# DRL-SFCP: Adaptive Service Function Chains Placement with Deep Reinforcement Learning

Tianfu Wang[1,2], Qilin Fan[1,2,*], Xiuhua Li[1,2], Xu Zhang[3], Qingyu Xiong[1,2], Shu Fu[1,4] and Min Gao[1,2]
[1]Key Laboratory of Dependable Service Computing in Cyber-Physical-Society, Ministry of Education, China
[2]School of Big Data & Software Engineering, Chongqing University, Chongqing, China
[3] School of Electronic Science and Engineering, Nanjing University, Jiangsu, China
[4] College of Microelectronics and Communication Engineering, Chongqing University, Chongqing, China
[*]Corresponding author: Qilin Fan (fanqilin@cqu.edu.cn)
Email:[1,2]{wangtianfu,fanqilin,xiong03,gaomin}@cqu.edu.cn, lixiuhua1988@gmail.com,
[3]xzhang17@nju.edu.cn, [4]shufu@cqu.edu.cn

***Abstract*—Network function virtualization (NFV) is a promising paradigm that network functions can be deployed on commodity servers instead of dedicated servers to enhance the resource utilization and reduce the management difficulty. Based on the NFV technology, a complex network service can be composed of a series of ordered virtual network functions, known as service function chain (SFC). In this context, how to efficiently place SFCs in acceptable running time to improve resource utilization and service quality while meeting the constraints of the physical network is a critical issue for infrastructure providers. In this paper, we propose a deep reinforcement learning-based approach called DRL-SFCP for adaptive SFC placement. DRL-SFCP maximizes the long-term average revenue by combining both the graph convolution network which extracts the features of the physical network and sequence-to-sequence model which captures the ordered information of the SFC request to generate placement strategies. It learns to make SFC placement decisions via observations of the corresponding performance of past decisions rather than a hypothetical environment. Extensive experimental results show that our DRL-SFCP can achieve 11.6% and 9.6% improvement in terms of the acceptance ratio and the long-term average revenue, compared with existing benchmarks.**

## I. Introduction

Network function virtualization (NFV) is a promising paradigm that network functions are executed with the assistance of software middle-boxes that run on top of commodity services rather than traditional dedicated servers. NFV has received extensive attention from academia and industry for its superior ability to enhance resource utilization and reduce management difficulty. Benefiting from NFV technology, a complex network service (NS) [1] can be composed of a series of ordered virtual network functions (VNFs) [2], known as service function chain (SFC). In an online scenario, continuous SFC requests are attempted to be placed in the underlying network under various resource constraints [3]. It is an issue actively followed by infrastructure providers (InPs) that place SFC requests as efficiently as possible to improve resource utilization and quality of service (QoS).

To solve the SFC placement problem, various approaches have been proposed. We divide them into three areas: i) Mathematical optimization-based. The authors in [4] jointly considered the placement of VNFs and flow distribution of SFCs and proposed a polynomial time algorithm based on linear relaxation and rounding. The authors in [5] modeled SFC placement as a set cover problem to minimize the setup cost and proposed two algorithms with a logarithmic approximation factor. However, due to the dependence of prior knowledge of SFCs and the high computational complexity, these offline approaches are difficult to apply to online placement scenarios; ii) Heuristic-based. The authors in [6] aimed to maximize the revenue-to-cost ratio based on the metric of global resource capacity. A dynamic programming-based algorithm was designed in [7] to minimize the resources cost. Nevertheless, these algorithms lack adaptability to various environments and might fall into local optimum; iii) Reinforcement learning (RL)-based. The authors in [8] formulated the problem as Markov decision process (MDP) and used the Monte Carlo tree search algorithm to maximize the revenue-to-cost and acceptance ratio. The authors in [9] utilized a policy gradient approach with Lagrange relaxation technique to minimize the overall power consumption. However, these approaches only consider the single resource in the physical server and assume the grid-like or start-like topology for the physical network.

