Deep Reinforcement Learning for NFV-based Service Function Chaining in Multi-Service Networks

Zili Ning*, Ning Wang*, Rahim Tafazolli*
Invited Paper
* 5G Innovation Centre, University of Surrey, Guildford, UK
*Email: z.ning@surrey.ac.uk, n.wang@surrey.ac.uk, r.tafazolli@surrey.ac.uk

(invited paper)

***Abstract*—With the advent of Network Function Virtualization (NFV) techniques, a subset of the Internet traffic will be treated by a chain of virtual network functions (VNFs) during their journeys while the rest of the background traffic will still be carried based on traditional routing protocols. Under such a multi-service network environment, we consider the co-existence of heterogeneous traffic control mechanisms, including flexible, dynamic service function chaining (SFC) traffic control and static, dummy IP routing for the aforementioned two types of traffic that share common network resources. Depending on the traffic patterns of the background traffic which is statically routed through the traditional IP routing platform, we aim to perform dynamic service function chaining for the foreground traffic requiring VNF treatments, so that both the end-to-end SFC performance and the overall network resource utilization can be optimized. Towards this end, we propose a deep reinforcement learning based scheme to enable intelligent SFC routing decision-making in dynamic network conditions. The proposed scheme is ready to be deployed on both hybrid SDN/IP platforms and future advanced IP environments. Based on the real GEANT network topology and its one-week traffic traces, our experiments show that the proposed scheme is able to significantly improve from the traditional routing paradigm and achieve close-to-optimal performances very fast while satisfying the end-to-end SFC requirements.**

***Index Terms*—SFC, NFV, Routing, Reinforcement Learning**

I. Introduction

Network Function Virtualization (NFV) leverages IT virtualization technologies to decouple network functions (NFs) from the dedicated physical devices on which they are executed at industry standard high-volume servers, switches and storage. It is widely reckoned that such a strategy will lead to significant reductions in operating expenses (OPEX) and capital expenses (CAPEX) and facilitate the deployment of new services with increased agility and shorter time to market [1] [2] [3]. Based on NFV, a service function chain (SFC) defines an ordered set of abstract service functions and constraints that should be applied to certain flows [4] [5]. Equipped with NFV and SFC, the role of the underlying network operators will go beyond the traditional data pipes, but in addition they will also be able to offer additional services to giving traffic flows that require such capabilities during their journeys.

978-1-7281-4846-5/20/$31.00 ©2020 IEEE

It is worth noting that not all customer traffic flows need NFV/SFC treatment and at present it is a common view that the majority of the Internet traffic will be still directly routed along static IP based data paths. In such a multi-service networks with coexistence of mixed types of Internet, it is desirable to apply dedicated traffic control mechanisms that cater for different type of traffic, depending on their characteristics and requirements. Specifically, flows without NFV/SFC (known as background traffic in this paper) which dominate today's Internet traffic, are typically routed through the traditional IP routing platforms for simplicity. In contrast, flows requiring NFV/SFC treatment can be controlled by more flexible and dynamic mechanisms according to both SFC requirements and end-to-end path conditions. Giving that both types of traffic share common network resources without necessarily reserving bandwidth resources for NFV/SFC traffic, the routing decision of such flows will need to take into account the resource usage (reflected by link load) by the background traffic whose volume dominates the overall traffic. In this case, depending on the dynamicity of the background traffic behavior, NFV service functions can be adaptively chained and traffic can be also routed based on the real-time condition of the overall traffic distributions. The ultimate purpose is to achieve overall load balancing (in terms of both traffic link load and computing load of NFV nodes) and low maximum flow path delay while satisfying the end to end requirements of NFV/SFC flows.

Based on such a vision, in this paper we propose a network model to present the optimization problem of SFC in such multi-service networks with coexistence of background flows and SFC flows. In order to solve such problem, we introduce a deep reinforcement learning (DRL) scheme to achieve optimized SFC operations. The proposed solution has combined the strengths of deep neural network and reinforcement learning model. With the DRL, it has grasped the complex relation pattern successfully between the traffic demand and optimal virtual links' weights for optimal network performance. And it leverages the existence of some regularity of traffic matrices of different time to work out the near optimal routing path with only training on one traffic matrix. In order to evaluate the performance of the proposed scheme, we also implemented the theoretical mixed linear programming (MILP) model as

the benchmark that can yield the optimal results for theoretical comparison. While MILP solution is quite time consuming to solve this problem, with DRL, we can get the near-optimal route very quickly with small extra effort and very close to the optimal network performance.

