Virtual Network Function Embedding under Nodal Outage using Reinforcement Learning

Swarna Bindu Chetty*, Hamed Ahmadi†, Avishek Nag *
*School of Electrical and Electronic Engineering, University College Dublin, Dublin, Ireland
†Department of Electronic Engineering, University of York, United Kingdom

***Abstract*—With the emergence of various types of applications such as delay-sensitive applications, future communication networks are expected to be increasingly complex and dynamic. Network Function Virtualization (NFV) provides the necessary support towards efficient management of such complex networks, by disintegrating the dependency on the hardware devices via virtualizing the network functions and placing them on shared data centres. However, one of the main challenges of the NFV paradigm is the resource allocation problem which is known as NFV-Resource Allocation (NFV-RA). NFV-RA is a method of deploying software-based network functions on the substrate nodes, subject to the constraints imposed by the underlying infrastructure and the agreed Service Level Agreement (SLA).**

This work investigates the potential of Reinforcement Learning (RL) as a fast yet accurate means (as compared to integer linear programming) for deploying the softwarized network functions onto substrate networks under several Quality of Service (QoS) constraints. In addition to the regular resource constraints and latency constraints, we introduced the concept of a complete outage of certain nodes in the network. This outage can be either due to a disaster or unavailability of network topology information due to proprietary and ownership issues. We have analyzed the network performance on different network topologies, different capacities of the nodes and the links, and different degrees of the nodal outage. The computational time escalated with the increase in the network density to achieve the optimal solutions; this is because Q-Learning is an iterative process which results in a slow exploration. Our results also show that for certain topologies and a certain combination of resources, we can achieve between 70-90% service acceptance rate even with a 40% nodal outage.

I. Introduction

Future communication networks (5G & Beyond) will support various types of technologies and applications. These new technologies and applications need machine-to-human and machine-to-machine communications, which require new types of network services (NSs) [1]. As discussed in [2] the new NSs will have short life span (short-lived services). This makes future networks likely to be more dynamic and complex.

In conventional systems, for the delivery of NSs, like the web services, the network operator defines the NSs according to the promised SLAs. These network services are comprised of a set of Network Functions (NFs). Each NF, such as firewall, router, Network Address Translation (NAT) etc., is integrated onto a dedicated hardware device (or middle-box) to perform a specific function. For the deployment of new services or updating the existing ones, the network operators must purchase, configure, and maintain the middle-boxes, which confine the flexibility and agility of the network functions [2]. This causes a significant rise in the deployment of the middle-boxes, leading to an escalation in the Operating Expenditures (OPEX) and Capital Expenditures (CAPEX) [3]. However, the utilization of this conventional strategy would not be a feasible solution for future communication networks. Especially when new NSs are frequently arriving, and the middle-boxes need to be constantly re-located and re-configured. This will further increase the expenses and reduce scalability. A novel network architecture, therefore, is necessary to support such 'short-lived' services with the flexibility to migrate the NFs, depending on the resource requirements.

The 'Virtualization' approach offers advancement to the network infrastructures by efficiently deploying the NFs. The Network Function Virtualization (NFV) framework virtualizes the NFs into software solutions by decoupling the NFs from their dedicated hardware. These softwarized NFs offer flexibility and agility to the network operators for embedding and re-embedding of network functionalities. The softwarized NFs are called Virtual Network Functions (VNFs) which are deployed on high-volume servers, providing isolation and independence to each VNFs. These VNFs are sequentially chained to determine a requested network service, referred to as Service Function Chaining (SFC). Once the SFC is established, the NFV-MANO (NFV Management and Orchestration) embeds the chained VNFs in a specified order onto the substrate nodes, as shown in Figure 1.