In this paper, we propose a DRL-based approach called DRL-SFCP for adaptive SFC placement. DRL-SFCP maximizes the long-term average revenue by combining both the graph convolution network (GCN) which extracts the features of the physical network and sequence-to-sequence (Seq2Seq) model which captures the ordered information of the SFC request to generate placement strategies. It learns to make SFC placement decisions via observations of the corresponding performance of past decisions rather than a hypothetical environment. The main contributions of this paper are summarized as follows:

- We investigate the use of DRL to guide online placement decisions for SFC requests. To this end, we formulate the problem as a MDP, in which the optimal solution can be considered as a sequence of decisions.
- We design a novel DRL-SFCP approach which utilizes deep neural networks (DNNs) to explore the non-Euclidean structural characteristics of the physical net-

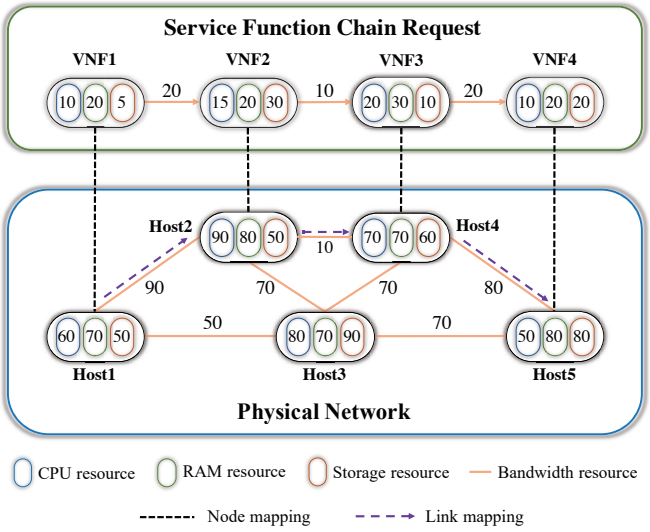


Fig. 1. An example of SFC placement.

work and the ordered information of the SFC request. Furthermore, we employ asynchronous advantage actor-critic (A3C), which maintains a policy and an estimate of the value function to train the DNN of DRL-SFCP in parallel.

- Experimental results show that compared with existing approaches, the proposed DRL-SFCP can improve the acceptance ratio and the long-term average revenue with low running time.

The rest of the paper is organized as follows. In Section II, we provide the system model and formal definition of the problem. The details of DRL-SFCP approach are presented in Section III. Section IV presents the experimental results. Finally, we conclude our work in Section V.

## II. System Model and Problem Formulation

In this section, we present the physical network model and the SFC request model. Besides, we provide the formal definition of SFC placement problem.

### A. Physical Network

The physical network can be modeled as a weighted undirected graph $G' = (N', L')$, where $N'$ denotes the set of nodes and $L'$ represents the set of links in the physical/substrate network. The set of node resources (e.g., central processing units (CPUs), random access memory (RAM), storage, etc.) provided by the physical network is denoted as $K$. The vector of remaining and maximum resources of physical node $n' \in N'$ are denoted as $R^r_{n'} = \left[R^r_{n',1}, \ldots, R^r_{n',k}, \ldots, R^r_{n',|K|}\right]$ and $R^m_{n'} = \left[R^m_{n',1}, \ldots, R^m_{n',k}, \ldots, R^m_{n',|K|}\right]$ respectively, where $R^r_{n',k}$ and $R^m_{n',k}$ are the remaining and maximum amount of resources $k \in K$ provided by $n'$. The link-oriented attribute could be bandwidth, latency, packet loss, etc. Without loss of generality, we only take bandwidth as the link attribute in physical network in this paper. The remaining and maximum bandwidth of physical link $l' \in L'$ are denoted as $B^r_{l'}$ and $B^m_{l'}$.

### B. SFC Requests

Each SFC request $i$ can be modeled as a weighted directed graph $G^i = (N^i, L^i)$, where $N^i$ denotes the set of VNFs and $L^i$ refers to the virtual links. $t^i$ is the arrival time of $i$. We denote $r_{n^i} = [r_{n^i,1}, \ldots, r_{n^i,k}, \ldots, r_{n^i,|K|}]$ as the vector of requested resources of VNF $n^i \in N^i$, where $r_{n^i,k}$ is the amount of resource $k \in K$ requested by VNF $n^i$. $b_{l^i}$ represents the bandwidth demand of the virtual link $l^i \in L^i$.

