DRIVE: Dynamic Resource Introspection and VNF Embedding for 5G using Machine Learning

Deborsi Basu,* *Graduate Student Member, IEEE*, Soumyadeep Kal†, Uttam Ghosh,‡ *Senior Member, IEEE* and Raja Datta,† *Senior Member, IEEE*

*G. S. Sanyal School of Telecommunications, Indian Institute of Technology Kharagpur, India.

†Dept. of Electronics and Electrical Communication Engineering, Indian Institute of Technology Kharagpur, India.

‡Dept. of Computer Science and Data Science, Meharry Medical College, TN, USA.

d.basu@ieee.org*, s.kal034@kgpian.iitkgp.ac.in†, ghosh.uttam@ieee.org‡, rajadatta@ece.iitkgp.ac.in†

***Abstract*—Network Slicing (NS) technique is comprehensively reshaping the next-generation communication networks (e.g. 5G, 6G). Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) predominantly control the flow of service functions on NS to incorporate versatile applications as per user demands. In the virtualized-Software Defined Networking vSDN environment, a chain of well-defined virtual network functions (VNFs) are installed on Service Function Chains (SFCs) by multiple Internet Service Providers (ISPs) concurrently. Generation, allocation, re-allocation, release and destroying associative VNFs on SFC is an extremely difficult task while keeping high selection accuracy. Towards solving this fundamental issue, in this work, we have proposed a multi-layered SFC formation for adaptive VNF allocation on dynamic slices. We have formulated an ILP to address the VNF-EAP (VNF-Embedding and Allocation Problem) over real network topology (AT&T Topology). Leveraging machine learning techniques we have shown an intelligent VNF selection mechanism to optimize resource utilization. The performance evaluation shows remarkable efficiency on ML-driven dynamic VNF selections over static allocations on SFCs by halving resource usage. Further, we have also studied a VNF typecasting technique for service backup on outage slices in the field of disaster management activities.**

***Index Terms*—SDN, NFV, 5G and beyond, VNF, SFC, TSPs, Machine Learning**

I. Introduction

The forthcoming 5G and beyond communication networks are reshaping the overall network architecture for ultimate user experiences. As the pace of development increases in the networking sector, it calls for better infrastructure to facilitate the growing and increasingly stringent demands from various sectors of society [1], [2]. There's an ever-growing need for better network speed, better connectivity and lower network latency. As a result, many advanced technologies have been brought to the limelight to meet these demands. Concepts like SDN, NFV, NS, ML, and AI are now the forerunners of streamlining the process of delivering better network connectivity but with a manageable cost to the network operators [3], [4]. Contrary to 2G, 3G, and existing 4G-LTE networks and the demands of those areas, modern network demands are now much more varied and the service requirements have also become more strict to ensure better service. As a result, the networks in 5G and beyond are generally 'sliced' to cater to specific needs according to the network's subscribers. A major portion of the realization and maintenance of these slices is from the management and orchestration of network slices. Furthermore, the allocation of resources becomes crucial in such scenarios, hence the introduction of robust machine learning algorithms to help us maintain quality service.

The heterogeneous network slices are done logically based on three major types of subscriber demands. The first one is eMBB (enhanced mobile broadband) which caters to mobile users and the main focus of this slice is maintaining a stable connection with at least moderate to high peak data rates. The second one is mMTC (machine type communication) and as the name suggests, this slice is catered towards meeting demands for communication between different machines. This slice finds itself most utilized in IoT where different sensors and controllers need communication for perfect working. Lastly, the URLLC (ultra-reliable low latency communication) slice is kept for applications and subscribers for whom low latency is crucial for their operation such as smart healthcare applications [5]. As a result, this slice is utilized by the medical sector, disaster management, autonomous vehicular networks, and UAV networks, to name a few [6], [7].

The indigenous traffic characteristics of vast user domains generate random and complex demands. Managing such diverse traffic often requires VNF concatenation and steering of network functions. Chaining and aligning multiple VNFs to serve demand-specific applications are termed SFC (Service Function Chaining) [8]. The advantage of network softwarization and virtualization leveraging SDN and NFV concepts helps multiple VNF instances to run on the shared platform. These VNF instances can be created, allocated, re-allocated, released, deleted, and updated on-demand basis. Orchestrating the VNF instances by optimizing SFC requests is an extraordinarily challenging and difficult task for any network operator. Further, network constraints like restricted resources (network bandwidth, CPU memory, etc.) generate additional hindrances towards traffic steering over shared resources.