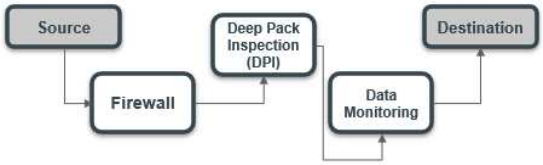

Fig. 1: Service Chain Function

There are a few advantages of adopting NFV-based networks. Firstly, reduction in the CAPEX and OPEX due to the virtualization of the NFs. By allowing more than one VNF to be deployed on a single high-volume server, this gives the advantage of reducing the purchase and maintenance costs of the hardware. Secondly, there will be a reduction in the deployment time for new network services as compared to the traditional method. This is because of the availability of the high-volume servers in the networks. Lastly, NFV-based

networks provide flexible migration of VNFs from one server to another, according to the change in the resource demand.

However NFV-based networks come with its own challenges and one of them is 'NFV-Resource Allocation' (NFV-RA). A SFC consists of multiple VNFs and virtual links with diverse resource requirements; these requested resources must be satisfied by the underlying infrastructure for successful deployment. The provisioning of resources by the network for the successful SFCs deployment is called 'NFV-RA problem'. The main objective of the SFC is to provide promised SLAs to the users, which is accomplished in three stages. In stage 1, a SFC is defined by chaining the VNFs in chronological order, and this process is called VNFs-Chain Composition (VNFs-CC) [3]. In stage 2, these chained VNFs are optimally deployed onto the substrate nodes subject to certain resource constraints. The graphical representation of the sequentially chained VNFs, which deliver an end-to-end network service is called VNF-Forwarding Graph (VNF-FG). The deployment of these graphs onto the network is called VNF-FG Embedding (VNF-FGE). This VNF-FGE problem is branched out into two sub-problems: Virtual Node[1] mapping and Virtual Link[2] mapping, as illustrated in Figure 2. Stage 3 concentrates on minimizing the VNF-FG's overall deployment time by providing a time-slotted strategy for arriving VNFs, this is called VNF Scheduling (VNF-SCH) [3] [4].

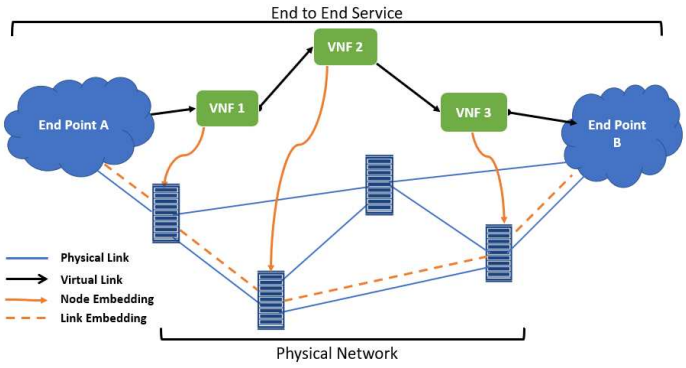

Fig. 2: VNF-FG Embedding

The VNF-FGE and network dynamism enhance the complexity of the NFV-RA problem by creating a challenging task of resource allocation for the requested VNF-FGs. Due to the complexity, the NFV-RA problem is categorized as a NP-hard optimization problem. Using conventional optimization, the global optimal solution can be achieved but at the cost of high computation time [3]. In fact, the computational time will intensify more for denser networks. To limit computational times, researchers have opted for sub-optimal solutions like heuristics-based approaches which provide a trade-off between the run-time and the global optimal solution. The disadvantage of this approach is for dynamic systems where the constraints and objectives are continually varying; these heuristic models are required to be redesigned. Moreover, during the convergence, it tends to get caught up at the local optimum, which produces ineffective results [5].

In this paper, we are considering the placement of dynamically arriving VNF-FGs defined by the network operator, and is performed using the Reinforcement Learning (RL) technique. In an environment where the network conditions change vigorously, learning from the past experiences will be beneficial for decision making; moreover, the RL approach has been proven to provide better solutions than supervised learning for NP-hard problems [6].

Additionally, this work also examines the performance of the Q-Learning (QL) under uncertain conditions like nodal outages due to disaster or vulnerability. Thus, in this work, for solving the VNF-FG problem, we have explored the QL performance under various scenarios like the complexity of VNF-FGs, networks, and node capacity.