### C. Problem Formulation

Fig. 1 gives an example of SFC deployment. It tries to map an arriving SFC $G^i$ to the physical network $G'$ under the constraints of resources. We introduce two binary variables $\phi^{n^i}_{n'}$, $\phi^{l^i}_{l'}$ to denote the placement of VNFs and virtual links to the nodes and links in the physical network, respectively. $\phi^{n^i}_{n'}$ ($\phi^{l^i}_{l'}$) is 1 if $n^i$ ($l^i$) is mapped at the physical node (link) $n'$ ($l'$), and 0 otherwise. The problem is subject to the following constraints:

- The total amount of required resource $k$ of SFC request $i$ mapped on $n'$ should not exceed its remaining amount of resources. We have:

$$\sum_{n^i} \phi^{n^i}_{n'} r_{n^i,k} \leq R^r_{n',k}, \forall n' \in N', \forall k \in K. \tag{1}$$

- The total bandwidth of SFC request $i$ mapped on $l'$ should not exceed its remaining bandwidth. We have:

$$\sum_{l^i} \phi^{l^i}_{l'} b_{l^i} \leq B^r_{l'}, \forall l' \in L'. \tag{2}$$

- Each VNF $n^i$ can only be placed on one physical node:

$$\sum_{n'} \phi^{n^i}_{n'} \leq 1, \forall n^i \in N^i. \tag{3}$$

- If SFC request $i$ is accepted, the path mapped in the physical network should traverse through VNFs following the order specified in the request. We denote $I(n')$ and $O(n')$ as the set of incoming and outgoing links of physical node $n'$, $n^i_s$ and $n^i_d$ denote the source and destination VNF of the virtual link $l^i$. We have:

$$\sum_{l' \in I(n')} \phi^{l^i}_{l'} - \sum_{l' \in O(n')} \phi^{l^i}_{l'} = \phi^{n^i_d}_{n'} - \phi^{n^i_s}_{n'}, \forall l^i \in L^i, l' \in L'. \tag{4}$$

In this paper, we assume that the InP charges SFCs by the "pay-as-you-go" pricing model which has been widely used in the cloud platform, the revenue obtained by each SFC request $i$ is defined as:

$$rev(i) = \begin{cases} \mu_k \sum_{n^i} r_{n^i,k} + \eta \sum_{l^i} b_{l^i}, & \text{if } i \text{ is accepted,} \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

where $\mu_k$ is the unit price of resource $k$ in physical nodes, and $\eta$ is the unit price of bandwidth.

We use $R(\pi)$ to denote the long-term average revenue, which is given by:

$$R(\pi) = \lim_{\tau \to \infty} \frac{1}{\tau} \sum_{i \in I_\tau} rev\,(i), \tag{6}$$

where $I_\tau = \{i \mid 0 < t^i < \tau\}$ denotes the SFCs that arrive before time $\tau$.

Our objective is to find a policy that maximizes the long-term average rate:

$$\pi^* = \arg\max_{\pi} R(\pi). \tag{7}$$

## III. DRL-SFCP Approach

### A. MDP Formulation

Under the DRL framework, the interaction between agent and environment is defined as MDP. We cast the placement of each SFC request $i$ as a finite-horizon MDP. The agent consecutively selects consecutively physical nodes to place $\left|N^i\right|$ VNFs until the horizon $T = \left|N^i\right|$ is reached. At each time step $t$ ($t = 1, \ldots, T$), we define three key elements: state, action and reward as follows:

*1) State:* The state should include both the current physical network situation and the requirement of SFC request. Thus, we define the state as $s_t = (s_t^p, s_t^r)$. Here, $s_t^p = (A, X)$ is the current state of physical network, where $A \in \mathbb{R}^{\left|N'\right| \times \left|N'\right|}$ is the adjacent matrix of network and $X \in \mathbb{R}^{\left|N'\right| \times M}$ is the feature matrix of physical nodes: i) Each row of $X$ refers to a $M$-dimensional feature vector of physical node $n'$. In this paper, we concatenate the remaining resources $R^r_{n'}$, maximum resources $R^m_{n'}$, the sum of remaining bandwidth $S^r_{n'} = \sum_{l' \in L(n')} B^r_{l'}$ and maximum bandwidth $S^m_{n'} = \sum_{l' \in L(n')} B^m_{l'}$ of adjacent links $L(n')$ as features of each physical server, and normalize them into $[0, 1]$; ii) $s_t^r$ is the current state of SFC request, which consists of the requested amount of resources $r_{n_t^i}$ of $t$-th VNF, required bandwidth $b_{l_t^i}$ of $t$-th virtual link and the number of VNFs remaining to be placed.