Main Motivation Behind the Work:

Recent state-of-the-art analysis shows the importance of the problem. In [9] authors have detailed the challenges faced while deploying next-generation infrastructure for the betterment of humanity. The surge in local tariff plans and uninitialized service plans has already started putting the load on consumers' pockets. The improper service distribution also causes a huge waste of network resources. Though technologies like SDN, NFV, NS, 5G, and next-generation AI extend their comprehensive contributions to resolve such critical issues [10], the existence of suitable resource sharing,

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3235382

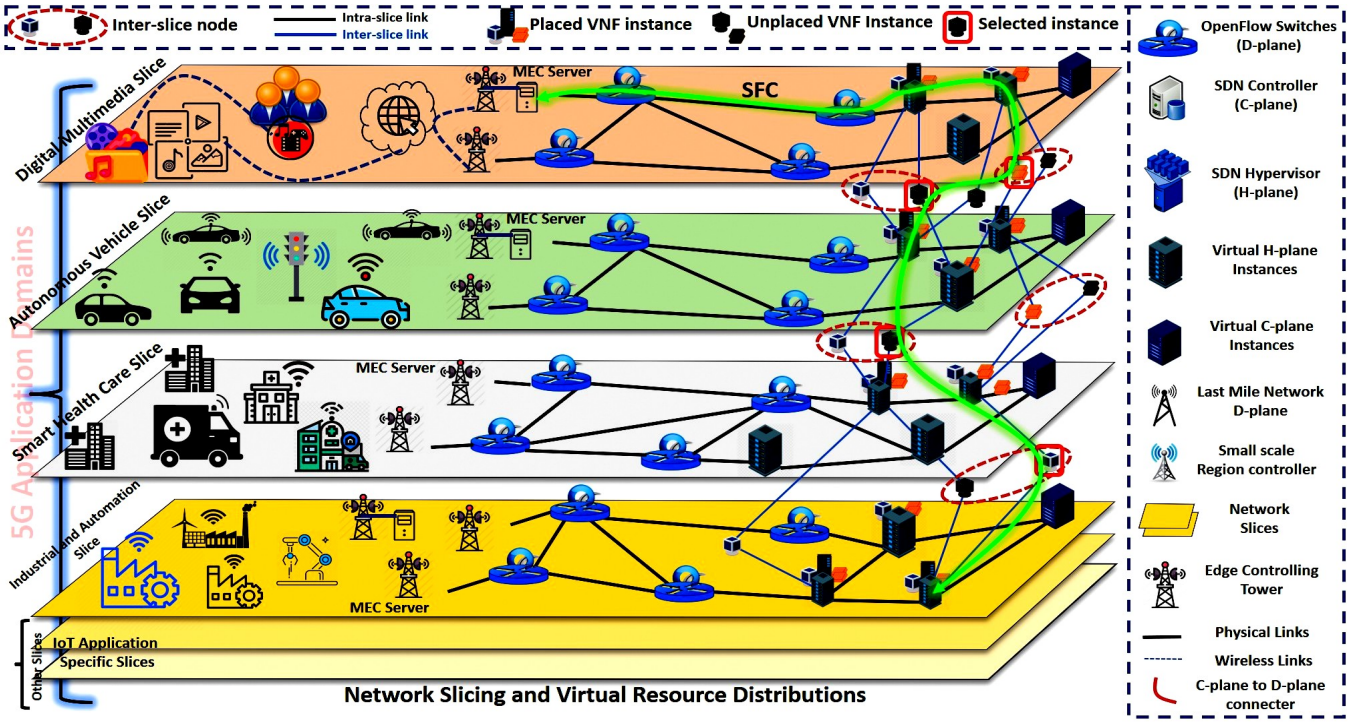


Fig. 1: vSDN-based network slicing architecture for 5G and beyond networks with shared-SFCs

re-sharing, rolling back, and controlling algorithms is still an open area for significant contributions. These factors motivate us to build an exclusive and user-friendly technique for the next-level quality of service.