The remaining paper is organized as follows. Section II provides a brief literature review. Followed by the problem formulation of VNF-FGE problem, which is modelled as a Markov Decision Process (MDP) and an overview about QL is given in Section III. Results and discussions are presented in section IV. Lastly, section V concludes the paper and additionally provides information about our future plan.

II. Literature Review

To solve the VNF-FGE problem, various approaches like Integer Linear Program (ILP), and Mixed Integer Linear Program (MILP), were proposed based on the characteristics of the problem. However, as mentioned in Section I, the achievement of the global solution is not affordable because of the increased dynamics and complexity of the networks. Therefore, most of the approaches are modelled using the exact method, but the optimal solution is achieved by using heuristic algorithms. The researchers have adopted the ILP approach for embedding the VNF onto the network by considering some strong assumptions like delay and loss rate induced by links [5], [7], which is not realistic. Few authors performed a joint optimization method for VNF mapping and VNF scheduling which are performed using the MILP model. In [8] achieving the optimal solution for a smaller instance is presented. However, for dense and complex networks, the authors proposed a heuristic model which provided a sub-optimal solution with a reduced amount of computation time. Similarly, others suggested a linear relaxation algorithm to obtain the optimal solution [5]. Some techniques are efficient; nonetheless, they do not assure the best solution and some of the parameters like latency, which is retrieved during the execution time are not considered.

In [9], the performance of the online mapping and scheduling is evaluated by using three greedy algorithms based on different greedy criteria: Greedy Fast Processing (GFP), Greedy Best Availability (GBA), and Greedy Least Loaded (GLL). Apart from these approaches, the authors proposed a meta-heuristic algorithm called Tabu Search, and this was implemented to eliminate the local minimum solutions by

[1]Mapping of VNF.
[2]The connection between two consecutive VNFs

keeping a record of all the previously visited solutions. The algorithm works on two objective functions: minimizing the flow time of the SFC and minimizing the cost of the resources. The authors have established a firm foundation for the online mapping and scheduling of the VNFs. However, this algorithm is not useful for more extensive space networks due to the iterative process. The authors neglected the virtual link mapping and its corresponding delays. For a more realistic scenario, the consideration of link delay is essential.

With an attempt to solve the above open issue, the authors of [5] proposed a model based on Deep Reinforcement Learning (DRL) called Enhanced Exploration Deep Deterministic Policy (E^2D^2PG). The author compared the performance of E^2D^2PG with the light network (BtEurope) and the dense network (Uninett). The performance of Uninett network was compromised as it has higher nodes and links. The VNF embedding is still an open problem for the larger-scale networks.

In this paper, we have adopted QL for solving the VNF-FGE problem. This is an early-stage work on studying the robustness of different QL models like Deep QL, Double-Deep QL (DDQL) on different network topologies with full or partial topology information. Towards that goal, here in this paper, we present some simulation results for estimating the performances of compromised networks where the nodal outages occurred due to disasters. Additionally, our models can also be applied for the multi-operator scenario where operators share a limited amount of network topology information.

III. VNF-FGE Problem

A. Problem Formulation

In this work, we consider a discrete time-step method, in which one VNF-FG arrives for deployment at each time-step. That VNF-FG is composed of various VNFs and VLs, depending upon the type of services requested by the users. These VNFs and VLs demand for a definite amount of resources like CPU allocation, Random Access Memory (RAM), Storage, Bandwidth and Latency. In this work, we are adopting the relationship between CPU and RAM as provided in paper [10], that is, each node will contain a certain number of CPU along with its RAM requirement. For VLs, we are considering the latency induced per link and initialize the bandwidth capacity. More details have been explained in section IV.

Objective: The objective of the constructed model is to maximize the service acceptance rate; that is, maximizing the number of VNF-FGs placement onto the networks.