From the simulation results based on the real case of GEANT network topology and its one-week traffic traces, we observe that the network performance achieved by DRL is very close to the optimal one produced by MILP, especially when the traffic load is high. To our knowledge, this is the first SFC optimization paper considering both the coexistence between the traditional background traffic and SFC flows considering their interactions.

II. Related Works

For SFC constrained routing, the authors of [6] [7] [8] proposed layered graph to determine the VNF order constrained routing paths. In [9] [10], the authors formulate the problem as integer linear programming and designed heuristic algorithms to work out the near-optimal service function placement and SFC flow routing jointly. In [11], the authors trained the deep belief networks with supervised learning approach, then the path is obtained with hop by hop routing according to the resource conditions in the network. In [12], the authors proposed a deep reinforcement learning for VNF forwarding graph(VNF-FG) embedding to maximize the number ofthe accepted VNF-FGs. While all these related studies have not considered multi-service flows with different routing mechanism coexist at the same time, which are different from this paper.

For applying intelligence to traditional network routing, there are quite a lot of studies on applying reinforcement learning, supervised learning, evolutionary algorithms, heuristic algorithms etc. [13] [14] [15] [16]. In [13], the author proposed a supervised learning to find best route with the consideration of network link usage. In [14], the author used both supervised learning and reinforcement learning to find the best routing path with consideration of minimum network link usage as well. In this study, the author suggested that reinforcement learning is more effective than supervised learning on network routing when the traffic condition do not exhibit high regularity. While in [14], the algorithm will have to train new models every time when new flow demands are coming, it will require quite a lot of time to calculate the optimized route.In [15], the author employed reinforcement learning for routing in heavy traffic network and pointed out that as supervised learning generally requires label information for the training data, it demands for massive manual efforts which is quite common. In [16], the author suggested that heuristics relying on operators' understanding of the workload and environment which is time consuming, and when the parameter setting mismatches with the changing traffic which is quite often in a dynamic environment, heuristics may suffer performance penalty frequently, so the author selected reinforcement learning instead. In this study, we adapted reinforcement learning to SFC routing under multi-service network environment which is a quite different scenario from the studies mentioned above.

III. Network Environment

The underlying network environment is similar to [18], where each network element node for carrying customer traffic supports both legacy IP based static routing and more flexible explicit routing enabled by SDN. The former function is used to carry the background traffic in the traditional manner, while the latter is responsible for dynamically steering SFC flows for service chaining according to network conditions. In this case, each network element is able to simultaneously get involved in the delivery of both plain background traffic and SFC flows. In addition, specific VNF capabilities are strategically embedded at a subset of network nodes, and for both robustness and load balancing requirements, multiple instances of a type of VNF can be embedded at different locations. How to geographically deploy such VNF capabilities has been extensively investigated in the literature and hence is outside the scope of this paper.

First of all, in this paper we consider ordered SFC, meaning that the order of individual VNFs on the end-to-end service chain needs to be pre-determined a priori. At the top level, the optimization objectives include the following 3 key features: (1) To avoid network links that are vulnerable to congestions caused by customer traffic including both background traffic and SFC traffic. (2) Load balancing on IT computing resources (e.g. CPU load) for VNF processing on individual nodes. (3) Optimize the end-to-end latency of the service chains comprising propagation delay, queueing delay (affected by the overall load of traversed links) and data processing delay at VNF nodes along the chain. The priorities of these optimizing objectives can be tuned by the network operator according to specific contexts but in this paper we will only consider representative scenarios.