*2) Action:* The action set is defined as: $A_t = \{\overline{\zeta}\} \cup \{n' \in N' | r_{n_t^i,k} \leq R^r_{n',k}, \forall k \in K\}$. An action $a_t$ selects a node $n'$ from candidate physical servers whose available resources exceed requirements requested by VNF $n_t^i$. Otherwise, $a_t = \overline{\zeta}$ forces the transition to the terminal state.

*3) Reward:* The reward signal is designed to encourage the agent to place SFCs with the aim of maximizing the long-term average revenue. To this end, when the SFC request $i$ is accepted at time step $t = T$, the agent receives the reward $r_t = rev_i$, where $rev_i = \mu_k \sum_{n^i} r_{n^i,k} + \eta \sum_{l^i} b_{l^i}$. In the intermediate steps (i.e., $t < T$), the agent receives a small reward $r_t = \xi rev_i$ when resources constraints are satisfied and $r_t = -\xi rev_i$ otherwise, where $\xi$ is the reward coefficient.

### B. Model Architecture

The model architecture of DRL-SFCP, illustrated in Fig. 2, consists of three components: i) Embedding of physical network; ii) Embedding of SFC request and iii) Policy generation. We will explain the components in detail.

*1) Embedding of physical network:* To explore the non-Euclidean structure of network topology, we utilize GCN [10] based on semi-supervised learning to extract features of physical network. At each time step $t$, the current state of physical network $s_t^p = (A, X)$ is fed into a GCN layer to learn a new representation matrix $Z_t \in \mathbb{R}^{\left|N'\right| \times U_{gcn}}$, where $U_{gcn}$ is the units number of GCN layer. The arithmetic operation of GCN is briefly formalized as:

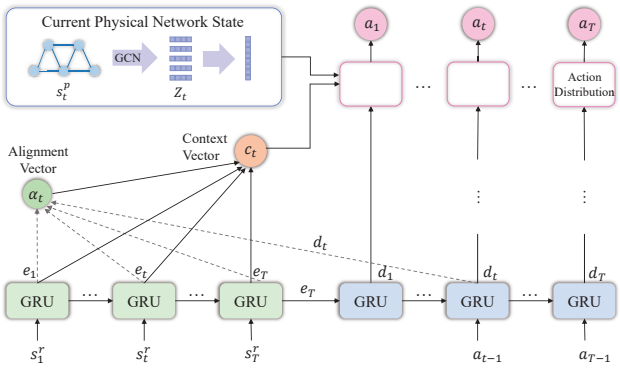


Fig. 2. The network architecture of DRL-SFCP.

$$Z_t = GCN(s_t^p) = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}XW), \tag{8}$$

where $\sigma$ is the activation function, $W$ is the trainable parameters. $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ is the approximated graph convolution filter that is similar to the convolutional neural network (CNN). $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ and $\tilde{A} = A + \Lambda$ is the adjacency matrix of physical network $G'$ with added self-connections using a renormalization trick, where $\Lambda$ is the identity matrix.

*2) Embedding of SFC request:* To capture the orderly requirement of SFC requests, we utilize the encoder of Seq2Seq model which is implemented by gated recurrent unit (GRU) network in this paper. It takes a sequence of inputs $s^r = (s_1^r, \ldots, s_T^r)$ and generates the hidden state $e_t$. For a given time step $t$, the GRU cell takes the current input $s_t^r$ as well as the hidden state of last time step $e_{t-1}$ as the input, and then outputs the hidden state $e_t$ of the current time step:

$$e_t = GRU(s_t^r, e_{t-1}). \tag{9}$$

The details of GRU are described as follows:

$$r_t = \sigma(W_r s_t^r + V_r e_{t-1} + b_r), \tag{10}$$

$$z_t = \sigma(W_z s_t^r + V_z e_{t-1} + b_z), \tag{11}$$

$$\tilde{e}_t = \tanh(W_e s_t^r + V_e(r_t \odot e_{t-1}) + b_e), \tag{12}$$

$$e_t = z_t \odot e_{t-1} + (1 - z_t) \odot \tilde{e}_t, \tag{13}$$

where $r_t$, $z_t$, $\tilde{e}_t$ denote the reset gate, update gate and candidate hidden state, respectively; $\{W, V, b\}$ are the parameters of the corresponding unit; $\sigma(\cdot)$ is the sigmoid activation function; $\odot$ denotes the element-wise multiplication.

