running time test on solving efficiency (F.1), adaptation and convergence analysis in both known and unknown distributions (F.2), scalability validation on large-scale network systems (F.3), and hyperparameter sensitivity study that explores the impact of the key hyperparameter ϵ (F.4).

6 Conclusion

In this paper, we proposed FlagVNE, a flexible and generalizable RL framework for VNE. Specifically, we developed a bidirectional action MDP modeling approach to enable the joint selection of virtual nodes and physical nodes, which expands the agent's search space. Additionally, we designed a hierarchical recorder with a bilevel policy to ensure adaptive output and high training efficiency. Furthermore, we presented a generalizable training method based on meta-RL that efficiently trains a set of size-specific policies to tackle VNRs of varying scales. We also developed a curriculum scheduling strategy that gradually incorporates larger VNRs, thus alleviating suboptimal convergence. Finally, we conducted extensive experiments to verify the effectiveness of FlagVNE.

Acknowledgements

This work was partially supported by National Natural Science Foundation of China (Grant No.92370204), Guangzhou-HKUST(GZ) Joint Funding Program (Grant No.2023A03J0008), Education Bureau of Guangzhou Municipality, Guangdong Science and Technology Department Project funded by China Postdoctoral Science Foundation (Grant No.2023M730785), National Natural Science Foundation of China (Grant No. 62102053).

References

[Blenk *et al.*, 2018] Andreas Blenk, Patrick Kalmbach, Johannes Zerwas, Michael Jarschel, Stefan Schmid, and Wolfgang Kellerer. NeuroViNE: A neural preprocessor for your virtual network embedding algorithm. In *IEEE International Conference on Computer Communications*, pages 405–413, 2018.

[Chen *et al.*, 2020] Liyi Chen, Zhi Li, Yijun Wang, Tong Xu, Zhefeng Wang, and Enhong Chen. Mmea: Entity alignment for multi-modal knowledge graph. In *International Conference on Knowledge Science, Engineering and Management*, pages 134–147, 2020.

[Chen *et al.*, 2022a] Jin Chen, Guanyu Ye, Yan Zhao, Shuncheng Liu, Liwei Deng, Xu Chen, Rui Zhou, and Kai Zheng. Efficient join order selection learning with graph-based representation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 97–107, 2022.

[Chen *et al.*, 2022b] Liyi Chen, Zhi Li, Weidong He, Gong Cheng, Tong Xu, Nicholas Jing Yuan, and Enhong Chen. Entity summarization via exploiting description complementarity and salience. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[Chen *et al.*, 2022c] Liyi Chen, Zhi Li, Tong Xu, Han Wu, Zhefeng Wang, Nicholas Jing Yuan, and Enhong Chen. Multi-modal siamese network for entity alignment. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022.

[Dehury and Sahoo, 2019] Chinmaya Kumar Dehury and Prasan Kumar Sahoo. DYVINE: Fitness-based dynamic virtual network embedding in cloud computing. *IEEE Journal on Selected Areas in Communications*, 37(5):1029–1045, 2019.

[Fan *et al.*, 2023] Weibei Fan, Fu Xiao, Mengjie Lv, Lei Han, Junchang Wang, and Xin He. Node essentiality assessment and distributed collaborative virtual network embedding in datacenters. *IEEE Transactions on Parallel and Distributed Systems*, 34(4):1265–1280, 2023.

[Finn *et al.*, 2017] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017.

[Fischer *et al.*, 2013] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, Hermann de Meer, and Xavier Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys Tutorials*, 15(4):1888–1906, 2013.

[Geng *et al.*, 2023] Haoyu Geng, Runzhong Wang, Fei Wu, and Junchi Yan. Gal-vne: Solving the vne problem with global reinforcement learning and local one-shot neural prediction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 531–543, 2023.

[Haeri and Trajković, 2017] Soroush Haeri and Ljiljana Trajković. Virtual network embedding via Monte Carlo tree search. *IEEE Transactions on Cybernetics*, 48(2):510–521, 2017.