Before elaborating the modelling of the optimization objectives, we now explain in more details on the working mechanism of the proposed framework. As previously mentioned, the routing of background traffic is based on the preconfigured OSPF or IS-IS link weights which normally remain static at long-time scale. Concerning the routing decision on SFC flows, the SDN controller will play an essential role. First of all, the SDN controller is responsible for monitoring the overall traffic load (i.e. the aggregated load of both background and SFC flows) on each network link as well as the usage of the IT resources for VNF processing. If it is necessary, additional network monitoring mechanisms is required such as NetFlow which is able to monitor the traffic condition for non-SDN flows. Based on this information, the SDN controller periodically computes and determines the actual VNF locations and the end-to-end paths for chaining them based on pre-configured time intervals, e.g. at the time interval of 15 minutes which has been the common practice [19]. To facilitate the SDN controller to determine the optimized path for SFC flows, we introduced a concept of virtual link weights (VLWs) used by the SDN controller for computing

the end-to-end paths for the service chain. In contrast to the static setting of OSPF link weights, based on the periodically monitored overall link load, combined with the static propagation link delay, the SDN controller computes the VLW based on which the shortest paths are determined for enforcing the SFC traffic delivery. The actual modelling of VLWs will be specified in Section III and by applying VLW, the actual path computation is able to take into account not only the objective of minimizing end-to-end path delay but also the requirement of avoiding highly loaded links for SFC flows. To be synchronized with common practice of periodical network monitoring at the order to tens of minutes in many traditional traffic engineering schemes [19] [20], the computation of VLW at the SDN controller side is also configured to be executed at the same time interval.

We now take Fig. 1 as an example. Background traffic (not shown in the figure) is exchanged between every pair of network nodes and routed based on pre-configured OSPF/ISIS link weights. Now we consider an SFC flow from node A destined to D with ordered chain of VNF1 and VNF2 along the end-to-end path. Through network monitoring described above, the SDN controller has the up-to-date knowledge about the traffic load on each link and the IT resources utilizations for VNF processing on individual nodes. Based on such information, the SDN will identify the optimized locations for these VNFs (e.g. nodes B and F respectively) and the end-to-end chaining path as shown in the Figure. Assuming the overall traffic load on link $B \rightarrow F$ is high, the SDN controller can rely on VLW to determine the alternative paths via node C. It is also worth mentioning that, with the

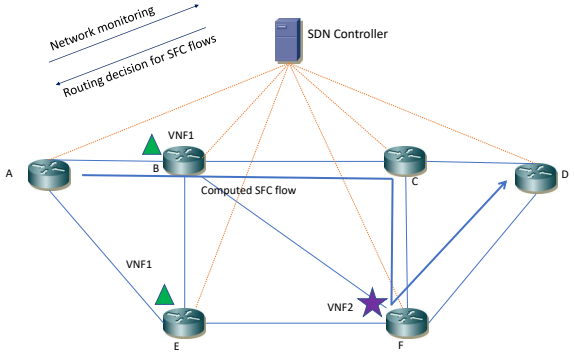

Fig. 1. Network Environment

evolution of networking technologies such as SRv6 [21] or even more radical paradigms, the proposed scheme can be in future supported in more network environments. Specifically, different types of customer traffic, or traffic with different service function chaining requirements can inject necessary metadata (e.g. traffic type, requires network treatments, list of NFV functions etc.) into packet headers with flexible addressing space. In this case, the future network edge is able to classify different types of traffic (e.g. background traffic and SFC traffic in this paper) at the ingress nodes, and further execute specific network treatment according to the metadata information being carried.

IV. Problem Formulation

A. Network Model

We model the network as a directed graph $G = (V, E)$, where V is the set of vertices and E is the set of directed edges. A VNF node can be attached to $v \in V$.