Understanding the factors is very easy here, given a resource-restricted environment where users are continuously growing. Now, the VNF instances that channel the services need to be coupled in such a way that the formation of SFCs should satisfy the targets. A simplistic overview of SFC formation in a multi-slice environment is shown in Fig. 1. Two main targets which have been addressed here are 1) how to select the suitable VNF clusters from the pool and 2) how efficiently concatenate them following restricted resource availability and network QoS. The problems are aggregated as VNF-EAP and a suitable and novel approach is adopted to solve them over real network topology. The next section explains our major contributions briefly.

Contributions Solving the complex VNF-EAP with dynamic resource sharing is done by formulating a BIP model (Binary Integer Programming model). The target objective is to optimize the QoS that relates to end-to-end SFC cost and resource utilization capacity. We have proposed two novel algorithms termed VNF-READ Algorithm (VNF Re-allocation, VNF Evoke, VNF Allocation, VNF Delete or Detention) and SFC-DRIVE using multi-graph layering.

- AT & T North America topology from [11] is considered as a multi-layered graph with shared-SFCs distributed over the entire region and exchanging information. The demand-based user traffic analysis is modelled following the MIP. The parameters of different factors and their corresponding values are given in tables (1) and (2) respectively.
- Beyond the problem of proper estimation of required resources to fulfil a network demand, we need to also place or 'embed' the VNFs in a network to actually service a particular network request. This brings us to the next part of our work. To actually serve a network request, all the parameters related to the request will have to be accommodated within the limits of the network topology and its capacities.
- Owing to the fact that these network requests are to be serviced by VNFs, the main attribute related to a particular request was its memory demand which is the amount of memory a particular VNF would consume to serve the request, its bandwidth demand which is how much bandwidth it would take up while travelling through the links between the nodes in any given network topology, and finally depending upon the bandwidth demand, there will be demand for CPU resources which we aimed to find out by intelligent methods.
- Keeping in mind the maximum available resources at the links and the nodes we have to route the request through the nodes and links in the topology which provides the fastest and most reliable service function chaining. Only from an end-to-end solution can we appreciate the increase in QoS through realistic VNF embedding along with an intelligent estimation of resource allocation for SFC requests.

Paper Organization: The rest of the paper is organized as follows: related state-of-the-art techniques and the existing solutions are shown in Section II. Section III explains the system model over the real network topology. VNF-EAP is formulated as a BIP problem in Section IV. Section V demonstrates the proposed VNF-READ and SFC-DRIVE algorithms with

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3235382

extensive simulations. Section VI does the result analysis and discussions to show the efficiency of our proposed approach. Finally, Section VII concludes the work with some open research directions.

II. Related Works

Network Slicing has emerged as one of the key areas on which the forthcoming 5G and emerging 6G technologies will rely. Several applications are already running in the market. Researchers have started contributing after identifying the problems. Several novel contributions have already been made but most of them are not sufficient enough to address the ever-growing complexities and all-new challenges. The progress is still growing and evolving [12], [13].

A. Resource Sharing in Network Slicing

A hierarchical decomposition method was proposed by Sunday. O. Oladejo *et al.* addressing the resource allocation problem of NS. They have formulated their optimization model as a maximum utility optimization problem. An extensive solution leveraging Monte Carlo simulation is done using a genetic algorithm. Their findings aim to minimize the misuse of available resources within a resource-constrained environment [11]. Dynamic resource allocation is addressed using a unique PDRA (Priority-Based Dynamic Resource Allocation) Scheme in [14]. An agent-based resource management algorithm is proposed which dynamically allocates resources based on priorities and demand profiles of the incoming request types. The optimization is formed as an LP problem (Linear Programming) on CMDP (Constrained Markov Decision Process). In [15] authors have studied the VNF's demand-based characteristics and predicted the approximate numbers of required functions based on real VNF data. S. Draxler *et al.* put forth a dynamic joint scaling and placement solution with a bi-directional traverse through VNFs and falling back to their respective origins. They have formulated the joint optimization as a MILP (Mixed Integer Linear Programming) model. The proposed concept of VNF reuse supports network functions with pre-defined locations and is static in nature. Their heuristic algorithm solves the NP-complete problem in a short time which they have proved later [16].