Constraints:

1) One of the critical aspects for a successful VNF deployment is the provisioning of sufficient resources by the substrate nodes, which will satisfy the requested resources by the VNFs. Thus, the first constraint will check the availability of the resources for the VNFs, i.e., $\sum_v Y_h^v p_{v,r} \leq a_{h,r}, \forall h, r$, where $p_{v,r}$ indicates the amount of requested resource r by the VNF v, and $a_{h,r}$ represents the availability of resource r on the substrate node h. Y_h^v is a binary variable, which indicates the placement of the v VNF onto the h substrate node.
2) Placing all VNFs of a VNF-FG onto a single substrate node will cause overload on the links as well as on the substrate nodes. This will lead to inadequate performance by the networks. Thus, it is crucial to deploy the VNFs of a VNF-FG onto different substrate nodes, which will be the second constraint, i.e., $\sum_h Y_h^v \leq 1, \forall v$.
3) The poor performance of a network can also be caused due to the links; thus, the third constraint focuses on the links and its resources. A successful deployment of the VL is achieved when the substrate link satisfies the demanded link requirements like Bandwidth, latency. Therefore, for bandwidth resource, $\sum_m Y_n^m p_{m,b} \leq a_{n,b}, \forall n, b$, where $p_{m,b}$ indicates the requested bandwidth b by the VL m, and $a_{n,b}$ represents the availability of bandwidth b on substrate link n. Y_n^m indicates the placement of m VL onto the substrate link n.
4) After the deployment of the VNFs and its VLs onto the substrate network, a continuous path between the head VNF, and end VNF needs to be defined, i.e., $\sum_n Y_n^m \leq 1, \forall m$.

If all the chained VNFs of a VNF-FG are deployed onto the substrate nodes by satisfying the above constraints (1-4), then it is called successful VNF-FG deployment. Once a VNF-FG is successfully embedded, then the algorithm selects the next VNF-FG for the implementation.

B. Q-Learning

RL is a learning process where at each time-step the agent observes a state S, and accordingly, an action A is executed; based on this action, the environment provides reward and next state. From the obtained reward, the agent improvises the decision strategy for achieving an optimal policy to attain an optimal solution. In other words, RL is a self-learning process where the models are trained with the help of online data. These models learn how to accomplish a definite aim by accumulating the rewards from the environment and avoiding errors from the received penalties. These trained models will maintain the performance even during the network dynamism. RL is based on the Markov Decision Process (MDP) that is, the environment is modelled as an MDP. RL's main aim is to find an optimal policy $\pi^* = S \rightarrow A$ for the decision-maker (agent). This optimal policy is obtained by maximizing the optimal action-value function; i.e., $\pi^*(s_t) = argmax_a Q^*(s_t, a_t)$. This action-value function is estimated using the Bellman Equation, i.e.,

$$Q^\pi(s_t, a_t) = \sum_{s_{t+1}} P(s_{t+1}, r_t | s_t, a_t)(r_t(s_t, a_t) + \gamma \sum_{a_{t+1}} \pi(a_{t+1}|s_{t+1})(Q^\pi(s_{t+1}, a_{t+1})) \quad (1)$$

s_t, a_t, and r_t are the state, action, and reward obtained at time-step t, respectively. Moreover s_{t+1} and a_{t+1} are the state and action for next time-step, and $P(s_{t+1}, r_t|s_t, a_t)$ is the probability of next state and reward for given current state and

action. Moreover, $\pi(a_{t+1}|s_{t+1})$ is the probability of selecting the next action under the π policy and γ is discounting factor.