*3) Policy generation:* We employ the attentional decoder of Seq2Seq model to generate the appropriate actions. The decoder takes the last state from the encoder $e_T$, and generate an output of size $T$, $a = (a_1, \ldots, a_T)$, based on the current state of the decoder $d_t$ and action $a_{t-1}$:

$$d_t = \begin{cases} GRU(a_{t-1}, d_{t-1}), & t \neq 1, \\ GRU(\varnothing, e_T), & t = 1. \end{cases} \tag{14}$$

To infer a reasonable placement order for a SFC request, context-based attention mechanism [11] is utilized to calculate

the correlations between input sequence and output sequence. The context vector is computed as a weighted sum of $e_j$ which is given by:

$$c_t = \sum_{j=1}^{T} \alpha_{t,j} e_j. \tag{15}$$

The weight $\alpha_{t,j}$ of each $e_j$ is defined as follows:

$$\alpha_{t,j} = \frac{\exp(score(d_t, e_j))}{\sum_{j'=1}^{T} \exp(score(d_t, e_{j'}))}. \tag{16}$$

Here, $score(d_t, e_j)$ measures how well the inputs around position $j$ and the output at position $t$ match, calculated as:

$$score(d_t, e_j) = v_a^{\mathrm{T}} \tanh(W_a[d_t; e_j]), \tag{17}$$

where ";" means the concatenation of two vectors, $v_a^{\mathrm{T}}$ and $W_a$ are trainable variables.

To obtain the probability distribution over candidate actions, we intuitively concatenate the current state of the decoder $d_t$, the current context vector $c_t$ and the flattened output of GCN layer $Z_t$, and then transform it into a fully connected layer to make the final output accordant with the number of physical network nodes. The agent executes an action $a_t$ to select a physical node to accommodate the current VNF based on the conditional probability which is given by:

$$\pi[\cdot|\{a_1, \ldots, a_{t-1}\}, d_t, c_t, Z_t] = \mathrm{softmax}(\widetilde{d}_t), \tag{18}$$

where $\widetilde{d}_t = v_b^{\mathrm{T}} \tanh(W_b[d_t; c_t; Z_t])$.

Furthermore, we leverage the Dijkstra algorithm to find the shortest path connecting $a_t$ and $a_{t-1}$ in the physical network. If there is a path $p_t$ satisfying the bandwidth demand of virtual link, the current VNF is successfully placed onto $a_t$; Otherwise, the current SFC fails to be placed and the physical resources previously occupied by it are released.

### C. Training Method

We employ the well-known A3C [12] approach to accelerate the speed of training and improve the robustness of model. A3C is a "master-worker" parallel training architecture composed of multiple worker agents $U$ and a master agent. Each agent contains two networks: i) The actor network maintains a parameterized placement policy $\pi_\theta(a_t|s_t)$ and generates an action according to the current state; ii) The critic network estimates the Q-value with $V_\omega(s_t, a_t)$ to evaluate whether the action performs well or poorly. After fetching the global shared parameters from the master agent, each agent $u \in U$ initializes its own parameters $\theta_u$ and $\omega_u$ using $\theta$ and $\omega$, explores the environment to collect experiences, and then send them to the master agent to update global shared parameters.

At each time step $t$, each agent selects an action $a_t$ according to the stochastic policy $\pi_{\theta_u}(a_t|s_t)$ and interacts with the environment to obtain the reward $r_t$ and the next state $s_{t+1}$. The critic network calculates the temporal difference (TD) error to evaluate the new state, which is given by:

$$\mathcal{A}(a_t, s_t) = r_t + \gamma V_\omega(s_{t+1}) - V_\omega(s_t), \tag{19}$$