B. VNF and SFC Coupling for future 5G

Jianing Pei *et al.* formulated the problem of SFC embedding in a geo-distributed network as a model based on binary integer programming (BIP) aiming to embed SFC requests at the minimum possible cost. Also, their SFC embedding approach which they called SFC-MAP and the dynamic release of VNF instances had been proposed for the purpose of embedding SFC requests and optimizing the number of VNF instances placed to service a particular request. Their performance evaluation results showed their algorithms provided higher performance in terms of SFC request acceptance rate and network throughput [17]. Mohammad M. Tajiki *et al.* in [18] formulated the problem of VNF placement in a network as an Integer Linear Programming based optimization problem while being aware of constraints on delay, link and server utilization. They proposed a heuristic solution to find a near-optimal solution within a realistic real-life time frame. Francesco Malandrino *et al.* studied the area of VNF sharing which involves sharing a particular VNF at a node to service multiple requests and adapting the virtual machines to handle the multiple service requests which are being serviced by it. They also proposed their solution with the goal of reducing total cost and end-to-end delay as well. Their solution provided a near-optimal solution to the VNF embedding problem via VNF sharing which ran in polynomial time [19].

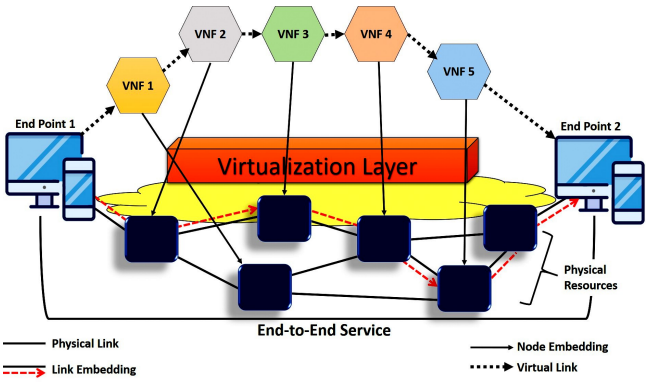


Fig. 2: VNF mapping over SFCs of physical resource units

C. Intelligent Resource Sharing using ML

Recent works are using extensive ML and DL techniques for resource-aware NS. In [20], the authors have implemented a machine learning-based solution for intelligent resource allocation. Their proposed approach has framed the traffic characteristics as a Convolutional Long-Short Term Memory (ConvLSTM). They have optimized the system delay considering a vehicular network. Their optimal slice weight allocation problem is solved using the primal-dual interior-point method. A deep reinforcement learning approach is used to solve the resource allocation problem in [21]. Authors have assumed the resource demand as 'states' and allocated resources as 'actions'. A GAN network model (Generative Adversarial Network model) is used to find the action state values using the deep Q-learning network method. The novelty was to check the system performance in the presence of noise.

The methods explained above do not consider any real network topology while allocating resources on-demand. The random and abrupt networking behaviour produces a strong barrier against the smooth flow of services [22]–[24]. In this work, we have applied the model over the real-network topology emphasising the practicality of our system model and to the best of our knowledge, this approach is the first-ever of its kind.

III. System Model Architecture on AT&T

A. SDN and NFV for NS

The introduction of SDN and NFV in networking opened up a lot of avenues for innovation. Before the introduction of such concepts, the data plane and the control plane had no separation between them and as a result, there was not much control over it, but with the introduction of SDN, the data plane and the control plane were separated which meant, we

could control the data plane more centrally. This meant that we could determine paths for data packets to take without the data having to go through unnecessary routes. As a result of SDN, network security increased as well as costs were cut down since the control plane was more centralized. SDN paved the way for Network Function Virtualization to appear in the picture.

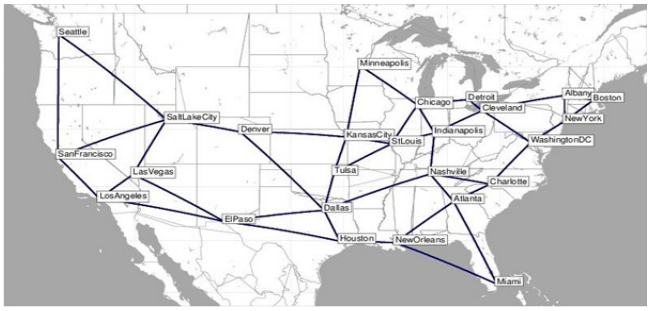


Fig. 3: AT&T North America Topology [25]

Fig. 3 represents our reference topological framework over which the entire experiment has been performed. Individual node points in the graph are the SDN controllers and NFV hypervisor entities [26]. The entire network has been virtualized by installing respective VNFs on the pre-fixed nodes. The SFCs are formed leveraging the connections of multiple VNFs for providing single or multiple network functions. The slicing for dedicated application delivery is done within the programmable hardware units. Table (1) summarizes the parametric entities along with their system model interpretations on AT&T north America topology.