TABLE I
Key Notations

Symbol	Description
G	Directed graph representing the NFV network, $G = (V, E)$
V	The set of routers or servers within the network, $V = \{v_1, v_2, \cdots\}$
E	The set of network links between the nodes, $E = \{e_1, e_2, \cdots\} \forall e = (v_i, v_j), v_i, v_j \in V$.
N	The server nodes which have VNF, $N \subset V, N = \{n_1, n_2, \cdots\}$
F	Virtual network functions, $F = \{f_1^1, \cdots, f_1^n, f_2^1, \cdots\}$, f_i^j is the j^{th} instance of type i VNF f
P	The path set of SFC flows, $P = \{p_1, p_2, \cdots\}, p_i = p_{sd}, i \in \{1, 2, \cdots\}$
p_{sd}	Path from source s to destination d, $p_{sd} = \{s, \cdots, d\}, s, d \in N$.
$D(p)$	Delay of path $p, p \in P$
$D(e)$	Delay of each link $e, e \in E$
$D(n)$	CPU processing delay of VNF node $n, n \in N$
$C(e)$	Bandwidth capacity of each link $e, e \in E$
$C(n)$	CPU capacity of VNF node $n, n \in N$
$U(e)$	Usage of link $e, e \in E$
$U(n)$	CPU Usage of VNF node $n, n \in N$
$L(e)$	Physical length of $e, e \in E$
$R(e)$	Total rate/bandwidth of flow passed link e including both SFC flows and background flows. $e \in E$
$R_{BK}(e)$	Total background flow rate/bandwidth passed link $e, e \in E$
$R_{SFC}(e)$	Total SFC flow rate/bandwidth passed link $e, e \in E$
$R_{SFC}(n)$	Total SFC flow rate/bandwidth passed VNF node $n, n \in N$
T_{BK}	Background flow's traffic matrix.
T_{SFC}	SFC flow's traffic matrix.
T_{BK}^{sd}	An element of background flow's traffic matrix T_{BK}, which is the bandwidth of the background flow for source s to destination d. $s, d \in V$
T_{SFC}^{sd}	An element of SFC flow's traffic matrix T_{SFC}, which is the bandwidth of the SFC flow for source s to destination d. $s, d \in V$
$T_{BK}^{sd}(e)$	Bandwidth of background flow which start from s to d and passed link $e, e \in E$
$T_{SFC}^{sd}(e)$	Bandwidth of SFC flow which start from s to d and passed link $e, e \in E$
$T_{SFC}^{sd}(n)$	Bandwidth of SFC flow which start from s to d and passed node $n, n \in N$

To satisfy the VNF order of the SFC requirement, layered graph mechanism is employed to satisfy the function order like [6]. In this paper we adapt three optimizing objectives for the SFC flows, which are maximum link usage(MLU), maximum path delay(MPD) and VNF maximum usage(VMU). The mathematics model for optimizing these three objectives are as following.

a. Minimize maximum link usage of all edges (MLU)

$$Minimize\ max(U(e_1), U(e_2), ...) \tag{1}$$

where,

$$U(e_i) = \frac{R(e_i)}{C(e_i)} \tag{2}$$

$$R(e_i) = R_{BK}(e_i) + R_{SFC}(e_i) \tag{3}$$

$$R_{BK}(e_i) = \sum_{s,d \in V} T^{sd}_{BK}(e_i) \tag{4}$$

$$R_{SFC}(e_i) = \sum_{s,d \in V} T^{sd}_{SFC}(e_i) \tag{5}$$

b. Minimize maximum path delay of all SFC flows (MPD)

$$Minimize\ max(D(p_1), D(p_2), ...) \tag{6}$$

where,

$$D(p_i) = \sum_{e_i \in p_i} D(e_i) + \sum_{n_i \in p_i} D(n_i) \tag{7}$$

$$D(e_i) = D_{propagation}(e_i) + D_{queue}(e_i) \tag{8}$$

$$D_{queue}(e_i) = \frac{U(e_i)}{1 - U(e_i)} * t^{e_i}_{tx} \tag{9}$$

$$D(n_i) = \frac{U(n_i)}{1 - U(n_i)} * t^{f}_{proc} \tag{10}$$

In (8), $D_{propagation}(e_i)$ is the propagating delay on link e_i. $t^{e_i}_{tx}$ is the average data unit transmitting time of e_i which is depending on the bandwidth of the link. t^{f}_{proc} is VNF f unit processing delay which is depending on the VNF CPU capacity and f. Some other time delays in the link are quite small, so not fully listed. They can also be referred from [6].

