Deep-NFVOrch: Deep Reinforcement Learning based Service Framework for Adaptive vNF Service Chaining in IDC-EONs

Baojia Li, Wei Lu, Zuqing Zhu

University of Science and Technology of China, Hefei, Anhui 230027, China, Email: zqzhu@ieee.org

Abstract: By leveraging deep reinforcement learning (DRL), we design Deep-NFVOrch as a novel service framework with resource pre-deployment to achieve adaptive virtual network function (vNF) service chaining in inter-datacenter elastic optical networks (IDC-EONs).

OCIS codes: (060.1155) All-optical networks; (060.4251) Networks, assignment and routing algorithms.

1. Introduction

Nowadays, to expedite the deployment of new services, network function virtualization (NFV) has become increasingly popular in today's inter-datacenter (Inter-DC) networks. This is because NFV enables service providers (SPs) to instantiate virtual network functions (vNFs) over general-purpose network elements in DCs, *e.g.*, servers, switches and storages. Then, to realize a new service on-demand, an SP can simply organize the required vNFs in the desired order and steer application traffic through the formed vNF service chain (vNF-SC). Meanwhile, from the network side, elastic optical networks (EONs) have been widely considered for inter-DC networks (IDC-EONs) [1], since they provide an agile optical layer to adapt to the highly dynamic traffic among DCs. Previously, people have studied how to provision vNF-SCs in IDC-EONs, *i.e.*, deploying the required vNFs in the DCs and setting up lightpaths in the IDC-EON to assemble the required vNF-SCs [2]. However, they overlooked a very important issue in practical vNF-SC provisioning, which is the long latencies from vNF instantiation and lightpath establishment. It was estimated that this issue can make the total setup latency of a vNF-SC reach the order of minutes and thus would jeopardize the quality-of-service (QoS) of network services that have a stringent requirement on setup latency [3].

Therefore, a more reasonable approach is to consider the service framework with pre-deployment [3]. Specifically, in a periodic manner, the framework uses a deep learning (DL) module to predict future vNF-SC requests, and then deploys both the required vNFs and related lightpaths in the IDC-EON in advance. Next, when a vNF-SC request actually comes in, the SP only needs to steer its traffic through the required vNFs in sequence and thus achieves immediate service provisioning. Nevertheless, the framework designed in [3] is still preliminary in the sense that it is not adaptive from two perspectives. Firstly, it invokes pre-deployment periodically with a fixed interval ΔT. However, for vNF-SC requests whose arrival rate is changing, a fixed ΔT can never lead to optimal service provisioning. This is because when ΔT becomes too long, the usage of pre-deployed resources would decrease statistically, while too short of a ΔT would cause frequent and unnecessary network reconfigurations. Secondly, the DL-based request predictor does not collect any feedback during operation, and thus cannot adapt to the changing of requests' arrival pattern.

In this work, we try to resolve the aforementioned issues by leveraging deep reinforcement learning (DRL) to design "Deep-NFVOrch", which is a novel service framework that enables adaptive vNF-SC provisioning in IDC-EONs. Specifically, we design a DRL module based on asynchronous advantage actor critic (A3C) [4] and use it as an observer in Deep-NFVOrch to improve adaptivity. When a service cycle (*i.e.*, including both the pre-deployment and provisioning phases [3]) ends, the observer collects instant performance metrics and uses them as the feedback to update the length of the next service cycle (*i.e.*, ΔT), the DL-based request predictor, and its own DNN, for getting better performance metrics next time. The DRL-based observer leverages an asynchronous training scheme to ensure superior learning capability for online operations. Our simulations indicate that Deep-NFVOrch can realize adaptive

