Energy-Efficient VNF Deployment for Graph-Structured SFC Based on Graph Neural Network and Constrained Deep Reinforcement Learning

Siyu Qi, Shuopeng Li, Shaofu Lin, Mohand Yazid Saidi, Ken Chen

Energy-Efficient VNF Deployment for Graph-Structured SFC Based on Graph Neural Network and Constrained Deep Reinforcement Learning

Siyu QI[1], Shuopeng LI[1,*], Shaofu LIN[1], Mohand Yazid SAIDI[2], Ken CHEN[2]
[1]*Faculty of Information Technology, Beijing University of Technology, Beijing, China*
[2]*L2TI, Institut Galilee, University Sorbonne Paris Nord, Villetaneuse, France*
[*]*Corresponding author: lishuopeng@bjut.edu.cn*

***Abstract*—Network Function Virtualization (NFV), which decouples network functions from hardware and transforms them into hardware-independent Virtual Network Functions (VNF), is a crucial technology for many emerging networking domains, such as 5G, edge computing and data-center network. Service Function Chaining (SFC) is the ordered set of VNFs. The VNF deployment problem is to find the optimal deployment strategy VNFs in SFC while guaranteeing the Service-Level Agreements (SLA). Existing VNF deployment researches mainly focus on sequences of VNFs without energy consideration. However, with the rapid development of user and application requirement, the SFCs evolve from sequence to dynamic graph and the service providers become more and more sensitive to the energy consumption in NFV. Therefore, in this paper, we identify the Energy-efficient Graph-structured SFC problem (EG-SFC) and formulate it as a Combinatorial Optimization Problem (COP). Benefiting from the recent advances in machine learning for COP, we propose an end-to-end Graph Neural Network (GNN) based on constrained Deep Reinforcement Learning (DRL) method to solve EG-SFC. Our method leverages the Graph Convolutional Network (GCN) to represent the Q-network of Double Deep Q-Network (DDQN) in DRL. The mask mechanism is proposed to deal with the resources constraints in COP. The experimental results show that the proposed method can deal with unseen SFC graphs and achieves better performances than greedy algorithm and traditional DDQN.**

Keywords—Network Function Virtualization, Service Function Chaining, Virtual Network Function, Graph Neural Network, Deep Reinforcement Learning

I. INTRODUCTION

Network Function Virtualization (NFV) is a network architecture proposed by European Telecommunications Standards Institute (ETSI)[1]. NFV decouples network functions from hardware and transforms them into Virtual Network Functions (VNF) that do not rely on dedicated hardware. VNF can be deployed on general-purpose hardware. When providing network services to users , network flows will pass through multiple VNFs in a specific order. An ordered set of VNFs that can describe the logical connection between each other is defined as Service Function Chaining (SFC). Compared to the traditional dedicated equipments, NFV facilitates business expansion and reduces the investment cost of operators.

The deployment of VNF requires the allocation of some resources which should be efficiently chosen. Actually, the VNF deployment problem is a network resource optimization problem that aims to allocate limited CPU, memory, bandwidth, etc., for different users or task to verify the requirements. The VNF deployment problem can be thus modeled as a Combined Optimization Problem (COP) which is a kind of optimization problem that looks for an optimum solution in a discrete domain. With the continuous expansion of the scale of the problems in practical applications and the increasing requirement for real-time solution, the traditional operation researches optimization algorithms cannot realize the online solution of COP. Pointer Network [2] is the first neural Network that can effectively solve COP. Various Pointer Network based methods, showing the powerful effects in solving COP, were then proposed. At the same time, Deep Reinforcement Learning (DRL) is often used to train models due to the lack of labeled data for COP in most cases [3].

DRL has also been applied to solve the VNF deployment problem. These researches generally only focus on the traditional chained SFC, although with the development of network services, SFC will present complex graph structure in many cases. At present, there have been researches focusing on graph-structured SFC. Traditional Machine Learning have poor performance when processing graph structure data. In response to this problem, we propose in this paper to use Graph Neural Network (GNN) to solve the VNF deployment problem. GNN can extract the topological information and complex features from the graph structure quickly. Although GNN has been applied to the deployment of VNF, the related researches usually use it to deal with the physical network topology, without considering the topological information of SFC.