**Algorithm 1:** DRL-SFCP with Parallel Training

**Input :** The physical network topology $G' = (N', L')$; The SFC request $G^i = (N^i, L^i)$; The maximum number of iterations $maxIter$;
**Output:** $placementResult$; $actionSet$; $pathSet$

```
/**Training Process**/
Initialize iter ← 0;
Initialize the parameters of the master agent θ and ω;
Initialize the independent environments for worker agents U;
while iter < maxIter do
    for u in U do
        Synchronize parameters θ_u ← θ and ω_u ← ω;
        Receive a SFC request;
        Sample a trajectory using Eq. (18);
        Obtain the reward r_t and calculate the TD error
          using Eq. (19);
        Update the parameter of critic ω using Eq. (20);
        Adjust the policy of actor π_θ(a|s) using Eq. (21);
    end
    iter++;
end
/**Testing Process**/
Initialize actionSet ← ∅, pathSet ← ∅;
Initialize the trained actor network with θ;
Input the state s^r of G^i to the encoder;
for t = 1, ..., T do
    Get the current state s_t^p of G';
    Extract the features Z_t of G' by GCN;
    Run decoder to select an action a_t;
    if a_t == ζ̄ then
        Undo all the previous placements;
        return false, ∅, ∅;
    end
    if t == 1 then
        Place the VNF n_t^i onto the physical node a_t;
        actionSet.add(a_t);
        Update the state of G' to s_{t+1}^p;
        continue;
    end
    Seek p_t between a_t and a_{t-1} according to Dijkstra;
    if ∃ p_t then
        Place the VNF n_t^i onto the physical node a_t;
        Place the virtual link onto the physical path p_t;
        actionSet.add(a_t);
        pathSet.add(p_t);
        Update the state of G' to s_{t+1}^p;
    else
        Undo all the previous placements;
        return false, ∅, ∅;
    end
end
return true, actionSet, pathSet;
```

where $\gamma \in (0, 1)$ denotes the discount factor.

After processing a SFC placement completely, the critic network updates the parameter $\omega$ by minimizing the squared TD error loss:

$$\omega = \omega - \varepsilon_\omega \frac{1}{T} \sum_t \nabla_\omega \mathcal{A}(a_t, s_t)^2, \tag{20}$$

where $\varepsilon_\omega$ is the learning rate of the critic network.

The actor network updates the policy parameter $\theta$ by

TABLE I
SIMULATION PARAMETERS

| Name | Value | Description |
|---|---|---|
| $U$ | 4 | the number of actor networks |
| $\mu_k$ | 0.001 | the unit price of resource $k$ |
| $\eta$ | 0.001 | the unit price of bandwidth |
| $\varepsilon_\theta$ | 0.00025 | the learning rate of actor |
| $\varepsilon_\omega$ | 0.0005 | the learning rate of critic |
| $\gamma$ | 0.95 | the discount factor of TD error |
| $\xi$ | 0.125 | the reward coefficient |
| $B$ | 64 | the batch size |
| $U_{gcn}$, $U_{emd}$, $U_{enc}$, $U_{dec}$ | 64 | the units number of GCN layer, embbeding layer, encoder hidden states and decoder hidden states |

maximizing the accumulated reward:

$$\theta = \theta + \varepsilon_\theta \frac{1}{T} \sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) \mathcal{A}(a_t, s_t), \tag{21}$$

where $\varepsilon_\theta$ denotes the learning rate of the actor network and $\log \pi_\theta(a_t|s_t)\mathcal{A}(a_t, s_t)$ represents the cross-entropy loss weighted by TD error.

The complete execution process of our DRL-SFCP approach is shown in Algorithm 1.

## IV. PERFORMANCE EVALUATION

### A. Experimental Setup

We randomly generate a physical network with 100 nodes and 500 links, following the Waxman topology model [13], which imitates a medium-sized InP. The resources of nodes and the bandwidth of links in the physical network are uniformly distributed with 50 to 100 units. In each episode, 2000 SFC requests arrive at the system sequentially according to a Poisson process with an average arriving rate of 20 per 100 time units. Specifically, each SFC request consists of different numbers of VNFs from 2 to 15 according to the uniform distribution and has a lifetime exponentially distributed with an average of 400. The node and link resources demand of SFC requests are to the uniform distribution of 2 to 30.

The whole model architecture is built with Tensorflow and Adam optimizer is employed to update the parameters of neural networks. Our simulation experiments are executed on a computer with 2.90 GHz Intel Core i7-10700F CPU and 16 GB RAM. In the training phase, actors utilize $\epsilon$-greedy search strategy to select actions according to the probability distributions, which facilitates to sufficient exploration and excellent exploitation. In the testing phase, only the actor network of the trained agent works to place 2,000 SFC requests, using the greedy search strategy. The values of simulation parameters [1] are given in Table I.

### B. Baseline Algorithms

To verify the effectiveness of the proposed DRL-SFCP approach, we select the following two approaches for comparison:

- *GRC* [6]: A heuristic-based algorithm based on global resource capacity to map VNFs onto physical nodes.
- *MCTS* [8]: A RL-based algorithm using Monte Carlo tree search to make the SFC placement decision.

Both of them utilizes the shortest path algorithm to conduct link mapping.