c. Minimize VNF maximum usages of all VNF CPUs (VMU)

$$Minimize\ max(U(n_1), U(n_2), ...) \tag{11}$$

where,

$$U(n_i) = \frac{R_{SFC}(n_i)}{C(n_i)} \tag{12}$$

$$R_{SFC}(n_i) = \sum_{s,d \in V} T^{sd}_{SFC}(n_i) \tag{13}$$

B. Deep reinforcement learning solution

In order to solve the optimization problem efficiently within each time interval based on the up-to-date knowledge about traffic conditions, we propose a novel solution based on deep reinforcement learning scheme. Reinforcement learning is a technique of learning how to map situations to actions by maximizing a reward signal returned by the environment [17]. Inspired from [14], our goal is to directly learn close-to-optimal mapping from the SCF traffic demands to the decisions on VNF instance selection as well as the determination of the end-to-end paths. Toward this end, the objective here is to find the optimized mapping from the traffic demands to virtual link weights (VLW) mentioned above and then apply simple shortest-path routing based on the VLW in order to achieve our goal. We are assuming there is a mapping pattern between the network traffic demand and the optimum VLW, and it has also been observed that real-life operating networks normally have regular traffic patterns on daily basis [19]. As such, we are able to train the model based on long-time scales in order to obtain close-to-optimal VLW.

We also adapted the classical Markov Decision Process (MDP) model (S, A, P, R, γ) in our DRL scheme. In Fig.2, the input of the neural network is the dynamic network status S, and the output action A is the VLW. The network status includes dynamic network traffic demands, static network configurations which include link bandwidth capacity, link length, VNF CPU capacity. $S = (T_{sfc}, C(e), L(e), C(n))$. $T_{sfc}, C(e), L(e), C(n)$ are the dynamic and static network information as in Table I. After the model has been trained successfully, when the traffic demands are changing, we do not need to train the model again. Instead, with the incoming new traffic demand as the input to the neural network, the optimization engine is able to work out the new VLW very quickly, based on which the near-optimal routing paths can be obtained based on shortest paths over the updated VLW for SFC traffic.

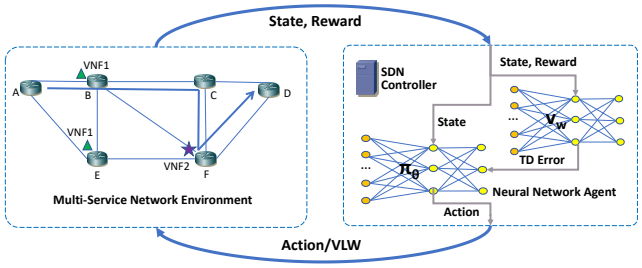

Fig. 2. DRL Train Architecture

Action A as mentioned before is the computed VLW that will be used to calculate the actual routing paths for the SFC flows. With the network performance feedback which is R, the system changes the neural network weights to work out a better action, i.e. VLW. This process is executed iteratively until a converged optimized network performance will be obtained. Reward R is the feedback from the network environment when the agent enforces the action A to the environment. R is used to evaluate the actual performance of this action A that is applied to the environment. To jointly achieve the three optimization objectives, namely MLU, MPD and VMU of (1)(6)(11), we model them together in the MDP reward function. We define R is the weighted sum of MLU, MPD and VMU's reciprocal (14), as the objective is to maximize the reward in DRL. Because the ranges of MLU, MPD and VMU are known, it is flexible to adjust the priorities of MLU, MPD and VMU with the coefficient α, β, λ according to the network requirements.

$$Reward = \frac{\alpha}{MLU} + \frac{\beta}{MPD} + \frac{\lambda}{VMU} \tag{14}$$

As we can evaluate the action is good or bad immediately when we enforce our action to the environment, which means only one action step to reach the terminal status, so the P, γ of the MDP are not necessary here. Although action A has only one step, its potential space can be very broad, and all positive continuous real values can be a potential solution of an action for determining VLW.