In this paper, we model the EG-SFC problem and propose an end-to-end GNN based on constrained DRL method. Our main contributions are as follows:

(1) the EG-SFC problem is modeled as a COP with objective of joint energy and delay consideration; (2) Graph Convolutional Network (GCN) is adopted to extract the graph-structured data of SFC input and to represent the Q network in Double Deep Q-Network (DDQN); (3) the mask mechanism is used in GCN-based DDQN to satisfy the resources constraints while selecting output nodes.

To validate our proposal, the experiments compare the accepted ratio, end-to-end delay and energy consumptions of random SFC graph requests obtained by our method against traditional DDQN and greedy methods. The numerical results show that our GCN-based method learn better strategy than DNN-DDQN for SFC graphs that have never seen before. Our method also performs better than the greedy method.

II. RELATED WORK

The VNF deployment problem allocates resources such as CPU, memory and bandwidth, so it can be regarded as COP.

Since the first Machine Learning model Pointer Network that can effectively solve COP was proposed, a large number of related methods have been proposed, and these methods are usually trained by Reinforcement Learning (RL). Bello et al.[4] used RL algorithm REINFORCE to train the Pointer Network, and introduced the Critic network as the baseline to reduce the training variance. Its effect exceeds the Pointer Network that used supervised learning. Deudon et al.[5] used the Transformer to improve the Pointer Network. The model still used REINFORCE to update, but its encoder used the Multi-head Attention to calculate the feature vector of the node; its decoder linearly mapped the most recent three-step decision to obtain the reference vector. In the VNF deployment problem, Li et al.[6] modeled the VNF deployment problem as a mixed integer programming problem, and trained the model with RL algorithm Deep Q-Network (DQN) to realize the online deployment of the VNF. Solozabal et al[7] formalized the VNF deployment problem as a constrained COP. Considering the state of the NFV infrastructure, they used the Seq2Seq and DRL to generate the deployment strategy of SFC with the minimum total power consumption in the physical network.

GNN can extract node features, and it is also used to solve COP. Dai et al.[8] first used GNN to solve COP. Ma et al.[9] used GNN to calculate graph embedding, and then used the Attention mechanism to construct the solution, the model achieved good optimization performance in Traveling Salesman Problem (TSP) and other problems. GNN can provide solutions to VNF deployment problem. Kim et al.[10] used the Edge-conditioned Filtered Graph Convolutional Neural Network to generate the state embedding of nodes. The state embedding was connected with the service list to generate the approximate optimal solution of VNF policy classes (add, remove, none). Habibi et al.[11] proposed a method for Variational Graph Autoencoder to accelerate virtual network embedding. The model used the adjacency matrix and the resources feature matrix of physical network to cluster physical nodes, and embedded virtual network by selecting servers in each cluster. Heo et al.[12] built an Encoder-Decoder structure combined with RL to generate the path connecting VNF instances. The encoder produced a vector representation of nodes with the Gated Graph Neural Network (GG-NN). Sun et al.[13] used Graph Network (GN) to extract the nodes and links resources of the network topology, and updated the model with RL to find the VNF deployment strategy with the lowest deployment cost. Rkhami et al.[14] used GCN to encode the graph structure representing the physical network and SFC respectively, and transformed the vector representation of physical network and SFC into state-level coding with Neural Tensor Networks (NTN). The model outputed the prediction corresponding to the strategy and value function.

Regardless of the method adopted, most of the current researches are geared towards the deployment of traditional chained SFC, and they focus more on to network topology rather than SFC. For this situation, we use GNN and DRL to carry out research on the deployment of graph-structured SFC.

III. EG-SFC Problem Formulation

A. Physical Network

We represent the physical network as an undirected graph $G^p=(N^p, L^p)$, where N^p represents a set of physical nodes (i.e. servers) and L^p represents a set of physical links. Server $n \in N^p$ has available computing resources r_n^p and its delay for processing a VNF instance is d_n^p. Each server can be deployed with multiple VNF instances. φ_n is a binary variable, $\varphi_n=1$ means that server n is working, and $\varphi_n=0$ means that server n is off. The bandwidth of the physical link nm, which connects adjacent servers n and m is b_{nm}^p. The transmission delay of service request on nm is d_{nm}^p.