However, to estimate the optimal action-value, we are considering Q-Learning (QL). QL is an off-policy model-free algorithm, where the target policy learns from the behavior policy to achieve an optimal solution. Thus the Eq.(1) is modified as;

$$Q^{\mu}(s_t, a_t) = r_t(s_t, a_t) + \gamma \max_a Q^{\mu}(s_{t+1}, a) \tag{2}$$

under the μ policy. In other words, the action-value (Q value) is defined as an expected discounted reward when an action is executed for a state under a policy. QL agent intends to explore the unknown environment by executing an action at each time-step for the observed state. According to this, the model receives feedback (in the form of rewards or penalty) from the environment, which helps in improvising the decision strategy. These learning values are stored in a table called Q-table. Therefore, these Q-table values provide crucial information about the state-action pair, which supports in discovering the best action for the current state. After each time-step, the Q value is updated, as shown in Eq.(3).

$$\begin{aligned} Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t)^{old} \times (1-\eta) \\ + \eta(r_t + \gamma \arg\max_a Q(s_{t+1}, a) - Q(s_t, a_t)) \end{aligned} \tag{3}$$

where η is the learning rate which is ranged from 0 to 1, r_t is the reward achieved by the agent for executing action a_t. γ indicates the significance of the future rewards for current state-action pair, we have considered γ as 0.99. $r_t + \gamma \arg\max_a Q(s_{t+1}, a) - Q(s_t, a_t)$ is called learned value, and $\arg\max_a Q(s_{t+1}, a)$ calculates the optimal future Q value.

For this work, we determined the states as the specifications of the VNF-FGs. That is the resources required by the VNFs and VLs, which is represented as $|V|$ x R_{vnf} + $|M|$ x R_{vl}. $|V|$ and $|M|$ represents the number of VNFs and VLs in a VNF-FG, respectively. R_{vnf} and R_{vl} indicated the resource requirement by the VNFs and VLs, respectively. This description of the state is given to the QL agent for estimating the optimum solution. Depending upon the description, the QL agent (i.e., Q table) will provide the best solution in terms of action. These actions are the nodes of the network, which satisfies the requested resources by the VNFs and VLs. The environment is the physical network infrastructure owned by the network operators. Based on the action, the environment rewards the agent only if the selected action successfully embeds the VNFs and VLs; else penalty is assigned to the agent. We have considered local and global rewards. The local reward is given to the agent for providing a satisfying substrate node per VNF, and substrate link per VL, whereas the global reward, is given upon the successful deployment of the VNF-FGs. Thus, the reward function is constructed based on the deployment of the VNFs and VLs onto the satisfying substrate nodes and links, respectively. The reward function also captures the fact that the communication delay between any two VNFs should not exceed the upper bound of the latency.

Once the VNFs are embedded, the path between them is defined according to the specified latency. In the literature, the majority of the researchers considered the path based on the shortest distance; adopting this will cause congestion on a few links, generating a high delay. This method will not be feasible for delay-sensitive applications. Thus we adopted a method where the path per VNF-FG is established based on the upper bound of the delay which we have set to 30 ms in our analyses. Algorithm 1 summarises our model for VNF-FG embedding. We have examined the performance of this model under several scenarios like different network density, resources present in each node, network capacity, and diverse nodal outages levels. Thus we have generated several results based on the network conditions, where few of them are presented in the next section.

Algorithm 1 VNF placement based on Q-Learning

Initialize η, γ, Exploration Rate ϵ, and Threshold Latency
foreach *episode i =1... N* **do**
  foreach *timestep i = 1... T* **do**
    Initialize Q-table $Q(s, a)$
    Select s_0 from state space S as initial state
    Using ϵ-Greedy method select an action a
    if $random(0, 1) > \epsilon$ **then**
      $a_t = \arg\max_a Q(s_t, a)$
    else
      $a_t = random(A(s))$
    end
    if *action embeds VNF successfully* **then**
      Reward is given
    else
      Penalty
    end
    if *all VNFs are placed* **then**
      x ← source(m), y ← Destination(m)
      Latency (x,y) ≤ Threshold Latency
      Reward is given
    else
      Penalty is given
    end
    Update Q table using Eq.(3)
  end
end

IV. Simulation Results

We have successfully constructed QL model for VNF-FGE problem with and without the nodal outages. Currently, in this paper, we considered two distinctive networks as our environment: Netrail Network with 7 Nodes and 10 links and BtEurope Network with 24 Nodes and 37 links [11]. The performances of the networks are evaluated based on the Service Acceptance Ratio (SAR), and Run-time.