### C. Experiment Results

Fig. 3 depicts the variation of actor loss and critic loss in the training process. We observe both actor loss and critic loss converge to local optimums of $0.05$ and $0.002$ at the $8000$-th training step. It implies that the approximation with DNNs in DRL-SFCP works well.

The acceptance ratio of all the three algorithms in testing phase is illustrated in Fig. 4. We can observe that at the beginning, the acceptance ratios of all the three algorithms decrease because the resources of the physical network are gradually occupied as SFC requests arrive. DRL-SFCP brings about the highest acceptance ratio of $69.9\%$ at the back half of testing phase. Compared to GRC and MCTS, DRL-SFCP improves performance up to $11.6\%$ and $5.2\%$.

In Fig. 5, we compare the average revenue for all the three algorithms as SFC requests arrive sequentially. After processing $2000$ SFC requests, DRL-SFCP achieves the highest long-term average revenue of $0.062$, up to $9.6\%$ and $4.5\%$ better than GRC and MCTS. It indicates that our algorithm could make SFC placement decisions intelligently assisted by the inherent information of physical network and SFC requests.

To simulate the demand variance of SFC requests between the busy hour and the idle hour, we increase the arrival rate of SFC requests from an average of 10 arrivals to 24 arrivals per 100 time units, rising by 2 in each step. Fig. 6 and Fig. 7 illustrate the acceptance ratio and the long-term average revenue under different arrival rates. It shows that the acceptance ratio decreases and the long-term average revenue grows as the arrival rate increases. Moreover, DRL-SFCP outperforms GRC and MCTS with an average of $8.8\%$ and $4.0\%$ on acceptance ratio, and $9.0\%$ and $4.4\%$ in terms of average revenue. This superior performance might be attributed to the excellent abilities of fitting and generalization of DNN.

The average running time of all the three algorithms for processing a SFC request with different length is depicted in Fig. 8. It shows that with the increase of SFC request's length, DRL-SFCP and GRC make SFC placement decisions faster while the running time of MCTS always maintains a high level. It demonstrates that our algorithm is well-suited for the application in online scenarios.

## V. CONCLUSION

In this paper, we have proposed a novel DRL-based approach named DRL-SFCP for SFC placement problem by modeling it as a MDP. DRL-SFCP combines both GCN and Seq2Seq model to extract the features of the physical network and the SFC request. Additionally, We have utilized A3C algorithm to accelerate the speed of the training procedure and enhance the robustness of our model. Benefiting from

[1]The souce code can be found at https://github.com/GeminiLight/drl-sfcp.

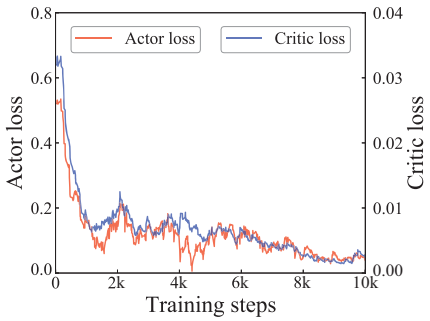


Fig. 3. Losses variation during training.

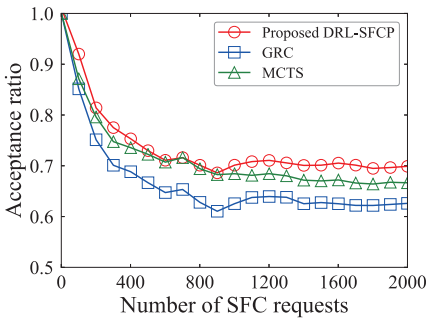


Fig. 4. Acceptance ratio over different SFC requests.

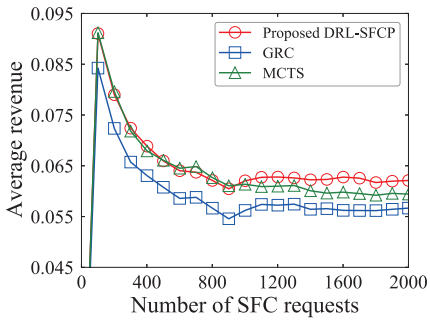


Fig. 5. Average revenue over different SFC requests.