B. Service Function Chaining

An SFC is represented by a directed graph $G^v=(N^v, L^v)$, where N^v represents a set of VNF and L^v represents a set of virtual links. The i-th VNF on SFC is denoted as N_i^v, and its required computing resources is r_i^v. When the i-th VNF is deployed to server n, the binary variable $\theta_i^n=1$, otherwise $\theta_i^n=0$. When the virtual link connecting VNF i and j is mapped to the physical link nm, the binary variable $\lambda_{ij}^{nm}=1$, otherwise $\lambda_{ij}^{nm}=0$. The bandwidth required by SFC $s \in G^v$ is b_s^v.

C. Virtual Network Function Deployment

When the SFC request arrives on the network, the required VNF modules need to be instantiated on the specified servers. The resources of the physical network, which include CPU, bandwidth, etc., are limited. When the amount of resources demanded by a request exceeds the amount of available resources on servers and links, the Service-Level Agreements (SAL) cannot be guaranteed. Therefore, the VNF deployment should meet the constraints of resources. The objective of VNF deployment problem is to find the optimal VNF deployment strategy to reduce network Operating Expenditure (OPEX) while guaranteeing service requirements. Fig. 1 is an example of the VNF deployment of a graph-structured SFC. The black numbers in Fig. 1 represent the computing resources of the server or the bandwidth of the physical link, and the red numbers represent the processing delay of the server or the transmission delay of the physical link. When the SFC request arrives, the VNFs of the SFC should be deployed. Combined with resources constraints and service requirements, VNF1, VNF2 and VNF3 are mapped to Server2, VNF4 and VNF5 are mapped to Server5. The virtual links are also mapped.

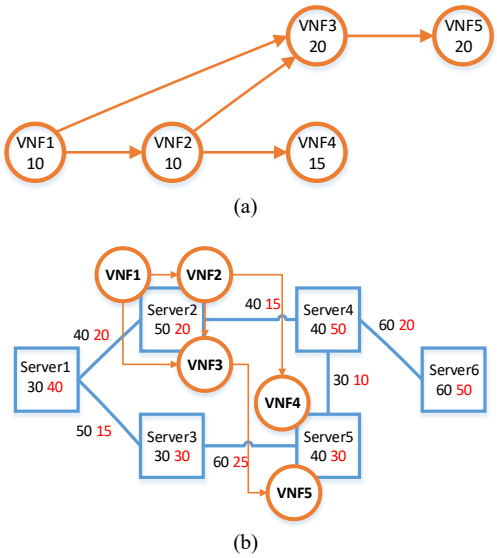

Fig. 1. VNF deployment. (a) is a SFC; (b) is the result of VNF deployment.

D. Optimization Problem

The objective of this paper is to minimize the energy consumption and end-to-end delay of SFC request while guaranteeing the success of VNFs deployment. We define this optimization problem as EG-SFC.

For simplicity, we assumed that the energy consumption of SFC is determined by the number of working servers in the physical network. Hence, at timestep t, we define the energy consumption of SFC as follows:

$$c(t)=\sum_{n\in N^p}\varphi_n(t) \tag{1}$$

The end-to-end delay of SFC mainly consists of processing delay and transmission delay. At timestep t, the processing delay and transmission delay of SFC are defined as:

$$d_p(t)=\sum_{i\in N^v}\sum_{n\in N^p}\theta_i^n(t)d_n^p \tag{2}$$

$$d_t(t)=\sum_{i,j\in N^v n,m\in N^p}\lambda_{ij}^{mn}(t)d_{nm}^p \tag{3}$$

where d_n^p and d_{nm}^p are normalized for a uniform order of magnitude. According to the energy consumption and end-to-end delay of SFC, we define the optimization objective $O(t)$ as:

$$O(t)=e_1\cdot\frac{c(t)}{\mathrm{num}(N^p)}+e_2\cdot\left(\frac{d_p(t)}{\mathrm{num}(N^v)}+\frac{d_t(\mathrm{t})}{\mathrm{num}(L^v)}\right) \tag{4}$$

Our objective objective consider jointly the energy consumption and delay. e_1 and e_2 are tradeoff parameters, and $e_1+e_2=1$. We also normalized the energy consumption.