Assuming that the link capacity of the substrate network is between 1 Gbps to 10 Gbps, and of 0 to 10 ms delay, which is initialized randomly among the links. Note that, when we

embed the VNF-FGs into the substrate network; we need to ensure that a path between any two VNFs should have a total delay of less than 30 ms which is a sum of these randomly initialized link delays between 0 to 10 ms. We have started by analyzing the performances of the network based on the different nodal capacity. For each run, we have considered one scenario out of four: 2, 4, 8, and 12 core CPU per node where each core can accommodate 2, 4, 8, and 16 VNFs. Therefore, we have evaluated the network performance for 16 different combinations of nodal capacity. However, this whole experiment is again studied for different maximum latency delay of 30 ms, 50 ms, and 100 ms per path of a VNF-FG. In this paper, we are presenting the results of the lowest latency, i.e., 30 ms. Moreover, we examined the performance of the network under the compromised situation by assuming the occurrence of nodal outage due to disasters or vulnerability. The above experiment is analyzed for 30 ms delay with 10% - 50% of the nodal outage.

Each run consists of 100 episodes, and each episode comprises 100 time-steps. Each time-step generates one VNF-FG with a unique combination of resource requirements, within the range of 3-5 VNFs. These VNF-FGs are produced using Erdős-Rényi model [12] with the epsilon value of 0.3. The probability of connectivity among the nodes (VNFs) depends upon the epsilon and number of VNFs per VNF-FG. Thus, these VNF-FGs are generated with a different degree of complexity. The generated VNF-FGs are directed graphs. We have used the Python language for our simulations.

A. Service Acceptance Ratio

Figure 3 and 4 demonstrates the performance of the Netrail network in terms of SAR for 4 core CPU with 16 VNFs per core, and 12 core CPU with 8 VNFs per core scenario.

Considering the Figure 3, the smaller network provides 100% of service acceptance for 0% of the nodal outage. But with the increase in the outage value, the network performances deteriorated. However, considering the Figure 4 with higher nodal capacity, the network provides satisfactory results until 30% of the nodal outage. Nevertheless, for above 50% of the outage, the netrail network rejected more than 50% of the arriving services for both specified scenarios. This is due to the unavailability of the resources in the network to satisfy the requested VNF-FG resource; in other words, the network resources got exhausted.

On the other hand, considering the more extensive network that is BtEurope Network. In Figure 5, for a scenario of 4 core with 4 VNFs per core, provides a gradual degradation in the network performance with the increase in nodal outage value. Like Netrail for 50% of the outage, a significant amount of rejections are noticed. However, the scenario 12 core with 2 VNFs per core, outperformed the remaining cases, even with the 60% of the outage the network provided upto 40% of rejections, unlike the others, as shown in Figure 6. This is due to the high availability of the resources, which was able to embed the majority of the requested VNF-FGs.