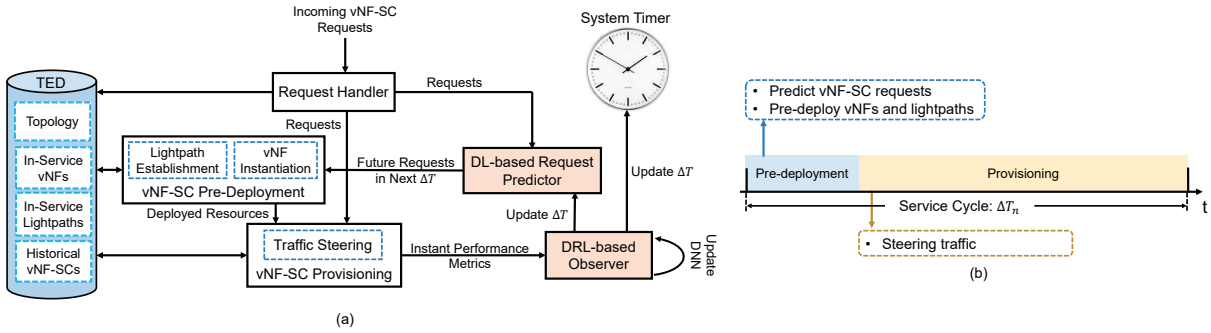

Fig. 1. (a) Architecture of Deep-NFVOrch, and (b) Working principle of Deep-NFVOrch in each service cycle.

vNF-SC provisioning and simultaneously balance the system performance on resource usage, network reconfiguration overhead, and blocking probability, under highly dynamic vNF-SC requests.

2. Proposed Service Framework

Fig. 2(a) shows the proposed architecture of Deep-NFVOrch, which is a periodic system driven with a timer to ensure adaptive service cycles. Each service cycle (*i.e.*, the duration of the n-th one is ΔT_n) includes a pre-deployment phase followed by a provisioning phase, as illustrated in Fig. 1(b). In the pre-deployment phase, the DL-based request predictor (DL-Predictor) forecasts future vNF-SC requests in the next ΔT_n according to historical requests, and then the prediction is sent to the vNF-SC pre-deployment module, which conducts lightpath establishment and vNF instantiation in the IDC-EON accordingly. Next, the system moves to the provisioning phase, which takes most of the time in ΔT_n, and it collects the actual incoming vNF-SC requests to serve. Specifically, when a new request arrives, the request handler records it in the traffic engineering database (TED) and dispatches it to the vNF-SC provisioning module, which steers application traffic through the required vNFs in sequence to realize immediate provisioning. Finally, when the provisioning phase is about to end, the DRL-based observer (DRL-Observer) collects instant performance metrics from the vNF-SC provisioning module, and uses them to evaluate the provisioning in this service cycle. DRL-Observer is trained to update its own DNN and ΔT intelligently based on the performance evaluation. It then forwards the new ΔT (*i.e.*, ΔT_{n+1}) to DL-Predictor and the system timer, and lets the system move to the next service cycle.

We model IDC-EON as a graph $G(V,E)$, where V and E are the sets of DCs and fiber links, respectively. Each DC $v \in V$ contains C_v IT resources for vNFs and there are F frequency slots (FS') on each link $e \in E$. Note that, there are four basic elements in a DRL model, *i.e.*, the agent, action, environment, and reward [4]. The agent is trained to take a proper action based on the environment where it is in and the reward that it has obtained, while the action will change the agent's environment and thus update its reward. In our problem, the agent is DRL-Observer. The action is to select a proper duration ΔT_{n+1} for the next service cycle at the end of service cycle ΔT_n, and we assume that $\{\Delta T_n,\ \forall n \in \mathbb{N}\}$ take discrete values within a finite range $[\Delta T^{min}, \Delta T^{max}]$. The environment $\mathbf{S}_n$ is the current network status, which refers to the current IT resource usage of all the DCs $\{\phi_v^n,\ \forall v \in V\}$, the current spectrum usage of all the fibers $\{\varphi_e^n,\ \forall e \in E\}$, and the number of arrived vNF-SC requests in this service cycle ψ^n. The reward r_n is calculated based on instant performance metrics, as $r_n = \alpha \cdot \bar{\phi} - \beta \cdot \log[\max(p_b, 10^{-6})] + \gamma \cdot \frac{1}{m}$, where $\bar{\phi}$ is the average usage of pre-deployed lightpaths and vNFs, p_b is the blocking probability, m is the number of network reconfigurations (*i.e.*, the total number of new lightpaths and new vNFs), and α, β and γ are positive constants to normalize the three terms.