We take the computing resources of physical nodes and the bandwidth of physical links as constraints. Combined with the optimization objective, the optimization problem is defined as:

$$\min O(t) \tag{5}$$

$$\sum_{n\in N^p}\theta_i^n(t)=1,\ \forall\ i\in N^v \tag{6}$$

$$\sum_{i\in N^v}\theta_i^n(t)r_i^v\le r_n^p,\ \forall\ n\in N^p \tag{7}$$

$$\sum_{i\in N^v}(\lambda_{ij}^{nm}(t)+\lambda_{ij}^{mn}(t))b_{ij}^v\le b_{nm}^p,\ \forall\ nm\in L^p \tag{8}$$

$$\sum_{i\in N^v}\lambda_{ij}^{nm}(t)-\sum_{i\in N^v}\lambda_{ij}^{mn}(t)=\theta_i^n(t)-\theta_j^m(t),\forall\ n,m\in N^p,\forall\ nm\in L^p \tag{9}$$

$$\theta_i^n(t)=\{0,1\},\ \forall\ i\in N^v,\ \forall\ n\in N^p \tag{10}$$

$$\lambda_{ij}^{nm}(t)=\{0,1\},\ \forall\ i,j\in N^v,\ \forall\ n,m\in N^p \tag{11}$$

$$\varphi_n(t)=\{0,1\},\ \forall\ n\in N^p \tag{12}$$

Equation(6)-Equation(12) guarantee the effectiveness of the optimization objective. Equation(6) is used to guarantee that the VNF on the SFC can only select one server for deployment. Equation(7) is used to guarantee that the sum of resource required by the VNF deployed on a given sever does not exceed the total computing resource of that server. Equation(8) guarantees that the sum of bandwidth requirements of all virtual links mapped to a physical link does not exceed the total bandwidth of that physical link. Equation(9) indicates that when two adjacent VNFs in the SFC are deployed to servers n and server m, there must be a continuous path between the physical links nm. Equation(10)-Equation(12) represent binary variable constraints for VNFs deployment, virtual links mapping and servers state respectively.

IV. System Modelling

EG-SFC is a COP. It is NP-hard since it aims to solve the NP-hard problem consisting in the VNF deployment and the virtual links mapping. Here, we focus on the VNF deployment.

A. Deep Reinforcement Learning Components

We model the VNF deployment problem as a Markov Decision Process (MDP). The state, action and reward of MDP are described below.

1) State: The state is represented by SFC graph $s\in G^v=(N^v, L^v)$. The features of the i-th VNF N_i^v include: (1) required computing resources r_i^v; (2) the flag indicating the deployment status of the VNF in servers, $\theta_i^n,\ \forall\ n\in N^p$; (3) the percentage of available computing resources on the server which the VNF is deployed, $p_i^m,\ \exists\ m\in N^p$; (4) bandwidth required for the SFC b_s^v; (5) the flag indicating whether the VNF has been deployed μ_i^v; (6) the flag ε_i^v indicating whether the VNF N_i^v is currently deployed or not. In the initial state $s(0)$, θ_i^n and μ_i^v of all nodes are all intialized to 0. $\varepsilon_1^v=1$, and this flag of other VNFs are all 0.

2) Action: The action is the index of server which deploy the currently processing VNF. At the timestep t, the agent only selects one server to instantiate the VNF. And a server can host multiple VNF instances.

3) Reward: We define a binary variable η_a to indicate whether the current action starts a new server:

$$\eta_a(t)=\begin{cases}0, & \varphi_a(t\text{-}1)=\varphi_a(t)\\ 1, & \varphi_a(t\text{-}1)\neq\varphi_a(t)\end{cases} \tag{13}$$

If the environment violates the constraints after the action is executed, the deployment of the current VNF is considered as failing. The environment will feedback a larger negative value to the agent as a penalty. Otherwise, the environment will generate a reward based on the changes in the number of working servers and delay. We define the reward function as:

$$\mathrm{r}(t)=\begin{cases}50, & \text{failed}\\ C\left(e_1\eta_a(t)+e_2\left(d(t)\text{-}d(t\text{-}1)\right)\right), & \text{otherwise}\end{cases} \tag{14}$$

where $d(\mathrm{t})=d_p(t)+d_t(t)$; C is a fixed constant.

We use DDQN to train the model. DDQN uses target-Q network and ReplayBuffer to improve performance. TargetQ is calculated using the following formula:

$$a^{\max}(s',\theta)=\arg\max_{a'}Q(s',a';\theta) \tag{15}$$

$$\mathrm{TargetQ}=r+\gamma Q(s',a^{\max}(\mathrm{s}',\theta);\theta^-) \tag{16}$$

where θ is the parameter of the Q network; θ^- is the parameter of the target-Q network; s', a', r are state, action and reward drawn from the ReplayBuffer.