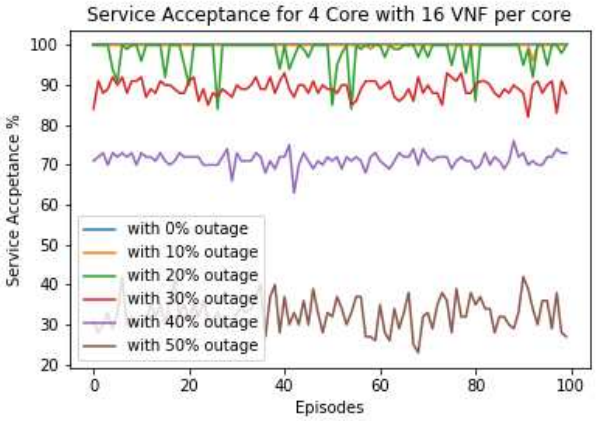

Fig. 3: SAR: Netrail Network with 30 ms Latency

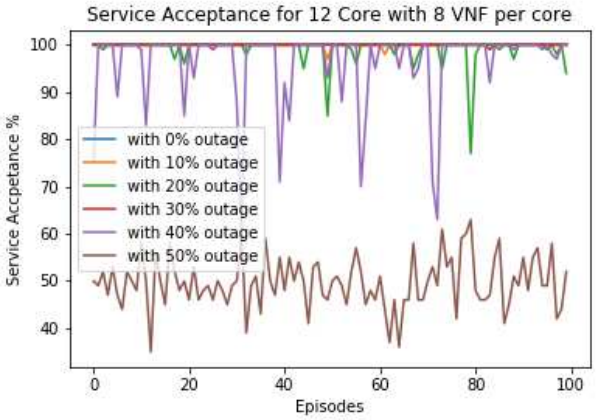

Fig. 4: SAR: Netrail Network with 30 ms Latency

B. Runtime

This section will describe the performance of the networks in terms of runtime. Overhere, we are presenting a runtime comparison between 0% and 30% of nodal outage for both networks, which is demonstrated in the Figure 7 and 8.

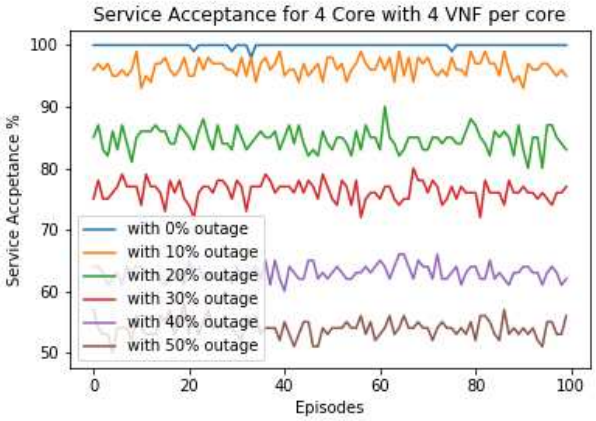

Fig. 5: SAR: BtEurope Network with 30 ms Latency

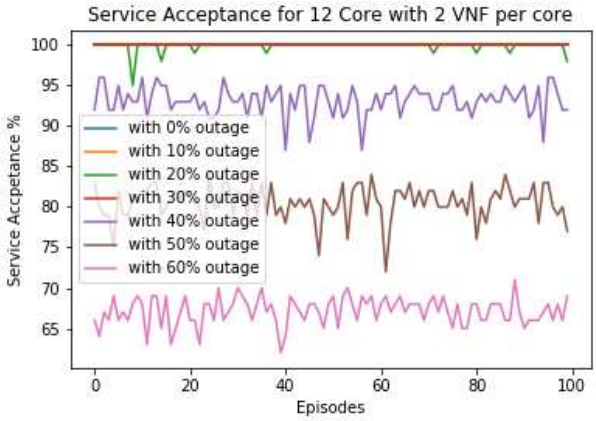

Fig. 6: SAR: BtEurope Network with 30 ms Latency

As expected, the outage network took a significantly high amount of time to solve the embedding problem that is finding the optimal solution for VNF-FG placement. This is because the optimal solution is achieved under high restriction and less availability of resources in the network, as mentioned in above SAR section.

On the comparison between the networks, the Netrail runtime is scantier than BtEurope. This is because, for a small network, the action space is within the range of 7, which causes a faster exploration to obtain the optimal solution. However, for a dense network like BtEurope, where the action space is much larger, this generates a more massive exploration time to find the optimal solution. Moreover, the runtime will enhance furthermore for the denser networks. Hence, this experiment justifies that for the more extensive networks, the runtime will be higher. The reason is that QL finds the optimal solution using the Q-table, which is an iterative process, leading to larger execution time for more extensive state and action space.