Action space $A = R^{|E|}$. $a \in A, a = (w_1, \cdots, w_{|E|})$ is the virtual link weights of all the edges, w_i is the virtual link weight for edge i. For continuous space control [17], our action can be sampled from the Gaussian distribution (15). In (15), $\mu \in R^{|E|}, \Sigma$ is the covariance which is a symmetric positive definite $|E| \times |E|$ matrix. μ and Σ are determined by the neural networks' parameter θ.

$$\pi_\theta(a|s) = \frac{1}{(2\pi)^{|E|/2}|\Sigma|^{1/2}} exp(-\frac{1}{2}(a-\mu)^T\Sigma^{-1}(a-\mu)) \tag{15}$$

To find the optimal θ which can maximize reward in (14), we adapted advantage actor-critic algorithms [17] which is effective and efficient for this type of optimization problems. We also have applied REINFORCE, PPO (Proximal Policy Optimization) and evolution guided policy gradient algorithms, while the comparison between these solutions are out of the scope of this paper. In our algorithm, the θ of actor network π_θ and w of critic network v_w are updated as (16) - (18), t is the iterative sequence number. δ_t is the temporal difference error, as our MDP model has only one step, δ_t is as (17). η^θ and η^w are the learning rate for the actor and critic network.

$$\theta_{t+1} = \theta_t + \eta^\theta \delta_t \nabla ln(\pi_\theta(a_t|s_t)) \tag{16}$$

$$\delta_t = R_{t+1} - v_w(s_t) \tag{17}$$

$$w_{t+1} = w_t + \eta^w \nabla \delta_t \nabla v_w(s_t) \tag{18}$$

The train algorithm for our model is as Algorithm 1.

Algorithm 1 SFC flow routing model training

Input: Traffic matrix of SFC flows(T_{sfc}), layered graph of network environment

Output: Trained model for calculating the near-optimal VLW

1: Initialize actor network π_θ, critic network v_w. Update layered graph traffic with T_{sfc}.
2: **while** reward from layered graph environment not converge, t = 0, 1, 2, ... **do**
3: Sample action a_t with current policy π_θ
4: Get estimated state value from critic v_w
5: Enforce a_t to environment and get reward R_{t+1}
6: Update δ_t as (17)
7: Update Actor network parameters θ as (16)
8: Update Critic network parameter w as (18)
9: Save trained model

V. Simulation and Result Analysis

In order to realistically evaluate the proposed scheme, we take GEANT network topology [22] and its traffic matrix dataset to be used in our simulation experiments. This network contains 23 nodes and 74 directed links. Each traffic matrix is a set of aggregated flow demands between all node pairs at the fixed interval of 15 minutes. In the simulation, we take the original traffic matrix as the background traffic. Without loss of generality, we assume that all node pairs can exchange SFC flows. SFC flows are synthetically generated with bandwidth requirement being statistically proportional to the background flow demand. We consider the scenario that there are two types of VNFs and each type of VNF has two instances distributed in the network topology.

In our simulation experiment, we trained the neural network with one representative TM at first from the dataset, and then applied this trained neural network to compute the optimal network link weights for a longer period of time of 7 days. The input size of the neural network is the length of $S = (T_{sfc}, C(e), L(e), C(n))$ which is 902. The output size of the neural network is 218 which is the size of the action, and the neural network has three hidden layers with size of 256. The optimal routing paths obtained by the MILP solution are calculated by solver Gurobi [23]. With DRL trained model, it takes less than 2ms to get the near-optimal network weights, and need about 20ms to compute the paths for all source-destination pairs by using an Intel Core i5-7200U 2.7GHz, 8G RAM machine. It has been verified that the DRL scheme is much quicker than MILP, consuming around 0.1% of MILP's time consumption. As shown in the performance figures, DRL is able to obtain the network performance very close to the optimal one from MILP.

Fig. 3 shows the network maximum link usage (MLU) obtained by MILP, DRL and OSPF. OSPF is the assumed traditional routing mechanism without optimizing. The MLU of OSPF is the worst among them as expected. The MLU performance obtained by DRL is very close to the optimal solution by MILP, even we only trained one specific traffic matrix and applied this trained model to all the 672 traffic matrices across 7 days. From the figure we can observe that when the network is in high load, the performance from DRL is very well matched with the optimal one, ensuring that the overall traffic do not have potential risks of congestion during the busy period. There is a slight difference when the network load is low during off-peak time, but this certainly does not affect any user experiences given that the traffic load is very low in any case.