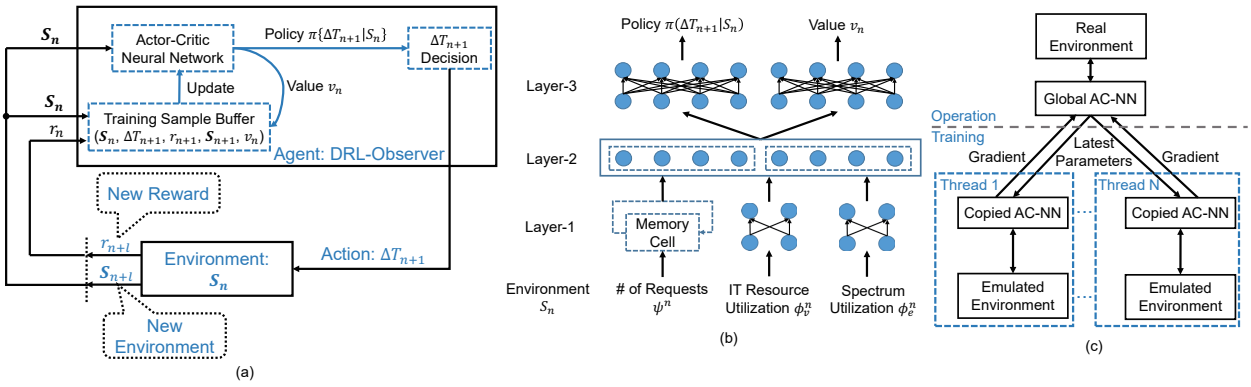

Fig. 2. (a) Architecture and operation of Deep-Observer, (b) Structure of AC-NN and (c) Asynchronously training scheme.

Fig. 2(a) explains the operation principle of DRL-Observer, *i.e.*, how to determine ΔT_{n+1} based on the current environment $\mathbf{S}_n$ and reward r_n. DRL-Observer consists of three parts, which are an actor-critic neural network (AC-NN), a training sample buffer, and a module for deciding ΔT_{n+1}. Here, the AC-NN is trained to take $\mathbf{S}_n$ as the input to generate a policy $\pi(\Delta T_{n+1}|\mathbf{S}_n)$, which is the probability of taking each possible value of ΔT_{n+1} under the current environment $\mathbf{S}_n$. Meanwhile, the AC-NN outputs a value v_n, which is the expected reward under the assumption that the policy is $\pi(\Delta T_{n+1}|\mathbf{S}_n)$ and the current environment is $\mathbf{S}_n$, and it is used to estimate the advantage brought by the selected action (*i.e.*, using ΔT_{n+1} as the length of the next service cycle). The AC-NN's structure is shown in Fig. 2(b), which consists of three layers. Layer-1 is for collecting $\mathbf{S}_n$. It uses a long short-term memory (LSTM) structure to take the number of arrived requests, while the resource utilizations are collected through two fully-connected structures. Layer-2 concatenates the information passed from Layer-1, and Layer-3 uses the concatenated information to get the outputs, which are the policy $\pi(\Delta T_{n+1}|\mathbf{S}_n)$ and the value v_n. Based on the policy provided by the AC-NN, the module

for deciding ΔT_{n+1} selects a proper value within $[\Delta T^{min}, \Delta T^{max}]$. Then, Deep-NFVOrch gets updated and moves to the next service cycle. At the end of ΔT_{n+1}, the environment transforms to $\mathbf{S}_{n+1}$ and DRL-Observer gets the new reward r_{n+1}. Next, DRL-Observer records the tuple $(\mathbf{S}_n, \Delta T_{n+1}, r_{n+1}, \mathbf{S}_{n+1}, v_n)$ as a sample in the training sample buffer.

We develop an online training scheme to ensure that DRL-Observer can be trained on-the-fly during the operation of Deep-NFVOrch. DRL-Observer gets a training sample at the end of each service cycle, but to avoid causing instability, we will not update the AC-NN until M new samples has been obtained. During the update process, DRL-Observer calculates the reward and updates the AC-NN based on the collected samples. Here, the reward is actually a long term reward that summarizes a series of instant rewards multiplied by their discount factors. This is because the environment changes are not memoryless. In other words, the action of selecting ΔT_{n+1} affects not only $\mathbf{S}_{n+1}$ and r_{n+1} but also the subsequent environment states and rewards. Meanwhile, to expedite the training process, we make the updates asynchronous as in Fig. 2(c). Specifically, we partially separate the operation and training of the AC-NN. The operation uses a global AC-NN to determine ΔT, while the training threads make copies of the global AC-NN constantly and let the copied AC-NNs interact with emulated environments and feedback the obtained gradients to the global AC-NN asynchronously.