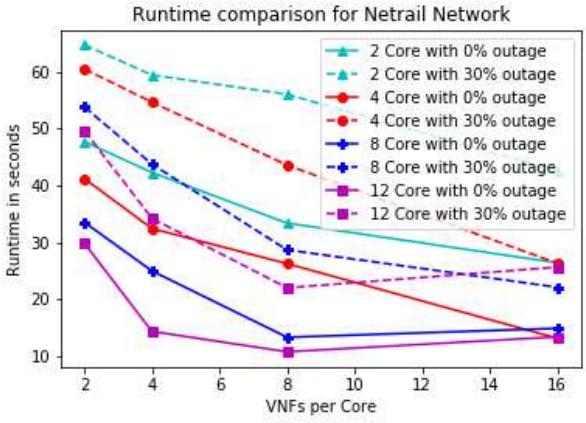

Fig. 7: Runtimes for Netrail Network

Thus this, model is not suitable for the dense network. In future, we will be exploring the advance RL methods for solving the VNF-FGE problem.

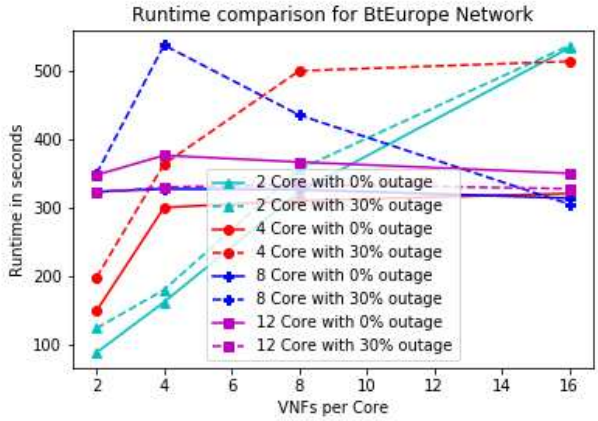

Fig. 8: Runtimes for BtEurope Network

V. Conclusion and Future Work

In conclusion, we examined the performance of Q learning for different network capacity under a various condition like the nodal outages. The achieved results confirm our hypothesis on Q Learning. That is, with the increase in the network density, the service acceptance rate escalates at the cost of high computational time. This will not be suitable for the real-time scenario; thus, our main challenge is the network dynamism and the complexity of VNF-FGE. Therefore, in future work, we will be considering advanced RL models for optimization, like DQL and DDQL. We will also be extending our work for multi-domain networks.

References

[1] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE network*, 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, 2015.

[3] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[4] M. A. T. Nejad, S. Parsaeefard, M. A. Maddah-Ali, T. Mahmoodi, and B. H. Khalaj, "vSPACE: VNF simultaneous placement, admission control and embedding," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 542–557, 2018.

[5] P. T. A. Quang, Y. Hadjadj-Aoul, and A. Outtagarts, "A deep reinforcement learning approach for VNF Forwarding Graph Embedding," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1318–1331, 2019.

[6] P. T. A. Quang, A. Bradai, K. D. Singh, G. Picard, and R. Riggio, "Single and multi-domain adaptive allocation algorithms for vnf forwarding graph embedding," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 98–112, 2018.

[7] Y. Xie, Z. Liu, S. Wang, and Y. Wang, "Service function chaining resource allocation: A survey," *arXiv preprint arXiv:1608.00095*, 2016.

[8] H. Cao, H. Zhu, and L. Yang, "Dynamic embedding and scheduling of service function chains for future SDN/NFV-enabled networks," *IEEE Access*, vol. 7, pp. 39 721–39 730, 2019.

[9] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2015, pp. 1–9.

[10] A. Gupta, M. F. Habib, U. Mandal, P. Chowdhury, M. Tornatore, and B. Mukherjee, "On service-chaining strategies using virtual network functions in operator networks," *Computer Networks*, vol. 133, pp. 1–16, 2018.
[11] The Internet Topology Zoo. [Online]. Available: http://www.topology-zoo.org/dataset.html
[12] P. Erdős and A. Rényi, "On random graphs I," *Publ. math. debrecen*, vol. 6, no. 290-297, p. 18, 1959.