The VNF deployment and the virtual links mapping need to meet resources constraints. In order to accelerate the training, in addition to returning a penalty in the reward function for the action that violates the constraints, we introduce the mask mechanism. When the required computing resources of VNF being deployed is greater than the remaining resources of the server, we set the Q value of the server to a large negative value to make it impossible to be selected:

$$\kappa=\begin{cases}0, r_n^p-\sum_{i\in N^v}\theta_i^n(t)r_i^v\geq r_c^v, \forall\ i\in N^v\\ 1, r_n^p-\sum_{i\in N^v}\theta_i^n(t)r_i^v<r_c^v, \forall\ i\in N^v\end{cases} \quad (17)$$

$$Q_n(s,a)=Q_n(s,a)-\kappa B, \forall n\in N^p \quad (18)$$

where $Q_n(s,a)$ is the Q value of the action representing deploying the current VNF on server n; B is a sufficiently large positive number; r_c^v is the computing resources required by the VNF currently being deployed.

B. Graph Neural Network

We use GCN as the Q network and the target-Q network in DDQN. Compared with ordinary GNN, GCN introduces the convolution operation. The structure of GNN is composed of the stacked form of the local transfer function and the local output function which applied to all nodes. The local transition function generates the state representation of the node, which contains the neighborhood features of the node and indicates the dependence of each node's state on neighbors. It is shared among all nodes and updates the state of the node according to the neighborhood. Its expression is:

$$h_v=f_w(x_v, x_{vu}^e, h_u, x_u) \quad (19)$$

where x_v is the features of node v; x_{vu}^e is the features of link connecting node u and v; h_u is the state representation of the neighbor nodes of node v; x_u is the features of the neighbor nodes of node v.

The local output function generates the final output vector representation of the node, which is expressed as:

$$o_v=g_w(h_v, x_v) \quad (20)$$

GCN introduces the convolution operation into the graph structure. The convolution layer formula is defined as:

$$h^{(l)}=\sigma(\overline{D}^{-\frac{1}{2}}\overline{A}\overline{D}^{-\frac{1}{2}}h^{(l-1)}W^{(l-1)}) \quad (21)$$

where $\sigma(\cdot)$ is the nonlinear activation function; $\overline{A}$ is the adjacency matrix; $\overline{D}$ is the diagonal matrix of $\overline{A}$; $W^{(l-1)}$ is the weight matrix of the (l-1)th layer.

The applications of GNN mainly include node classification, edge classification, link prediction and graph classification. Instead of using node classification to represent the deployment of the VNF, we use the graph classification to generate the deployment strategy of processing VNF based on the topological information of current SFC graph, and the resources requirements and deployment situation of all VNFs. When GNN is used for graph classification, it is necessary to obtain the representation of the graph based on the features of each node. This operation requires aggregating as much information as possible in the graph and is called readout. We use a simple method to aggregate and readout the features of all nodes to obtain the representation of the graph. The formula is as follows:

$$h_g=\frac{1}{|N|}\sum_{v\in N}h_v \quad (22)$$

where h_g is the representation of graph g; N is the set of nodes of g.

C. Learning Process

The learning process of the method is shown in Fig. 2. We use DDQN to train the model, and use GCN as Q network and target-Q network of DDQN. In each episode, we deploy VNFs of an SFC. The SFC graph that contains the resources requirements and deployment status is the input state of DRL. In GCN, each convolutional layer will calculate the graph convolution according to the topological information and nodes features. The graph convolution is passed to the next convolutional layer after the ReLu activation function. Through multiple convolution layers and the output layer, GCN generates the node representation. We readout the representations of all nodes as the Q value to generate the action of DRL, that is, the server that the current VNF will deploy. After a VNF is deployed, we use the shortest path algorithm to map the virtual links related to this VNF to physical links. The method deploy one VNF at each step. The episode ends when all VNFs of the SFC are deployed.

V. Evaluation

We conduct three experiments to evaluate our method. Our method deploys one VNF at each step, and the topological relationship between VNFs in the graph-structured SFC is complicated, so we first study the impact of the deployment order of VNFs. On the basis of the first experiment, we study the effectiveness of GCN for SFC with different numbers of VNFs. Finally, we study the rationality and effectiveness of our optimization objective.