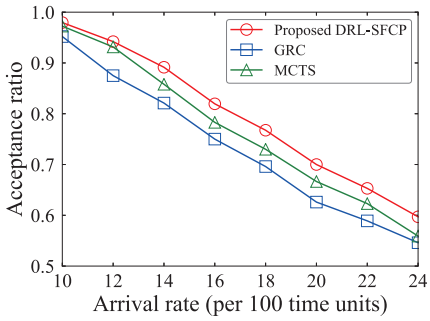


Fig. 6. Acceptance ratio over different arrival rates.

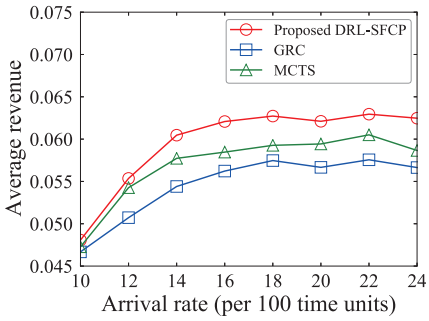


Fig. 7. Average revenue over different arrival rates.

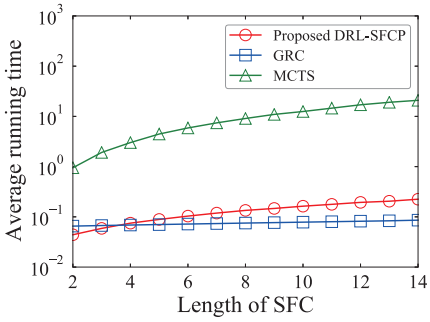


Fig. 8. Average running time over different length of SFC.

the sufficient information captured from the state, DRL-SFCP can generate SFC placement strategies intelligently, which contributes to improving the resource utilization of physical network. Experimental results have shown that our proposed DRL-SFCP makes placement decisions within acceptable time and outperforms the other existing algorithms in terms of the acceptance ratio and the long-term average revenue.

## Acknowledgements

This work is supported in part by Major Special Program for Technical Innovation & Application Development of Chongqing Science & Technology Commission (No. CSTC 2019jscx-zdztzxX0031), National NSFC (No. 61902044, 62072060, 61902178), Fundamental Research Funds for the Central Universities (No. 2020CDJQY-A001, 2020CDJQY-A022, 2018CDXYRJ0030), National Key R & D Program of China (No. 2018YFF0214700, 2018YFB2100100), Chongqing Research Program of Basic Research and Frontier Technology (No. cstc2019jcyj-msxmX0589, cstc2018jcyjAX0340), NSF of Jiangsu (No. BK20190295), Leading Technology of Jiangsu Basic Research Plan (No. BK20192003) and EU's Horizon 2020 Programme under the Marie Skłodowska-Curie (No. 898588).

## References

[1] L. Sanabria-Russo, D. Pubill, J. Serra, and C. Verikoukis, "Iot data analytics as a network edge service," in *Proc. IEEE INFOCOM*, Oct. 2019, pp. 969–970.

[2] I. Sarrigiannis, K. Ramantas, E. Kartsakli, P. Mekikis, A. Antonopoulos, and C. Verikoukis, "Online vnf lifecycle management in an mec-enabled 5g iot architecture," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4183–4194, 2020.

[3] P. Pan, Q. Fan, S. Wang, X. Li, J. Li, and W. Shi, "Gcn-td: A learning-based approach for service function chain deployment on the fly," in *Proc. IEEE GLOBECOM*, Dec. 2020, pp. 1–6.

[4] I. Jang, D. Suh, S. Pack, and G. Dán, "Joint optimization of service function placement and flow distribution for service function chaining," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2532–2541, 2017.

[5] A. Tomassilli, F. Giroire, N. Huin, and S. Pérennes, "Provably efficient algorithms for placement of service function chains with ordering constraints," in *Proc. IEEE INFOCOM*, Apr. 2018, pp. 774–782.

[6] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 1–9.

[7] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Transactions on Network and Service Management*, vol. 13, pp. 725–739, 2016.

[8] S. Haeri and L. Trajković, "Virtual network embedding via monte carlo tree search," *IEEE Transactions on Cybernetics*, vol. 48, no. 2, pp. 510–521, 2018.

[9] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, and F. Liberal, "Virtual network function placement optimization with deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. PP, no. 99, pp. 1–1, 2019.

[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[11] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. ICLR*, May. 2015.

[12] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. ICML*, Jun. 2016, pp. 1928–1937.

[13] Z. Yan, J. Ge, Y. Wu, L. Li, and T. Li, "Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1040–1057, 2020.