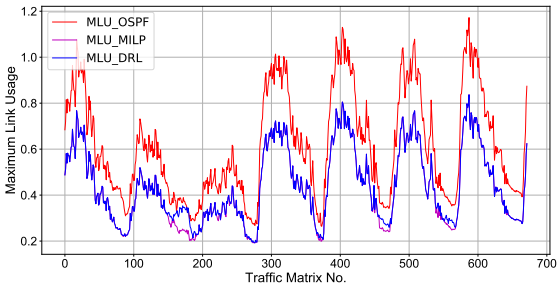

Fig. 3. Maximum Link Usage (MLU) of different solutions

The network max path delay (MPD) is showed in Fig. 4. As expected, when the link usage approaches the physical capacity which is the case for OSPF routing, the end-to-end path delay can be dramatically high. On the other hand, the MPD difference between MILP and DRL on max path delay is less than 1% and consistently maintained below 70ms of all

traffic matrices demands in the simulation.

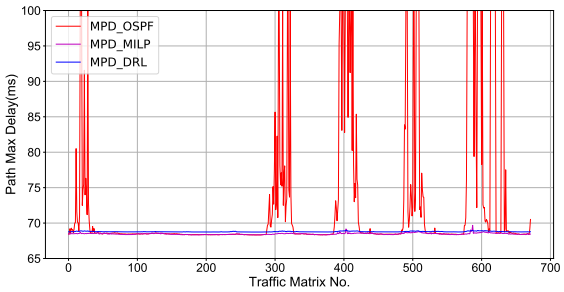

Fig. 4. Max Path Delay (MPD) of different solutions

In Fig. 5, the VNF CPU maximum usages (VMU) of OSPF, DRL and MILP are presented. Although the VMU between the DRL and optimal solution is not as close as other performance metrics in previous figures, as VMU is treated as the least priority in the simulation. The figure still clearly shows significant improvement of the DRL solution compared with OSPF.

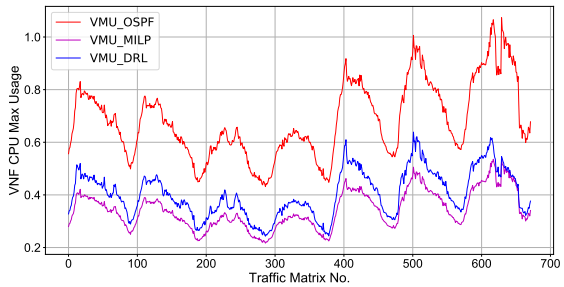

Fig. 5. VNF Maximum Usage (VMU) of different solutions

VI. Conclusion

In this paper we proposed a novel deep reinforced learning (DRL) scheme for computing VNF and path selection decisions leading towards optimized performances in multi-service networks with coexistence of background traffic and SFC flows. The proposed network model can be flexibly adapted to optimize multiple performance metrics including network load balancing, minimizing end-to-end path-delays of all SFC flows, as well as balancing computation load across different NFV nodes. With the DRL solution, we are able to obtain near-optimal network performance that is very close to the optimal solution solved by MILP, and make significant improvement compared with the traditional solution without such intelligence. The DRL is only required a very lightweight training before being applicable to all other different traffic matrices. The result based on the GEANT network traffic has proved that our proposed has the capability to efficiently obtain near-optimal network performances under dynamic traffic conditions very quickly.

Acknowledgement

This work is funded by EPSRC NG-CDI project (EP/R004935/1). The authors would also like to acknowledge the support of the University of Surrey's 5G Innovation Centre (5GIC) (http://www.surrey.ac.uk/5gic) members for this work.

References

[1] ETSI.(Oct.2012), Network Functions Virtualization - Introductory White Paper. https://portal.etsi.org/NFV/NFV_White_Paper.pdf [Online, accessed October 2019].