3. Performance Evaluation

The performance evaluation is based on simulations with an IDC-EON that uses the 14-node NSFNET topology. The IDC-EON supports 5 types of vNFs, and these vNFs can form 10 types of vNF-SCs, each of which includes $[2,4]$ vNFs. We generate dynamic vNF-SC requests based on real wide-area TCP connection traces [5] in which the arrival rate of the requests fluctuates as shown in Fig. 3(a). DRL-Observer can select ΔT_n from $\{1,2,\cdots,10\}$ hours. As Deep-NFVOrch can determine ΔT adaptively, we use the scheme that uses a fixed ΔT as the benchmark. Specifically, the benchmark possesses every model in Fig. 1(a) except the DRL-Observer, replacing it with a fixed setting on ΔT. Three fixed settings are considered in the simulations, *i.e.*, $\Delta T = \{1,5,10\}$ hours. The simulations test different loads by changing the arrived vNF-SC requests per hour for a whole simulation from 50 to 200. The overall resource utilization in Fig. 3(b) refers to the average utilization of deployed lightpaths and vNFs in a simulation. We observe that DRL-Observer achieves significantly higher resource utilization than the benchmarks using fixed ΔT at $\{5,10\}$, verifying the benefit of making ΔT adaptive. The resource utilization of $\Delta T = 1$ is the highest, but to achieve this, it has to invoke network reconfigurations at the highest frequency, causing tremendous overhead. The results on total number of network reconfigurations in Fig. 3(c) confirm this analysis. Here, one reconfiguration is for setting up a new lightpath or deploying a new vNF. The scheme with $\Delta T = 1$ invokes the most reconfigurations, while the reconfigurations by DRL-Observer are even less than those from $\Delta T = 5$, which suggests that DRL-Observer achieves the best tradeoff between resource utilization and reconfiguration overhead. The conclusion can be further verified by the blocking probability in Fig. 3(d), which indicates that DRL-Observer achieves much lower blocking probability than the benchmarks with $\Delta T = \{5,10\}$ and its blocking performance is just slightly worse than that of $\Delta T = 1$.

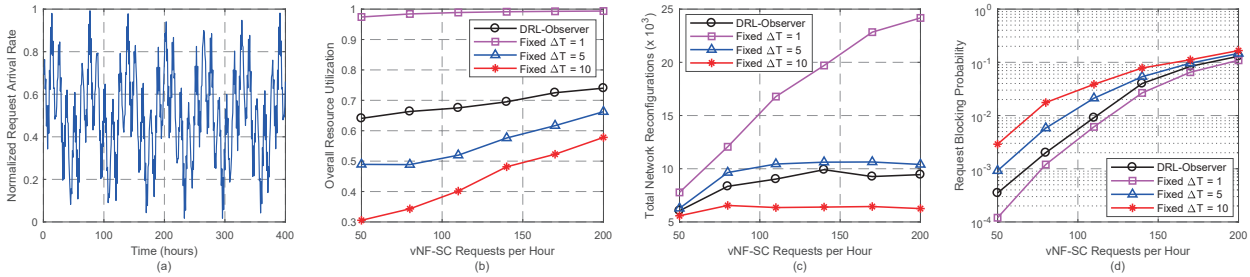

Fig. 3. (a) Dynamic arrival pattern of vNF-SC requests, (b) Overall resource utilization, (c) Number of network reconfigurations, and (d) Request blocking probability.

4. Summary

This work leveraged DRL to design Deep-NFVOrch for realizing adaptive vNF-SC provisioning in IDC-EONs. Simulation results confirmed that Deep-NFVOrch can adjust the duration of service cycles adaptively for properly balancing the performance tradeoff among resource utilization, network reconfiguration overhead, and blocking probability.

References

[1] P. Lu *et al.*, *IEEE Netw.*, vol. 29, pp. 36-42, Sept./Oct. 2015.
[2] W. Fang *et al.*, *IEEE Commun. Lett.*, vol. 20, pp. 1539-1542, Aug. 2016.
[3] B. Li *et al.*, *J. Opt. Commun. Netw.*, vol. 10, pp. D29-D41, Oct. 2018.
[4] V. Mnih *et al.*, *in Proc. of ICML 2016*, pp. 1928-1937, Jun. 2016.
[5] V. Paxson, *IEEE/ACM Trans. Netw.*, vol. 2, pp. 316-336, Aug. 1994.