[He *et al.*, 2023a] Huiguo He, Tianfu Wang, Huan Yang, Jianlong Fu, Nicholas Jing Yuan, Jian Yin, Hongyang Chao, and Qi Zhang. Learning profitable nft image diffusions via multiple visual-policy guided reinforcement learning. In *Proceedings of the 31st ACM International Conference on Multimedia*, page 6831–6840, 2023.

[He *et al.*, 2023b] Nan He, Song Yang, Fan Li, Stojan Trajanovski, Liehuang Zhu, Yu Wang, and Xiaoming Fu. Leveraging deep reinforcement learning with attention mechanism for virtual network function placement and routing. *IEEE Transactions on Parallel and Distributed Systems*, 34(4):1186–1201, 2023.

[Huang *et al.*, 2022] Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022.

[Jiang and Zhang, 2021] Chunxiao Jiang and Peiying Zhang. *VNE-HPSO Virtual Network Embedding Algorithm Based on Hybrid Particle Swarm Optimization*, pages 129–152. Springer Singapore, Singapore, 2021.

[Jin *et al.*, 2020] Panpan Jin, Xincai Fei, Qixia Zhang, Fangming Liu, and Bo Li. Latency-aware VNF chain deployment with efficient resource reuse at network edge. In *IEEE International Conference on Computer Communications*, pages 267–276, 2020.

[Joshi *et al.*, 2022] Chaitanya K Joshi, Quentin Cappart, Louis-Martin Rousseau, and Thomas Laurent. Learning the travelling salesperson problem requires rethinking generalization. *Constraints*, 27(1-2):70–98, 2022.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[Liu *et al.*, 2023] Shuncheng Liu, Yuyang Xia, Xu Chen, Jiandong Xie, Han Su, and Kai Zheng. Impact-aware maneuver decision with enhanced perception for autonomous vehicle. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 3255–3268, 2023.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In

International Conference on Machine Learning, pages 1928–1937, 2016.

[Rost and Schmid, 2020] Matthias Rost and Stefan Schmid. On the hardness and inapproximability of virtual network embeddings. *IEEE/ACM Transactions on Networking*, 28(2):791–803, 2020.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Shahriar *et al.*, 2018] Nashid Shahriar, Shihabur Rahman Chowdhury, Reaz Ahmed, Aimal Khan, Siavash Fathi, Raouf Boutaba, Jeebak Mitra, and Liu Liu. Virtual network survivability through joint spare capacity allocation and embedding. *IEEE Journal on Selected Areas in Communications*, 36(3):502–518, 2018.

[Su *et al.*, 2014] Sen Su, Zhongbao Zhang, Alex X. Liu, Xiang Cheng, Yiwen Wang, and Xinchao Zhao. Energy-aware virtual network embedding. *IEEE/ACM Transactions on Networking*, 22(5):1607–1620, 2014.

[Sutton and Barto, 2018] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[Wang *et al.*, 2021a] Chao Wang, Hengshu Zhu, Qiming Hao, Keli Xiao, and Hui Xiong. Variable interval time sequence modeling for career trajectory prediction: Deep collaborative perspective. In *Proceedings of the Web Conference 2021*, pages 612–623, 2021.

[Wang *et al.*, 2021b] Chao Wang, Hengshu Zhu, Peng Wang, Chen Zhu, Xi Zhang, Enhong Chen, and Hui Xiong. Personalized and explainable employee training course recommendations: A bayesian variational approach. *ACM Transactions on Information Systems*, 40(4):1–32, 2021.

[Wang *et al.*, 2021c] Tianfu Wang, Qilin Fan, Xiuhua Li, Xu Zhang, Qingyu Xiong, Shu Fu, and Min Gao. DRL-SFCP: Adaptive service function chains placement with deep reinforcement learning. In *IEEE International Conference on Communications*, pages 1–6, 2021.

[Wang *et al.*, 2021d] Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):4555–4576, 2021.