Table I: Sets and Parameters for VNF-EAP

Set/Parameter	***Description***
$\mathcal{N}$	Number of total nodes at AT&T where $n \in \mathcal{N}$; n $\rightarrow$ $node\ samples$
$\mathcal{L}$	Number of links where $l \in \mathcal{L}$
$\mathcal{R}$	Number of requests $req \in \mathcal{R}$
src_{L_i}	Source node for link L_i; $\forall i \in I$
$dest_{L_i}$	Destination node for link L_i $\forall i \in I$
$dist_{L_i}$	Distance between source and destination for L_i
$src_{\mathcal{R}_)}$	Source node for SFC request L_i $\forall i \in I$
$dst_{\mathcal{R}_)}$	Destination node for SFC request L_i $\forall i \in I$
$R_{tx,i}$	Transmission delay of link $i \in \mathbb{E}_k$
$R_{proc,i}$	Processing delay in node n_i
$R_{proc,ij}$	Processing delay in VNF instance i on node j
mem_i	Memory is available on each node

B. Model design for the virtualized plane for SFC

To provide an end-to-end network slice orchestration and management solution, we have formulated a solution for the VNF placement in a network. From the given system model of the proposed solution in Fig. 2, we can see that, in order to solve the problem of embedding VNFs in a network, we need to consider the topology of the given network, that is how many nodes(or base stations) are there, how are they linked and what are the constraints present in them. A particular node can have multiple servers running in them thereby facilitating multiple VNFs and in the process enabling network slicing as well. We can see that for a particular application or communication type, the network has been sliced and all the slices utilize the same topology but different links as well as different VNF instances at the nodes according to their specific requirements. Once again we see the SDN hypervisor coming into play for the purpose of slicing the networks, and the separated control plane and forwarding plane facilitated by SDN coming in handy to maintain control over the slices of the network.

IV. Problem Formulation for VNF-EAP

For this part of the work, we have collected network topological parameters from [25] which contain various network topologies from various regions around the world. Each of these topologies contains $\mathcal{N}$ nodes with given 'x' and 'y' coordinates, $\mathcal{L}$ links connecting any two nodes from the $\mathcal{N}$ nodes and the bandwidth capacity of each link and finally $\mathcal{R}$ number of SFC requests with a source node and destination node. In this work, we have considered the AT & T North America topology (first world country of the US) as shown in Fig. 3. The US topology contains $\mathcal{N}$ = 26 nodes, $\mathcal{L}$ = 84 links and $\mathcal{R}$ = 650 SFC requests per node. The parameters are customizable, so in future, the application-based demand-specific modifications can be done as per any requirements. A flowchart-based representation of the proposed algorithm is given in Fig. 4.

A. XML parsing and model creation

The first step was to parse the XML files from *SDNlib* dataset to extract the data present in the nodes, the links and the SFC requests. Each node contained its ID, its x-coordinate and its y-coordinate. For each link, the data contained were which two nodes were being connected by that link and what was the bandwidth capacity of that node. Finally, for each SFC request, the source node, and the destination node were extracted. Furthermore, each node was allocated some memory (mem_i), V_i instances of VNFs in each node each having a particular VNF type associated with it ($V_{type_{ij}}$), and some CPU computing power V_{ij}. For each link, the Euclidean distance between the source src_{Li} and the destination node (dst_{Li}) is found out, $dist_{Li}$ which has been calculated with the help of the 'x' and 'y' coordinates for src_{Li} and dst_{Li}. Lastly, for the SFC requests, apart from the source node and destination node for each SFC request src_{Di} and dst_{Di}, each SFC request has some CPU computing power required (cpu_{Di}), bandwidth requirement (bw_{Di}), some memory requirement (mem_{Di}) and a VNF type ($V_{type_{ij}}$) associated with it. Furthermore, each request has a lifetime associated with it, because of which the start time and end time of each request, ts_{Di} and te_{Di}, respectively, have been considered. There are two ways the cpu_{Di} has been allocated to each request, one of them being static or fixed allocation of CPU which does not take into consideration how much computing power the request actually demands, the other one being intelligent adaptive CPU allocation which is predicted based on the bandwidth requirement of the request. This intelligent prediction of CPU computing power required is done by Algorithm (1) which is presented earlier. The parameters of the model can be found in Tables (1) and (2). After the modelling of the nodes, links and SFC requests, a multilayered graph is created using an adjacency matrix representation. The multilayered nature of