[2] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck and R. Boutaba, Network function virtualization: State-of-the-art and research challenges, IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pp. 236–262, 2016

[3] B. Yi, X. Wang, K. Li, S. k. Das, M. Huang, A comprehensive survey of Network Function Virtualization, Computer Networks 133 (2018) 212–262

[4] J. Halpern, C. Pignataro, Service function chaining (SFC) architecture, in RFC 7665, October 2015. https://tools.ietf.org/html/rfc7665, 2015, [Online; accessed October 2019].

[5] H. Hantouti, N. Benamar, T. Taleb and A. Laghrissi, Traffic Steering for Service Function Chaining, IEEE Communications Surveys & Tutorials, VOL. 21, NO. 1, FIRST QUARTER 2019

[6] A. Dwaraki and T. Wolf, Adaptive service-chain routing for virtual network functions in software-defined networks, in Proc. Workshop Hot Topics Middleboxes, New York, NY, USA: ACM, 2016, pp. 32–37

[7] G. Sallam, G. R. Gupta, B. Li, and B. Ji, Shortest Path and Maximum Flow Problems Under Service Function Chaining Constraints, IEEE INFOCOM 2018 - IEEE Conference on Computer Communications

[8] N. Huin, B. Jaumard, and F. Giroire, Optimal network service chain provisioning, IEEE/ACM Transactions on Networking, vol. 26, no. 3, pp. 1320–1333, June 2018

[9] M. Tajiki et.al., Joint Energy Efficient and QoS-Aware Path Allocation and VNF Placement for Service Function Chaining, IEEE Transaction on Network and Service Management(TNSM), vol.16, NO.1, March 2019

[10] S. Dräxler and H. Karl, "SPRING: Scaling, Placement, and Routing of Heterogeneous Services with Flexible Structures," 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, 2019.

[11] J. Pei, P. Hong and D. Li, Virtual Network Function Selection and Chaining based on Deep Learning in SDN and NFV-Enabled Networks, IEEE International Conference on Communications, 2018

[12] P. Quang, Y. Aoul, A. Outtagarts, A Deep Reinforcement Learning Approach for VNF Forwarding Graph Embedding, IEEE Transaction on Network and Service Management(TNSM), vol. 16, NO.4, December 2019

[13] J. Reis et al., Deep Neural Networks for Network Routing, International Joint Conference on Neural Networks. Budapest, Hungary. 14-19 July 2019

[14] A. Valadarsky, M. Schapira, D. Shahaf and A. Tamar, Learning to Route with Deep RL, NIPS 2017

[15] R. Ding, Y. Xu, F. Gao, X. Shen and W. Wu, Deep Reinforcement Learning for Router Selection in Network with Heavy Traffic. IEEE Access 2019, 7, 37109–37120.

[16] L. Chen, J. Lingys, K. Chen, F. Liu, AuTO: Scaling Deep Reinforcement Learning for Datacenter-scale Automatic Traffic Optimization, Proc. ACM SIGCOMM, 2018, pp. 191–205

[17] Richard S. Sutton and Andrew G. Barto, Reinforcement Learning - An Introduction 2nd edition

[18] Y. Guo et al, Traffic engineering in hybrid SDN networks with multiple traffic matrices, Computer Networks, 126 (2017) 187-199

[19] N. Wang, K-H. Ho, G. Pavlou and M. Howarth, An Overview of Routing Optimisations In Internet Traffic Engineering, IEEE Communication Surveys and Tutorials, Vol. 10, No. 1, 2008, pp. 36-56

[20] N. Wang, K-H. Ho and G. Pavlou, AMPLE: An Adaptive Traffic Engineering System Based on Virtual Routing Topologies, IEEE Communications Magazine, Vol. 50, No. 3, 2012, pp. 185-191

[21] P. L. Venture, M. M. Tajiki, S. Salsano, C. Filsfils, SDN Architecture and Southbound APIs for IPv6 Segment Routing Enabled Wide Area Networks, IEEE Transactions on Network and Service Management, Vol. 15, Issue 4, 2018

[22] MOME project, https://www.ist-mome.org/database/index20ea.html [Online, accessed March 2019].

[23] Gurobi optimization, https://www.gurobi.com/ [Online, accessed March 2019].