[Wang *et al.*, 2023a] Chao Wang, Hengshu Zhu, Chen Zhu, Chuan Qin, Enhong Chen, and Hui Xiong. Setrank: A setwise bayesian approach for collaborative ranking in recommender system. *ACM Transactions on Information Systems*, 42(2):1–32, 2023.

[Wang *et al.*, 2023b] Tianfu Wang, Li Shen, Qilin Fan, Tong Xu, Tongliang Liu, and Hui Xiong. Joint admission control and resource allocation of virtual network embedding via hierarchical deep reinforcement learning. *IEEE Transactions on Services Computing*, 2023.

[Waxman, 1988] B.M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, 1988.

[Xiao *et al.*, 2019] Yikai Xiao, Qixia Zhang, Fangming Liu, Jia Wang, Miao Zhao, Zhongxing Zhang, and Jiaxing Zhang. Nfvdeep: Adaptive online service function chain deployment with deep reinforcement learning. In *IEEE/ACM International Symposium on Quality of Service*, 2019.

[Yan *et al.*, 2020] Zhongxia Yan, Jingguo Ge, Yulei Wu, Liangxiong Li, and Tong Li. Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks. *IEEE Journal on Selected Areas in Communications*, 38(6):1040–1057, 2020.

[Yang *et al.*, 2021] Song Yang, Fan Li, Stojan Trajanovski, Ramin Yahyapour, and Xiaoming Fu. Recent advances of resource allocation in network function virtualization. *IEEE Transactions on Parallel and Distributed Systems*, 32(2):295–314, 2021.

[Yang *et al.*, 2022a] Long Yang, Jiaming Ji, Juntao Dai, Linrui Zhang, Binbin Zhou, Pengfei Li, Yaodong Yang, and Gang Pan. Constrained update projection approach to safe policy optimization. *Advances in Neural Information Processing Systems*, 35:9111–9124, 2022.

[Yang *et al.*, 2022b] Long Yang, Yu Zhang, Gang Zheng, Qian Zheng, Pengfei Li, Jianhang Huang, and Gang Pan. Policy optimization with stochastic mirror descent. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8823–8831, 2022.

[Yao *et al.*, 2020] Haipeng Yao, Sihan Ma, Jingjing Wang, Peiying Zhang, Chunxiao Jiang, and Song Guo. A continuous-decision virtual network embedding scheme relying on reinforcement learning. *IEEE Transactions on Network and Service Management*, 17(2):864–875, 2020.

[Zhang *et al.*, 2018] Peiying Zhang, Haipeng Yao, and Yunjie Liu. Virtual network embedding based on computing, network, and storage resource constraints. *IEEE Internet of Things Journal*, 5(5):3298–3304, 2018.

[Zhang *et al.*, 2022] Peiying Zhang, Yu Su, Jingjing Wang, Chunxiao Jiang, Ching-Hsien Hsu, and Shigen Shen. Reinforcement learning assisted bandwidth aware virtual network resource allocation. *IEEE Transactions on Network and Service Management*, 19(4):4111–4123, 2022.

[Zhang *et al.*, 2023] Xiaoquan Zhang, Lin Cui, Fung Po Tso, Zhetao Li, and Weijia Jia. Dapper: Deploying service function chains in the programmable data plane via deep reinforcement learning. *IEEE Transactions on Services Computing*, pages 1–14, 2023.

[Zhou *et al.*, 2023] Jianan Zhou, Yaoxin Wu, Wen Song, Zhiguang Cao, and Jie Zhang. Towards omni-generalizable neural methods for vehicle routing problems. In *the 40th International Conference on Machine Learning*, 2023.

[Zhuang *et al.*, 2020] Weihua Zhuang, Qiang Ye, Feng Lyu, Nan Cheng, and Ju Ren. SDN/NFV-empowered future IoV with enhanced communication, computing, and caching. *Proceedings of the IEEE*, 108(2):274–291, 2020.