A. Data Description

1) Physical network: We use Internet2[10] network topology as the physical network topology. The topology consists of 12 nodes and 15 edges. The available computing resources of servers are randomly selected in [30, 40, 50, 60], and the processing delay is randomly selected in [20, 30, 40, 50]. The bandwidth of physicals links is randomly selected in [50, 60, 70, 80], and the transmission delay is randomly selected in [10, 15, 20, 25].

2) Service Function Chaining: In order to adapt the model to various topologies, we used SFC with different topologies to train the model. We randomly change the topology of the SFC within a fixed number of VNFs. In the process of changing the topology, it is necessary to ensure that the topology is connected and directed acyclic. The required bandwidth of SFC and the required computing resources of VNFs are randomly selected in [10, 15, 20]. The number of VNFs in SFC is adjusted according to the experiment content. For different quantity ranges, we randomly generated 100 SFCs and randomly selected one for training at the beginning of each episode. We used the same strategy to generate another 100 SFCs as test data.

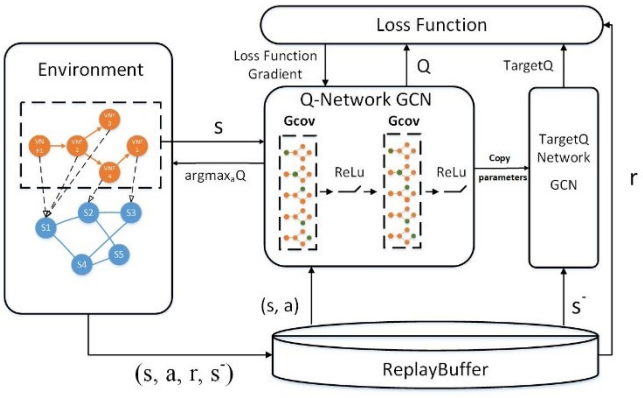

Fig. 2. Learning process of GNN-based VNF deployment method

B. GCN and DRL Parameters

We set the number of GCN layers to 3, and we use ReLu as the activation function. For DDQN, we use the Adam optimizer and set the discount rate to 0.99. The capacity of the ReplayBuffer is 600, and the batch size is 64. We use ϵ-greedy sampling to explore, where ϵ decreases with the number of steps and is finally fixed at 0.05. we set the maximum norm for the gradient clipping to 0.5 to help stabilise training. We synchronize the parameters of Q network and target-Q network every 400 episodes. Each experiment is conducted for 10,000 episodes, and the best model is saved.

C. Baseline model

1) Least Delay Greedy: Least Delay Greedy (LDG) deploye the first VNF on the server with the least processing delay. When the available computing resources that server deployed are greater than the resources required by the current VNF, continue to deploy the VNF on this server, otherwise deploy the VNF on the server with the minimum sum of the processing delay and the transmission delay between the last server where VNF is deployed.

2) DNN-DDQN: DNN-DDQN use Fully Connected Neural Network as Q network and target-Q network. The number of neural network layers is 3, which converts the connection of SFC features and adjacent information into the vector as input. The other parameters of DNN-DDQN are the same as the method we proposed. The mask mechanism is also introduced in DNN-DDQN.

D. Evaluation

1) Impact of deployment order: In this part, we show the impact of the deployment order of VNFs. We deploy VNFs in topological sort order and random order respectively. We set the number of VNFs in one SFC to 7-10. The tradeoff parameters are set to e_1=0.5,e_2=0.5. The results are shown in TABLE I.

In the experiment, compared with the random order, the number of working servers of the strategy generated by topological sort order was 3.67% higher, and the end-to-end delay was 24.78% lower. According to the results, the number of working servers of two strategies are similar, but the strategy generated according to the topological sort order has significantly smaller end-to-end delay. Therefore, we deploy VNFs in the topological sort order in following experiments.

2) Impact of the number of VNFs: In this part, we show the impact of the amount of VNFs in SFC. We set the tradeoff parameters to e_1=0.5,e_2=0.5 and deploy VNFs in topological sort order. We increase the number of VNFs in SFC from 5-8 to 11-14, rising the upper and lower bounds by 2 in each step. The results is shown in Fig. 3.