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3235382

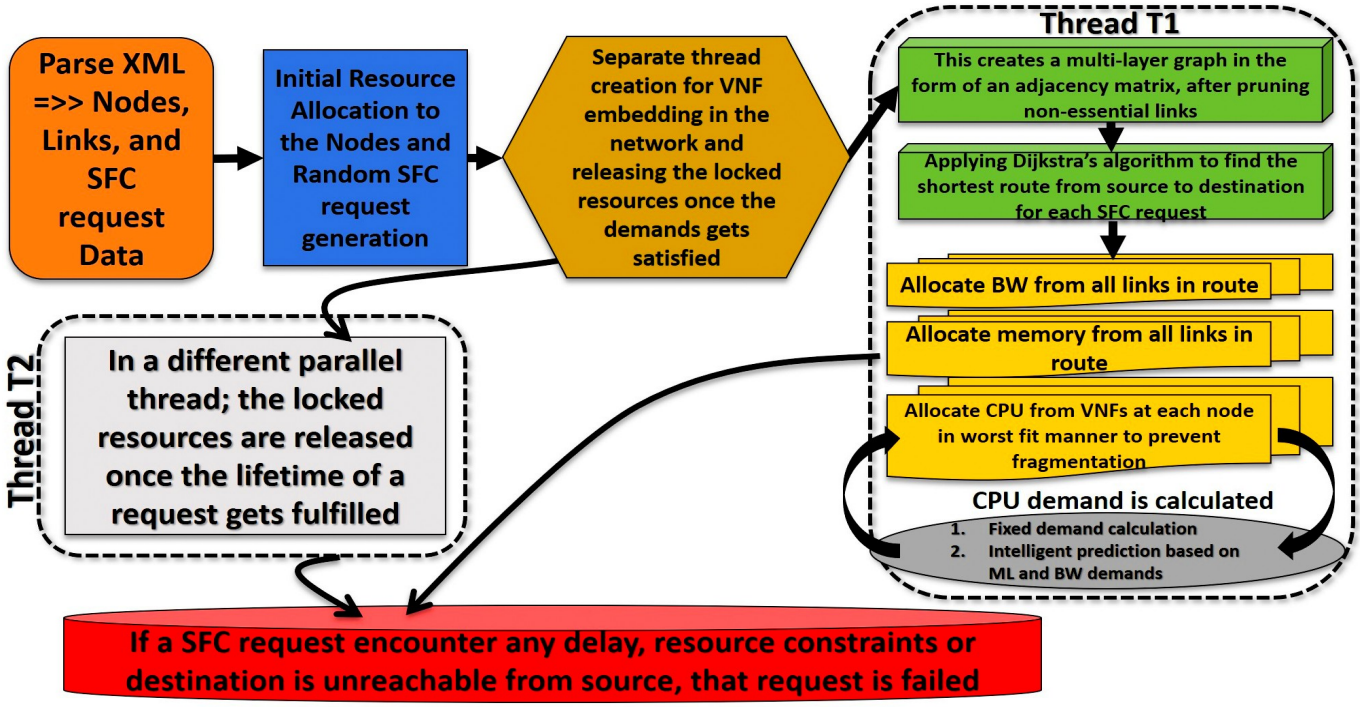


Fig. 4: Algorithmic flow diagram for VNF-READ and SFC-DRIVE

Table II: System model specifications for VNF-EAP

Set/Parameter	*Description*	*Values*
mem_i	Total available memory at each node	1000 MB
v_i	Number of VNF instances at each node i	20
v_{ij}	Total CPU for each VNF instance i at node j	1.0
$V_{type_{ij}}$	VNF type for each VNF instance i at each node j	type 1,..type 4
cpu_{Di}	CPU demand for SFC request i	0.0, 1.0
bw_{Di}	Bandwidth demand for SFC request i	0,.., Max BW of the App.
mem_{Di}	Memory demand for SFC request i	0, 10 MB
cpu_{max}	Maximum allowable CPU for each VNF at each node	1.0
ts_{Di}	Lifetime start time for SFC request i	1, 100 msec.
te_{Di}	Lifetime end time for SFC request i	500, 1500 msec.
$d_{tx,i}$	Transmission delay on link i	1.5 sec.
$d_{proc,i}$	Processing delay on node i	10 sec
$d_{prc,ij}$	Processing delay in VNF instance i on node j	1 msec.
$d_{Drop,i}$	Propagation delay along link i	topolody defined
D_{max}	Maximum tolerable delay	100 msec