TABLE I. TEST RESULTS OF VNF DEPLOYMENT ORDER

deployment order	*working servers*	*end-to-end delay*
random	3	517.78
topological sort order	3.11	389.45

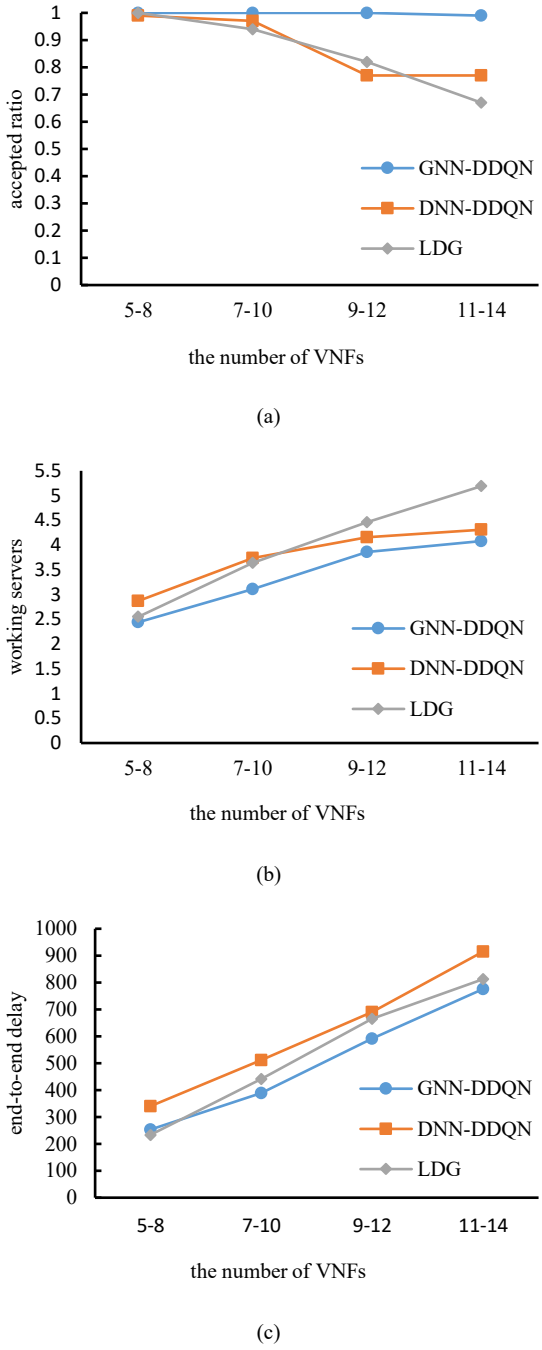

Fig. 3. Test results of the number of VNFs

The results show that the accepted ratio of the strategy generated by our method is the same as LDG when the number of VNFs in SFC is 5-8, and 1% higher than DNN-DDQN. When and the number of VNFs is 7-10, 9-12 and 11-14, the accepted ratio of our deployment strategy is 6%, 18% and 32% higher than LDG, and 3%, 23% and 22% higher than DNN-DDQN. For the number of working servers, when the number of VNFs in SFC is 5-8, 7-10, 9-12 and 11-14, our strategy is 4.31%, 14.56%, 13.45% and 21.39% lower than LDG, and 14.98%, 16.84%, 7.21% and 5.34% lower than DNN-DDQN. For the end-to-end delay, our strategy is 8.43% higher than LDG and 25.56% lower than DNN-DDQN when the number of VNFs in SFC is 5-8. When and the number of VNFs is 7-10, 9-12 and 11-14, the end-to-end delay of our strategy is 11.75%, 11.14% and 4.50% lower than LDG, and 23.92%, 14.29%, 15.17% lower than DNN-DDQN.

Our method is better than the baseline models in terms of accepted ratio, and the advantage becomes more pronounced as the number of VNFs increases. The results show that the number of working servers and end-to-end delay are positively correlated with the number of VNFs in the SFC. As the number of VNFs increases, the advantage of using our method tends to decrease over DNN-DDQN in the number of working servers and over LDG in end-to-end delay. Compared with these two algorithms, the deployment strategy derived from our method can minimize the energy consumption while keeping the minimal end-to-end delay, so our method is energy-efficient.

Combined with the accepted ratio, the number of working servers and the end-to-end delay, the performance of our method is superior to DNN-DDQN using traditional method and LDG based on greedy algorithm. Experiments show that GCN is effective for graph-structured SFC, and it can capture the topological information, which is ignored by traditional methods.

3) Impact of the tradeoff parameters: The tradeoff parameters will affect the VNF deployment strategy generated by our method. To make the influence of the tradeoff parameters more obvious, we randomly select the computing resources and processing delay of servers in [20, 30, 40]. And we set the large available computing resources and processing delay for two of the servers. For the physical links, the transmission delay is randomly selected in [5, 10, 15]. We use SFCs with 7-10 VNFs to test the impact of the tradeoff parameters. We set two extreme tradeoff parameters, e_1=0.95, e_2=0.05 and e_1=0.05,e_2=0.95. The results are shown in TABLE II.

Compared with e_1=0.95 and e_2=0.05, the number of working servers is 86.7% higher when e_1=0.05 and e_2=0.95, and the end-to-end delay is 59.03% lower. When $e_1 > e_2$, the method will generate a strategy with lower energy consumption. Otherwise, the method will pay more attention to the end-to-end delay. When the tradeoff parameter of one indicator is significantly smaller than the other, the change of this indicator has little impact on the optimization objective, and the change of the other will significantly change the optimization objective. The results show that our selected optimization objective is effective. Since, our method can balance the energy consumption and the end-to-end delay according to the needs of different scenarios.

TABLE II. TEST RESULTS OF TRADEOFF PARAMETERS

tradeoff parameters	*working servers*	*end-to-end delay*
e_1=0.05, e_2=0.95	4.07	319.85
e_1=0.95, e_2=0.05	2.18	780.70

VI. CONCLUSION

In this paper, we define the SFC problem with objective of joint energy consumption and delay as a COP. We propose a VNF deployment method based on GNN and DRL. GCN is adopted to extract the information of the graph-structured SFC input and represent the Q network in DDQN. We introduce the mask mechanism to deal with the resources constraints in COP. The experimental results show that our proposal could efficiently deal with unseen SFC graphs without redesigning and training, and achieves better performances than greedy method and traditional DDQN.

ACKNOWLEDGMENT

This research was supported by the National Key Research and Development Program of China (2020YFF0305400) and the International Research Cooperation Seed Fund of Beijing University of Technology (No. 2021B02).

REFERENCES

[1] I. Alam et al., "A survey of network virtualization techniques for internet of things using sdn and nfv," ACM Computing Surveys (CSUR), vol. 53, no. 2, pp. 1-40, 2020.

[2] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," arXiv preprint arXiv:1506.03134, 2015.

[3] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," Computers & Operations Research, p. 105400, 2021.

[4] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," arXiv preprint arXiv:1611.09940, 2016.

[5] M. Deudon, P. Cournut, A. Lacoste, Y. Adulyasak, and L.-M. Rousseau, "Learning heuristics for the tsp by policy gradient," in International conference on the integration of constraint programming, artificial intelligence, and operations research, 2018: Springer, pp. 170-181.

[6] J. Li, W. Shi, N. Zhang, and X. Shen, "Delay-aware VNF scheduling: A reinforcement learning approach with variable action set," IEEE Transactions on Cognitive Communications and Networking, 2020.

[7] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco, and F. Liberal, "Virtual network function placement optimization with deep reinforcement learning," IEEE Journal on Selected Areas in Communications, vol. 38, no. 2, pp. 292-303, 2019.

[8] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," arXiv preprint arXiv:1704.01665, 2017.

[9] Q. Ma, S. Ge, D. He, D. Thaker, and I. Drori, "Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning," arXiv preprint arXiv:1911.04936, 2019.

[10] H.-G. Kim et al., "Graph neural network-based virtual network function management," in 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS), 2020: IEEE, pp. 13-18.

[11] F. Habibi, M. Dolati, A. Khonsari, and M. Ghaderi, "Accelerating Virtual Network Embedding with Graph Neural Networks," in 2020 16th International Conference on Network and Service Management (CNSM), 2020: IEEE, pp. 1-9.

[12] D. Heo, D. Lee, H.-G. Kim, S. Park, and H. Choi, "Reinforcement Learning of Graph Neural Networks for Service Function Chaining," arXiv preprint arXiv:2011.08406, 2020.

[13] P. Sun, J. Lan, J. Li, Z. Guo, and Y. Hu, "Combining Deep Reinforcement Learning With Graph Neural Networks for Optimal VNF Placement," IEEE Communications Letters, 2020.

[14] A. Rkhami, T. A. Q. Pham, Y. Hadjadj-Aoul, A. Outtagarts, and G. Rubino, "On the Use of Graph Neural Networks for Virtual Network Embedding," in 2020 International Symposium on Networks, Computers and Communications (ISNCC), 2020: IEEE, pp. 1-6.