the graph originates from the fact that there are multiple VNFs, V_i, at a particular node, which can be used to serve multiple SFC requests simultaneously. For each SFC request, however, this multilayered graph is converted into a single-layer graph, before further VNF allocation. Algorithms (1) and (2) together solve the problem with the heuristic approach and extend a comparative analysis of different state-of-the-art machine learning algorithms. In this work, we have considered SVR and KRR algorithms for the interpolation of average VNF sample points with respect to any new incoming SFC request.

B. Multi-threaded operation for VNF allocation and re-allocation

After the successful creation of a proper model of the network topology and the SFC requests, we propose a multi-threaded approach for the allocation and de-allocation of resources. As time elapses, the start times of the SFC requests are monitored and thread *T1* takes the responsibility of creating the single-layer graph for each SFC request. This thread first prunes all the nodes and links from the graph which contain insufficient resources to service the request. Upon pruning, a minimum distance algorithm is applied to find out the shortest path from the source node to the destination node. For this purpose, we have employed Dijkstra's algorithm to find the shortest route from the source to the destination or target node. After pruning, if we find that the target node is unreachable from the source node, that particular SFC request is taken as failed. For successful service of a request, the following constraints had to be met regarding the limited availability of resources.

$$\sum_{i=0}^{R} mem_{Di} \times z_{ij}^{mem} \leq mem_{max}, \forall j \in \mathcal{N} \quad (1)$$

$$\sum_{i=0}^{R} bw_{Di} \times z_{ij}^{mem} \leq bw_{max}, \forall j \in \mathcal{L} \quad (2)$$

$$\sum_{i=0}^{R} cpu_{Di} \times z_{ij}^{mem} \leq cpu_{max}, \forall j \in \mathcal{N} and \forall k \in V_j \quad (3)$$

Here $z_{ij}^{mem}, z_{ij}^{bw}, z_{ijk}^{cpu}$ are binary variables which indicate j^{th} node is being used by SFC request i or not (1 or 0), j^{th} link is being used by SFC request i or not(1 or 0) and whether k^{th} VNF instance of j^{th} node is being used for SFC request i or not (1 or 0), respectively.

Thread *T1* is also responsible for checking for available resources at each node and links in the route from the source and target. It checks if all the connecting links have enough bandwidth left to service the request. If available, the demanded bandwidth is deducted from the available bandwidth till the end time of that SFC request, te_{Di}. It also checks if all the nodes in the route from source to target have enough available memory to accommodate the SFC request with its memory demand. As far as the CPU goes, we have implemented the worst-fit algorithm amongst all the VNFs, V_i, at a particular node, i, to service the request. This means that, whatever the demand is, the VNF containing the maximum remaining CPU resource capable of meeting the demands and is of the same VNF type, V_{typeij}, as the request, is allocated to that particular SFC. This is done to prevent fragmentation of resources where there might be enough collective resources to provide service but no singular VNF have enough resources to provide the service, thus resulting in failure of that particular request and leading to poor QoS. Each VNF is also associated with a particular type of VNF or service. Each SFC request also has a corresponding VNF type or request type associated with it. At any node, only the same type of VNF can service a request corresponding to that type of request. We have proposed two approaches to how the VNF types are divided at a particular node: a) Uniform division of VNF Types among all the VNFs at a node and b) Random division of VNF Types among all the VNFs at a node, which is a more realistic scenario. We have showcased the results for both of these VNF Type allocations for both the Indian and the US topology.

Beyond the constraints imposed by the limited availability of resources, the VNF embedding process also faces constraints regarding delays. Instead of a simple additive delay model, we have tried to emulate a more realistic scenario as proposed by [27] where the service may be delayed more if there is less amount of resources. This has been brought into play by means of a utilization factor, which is given by, $U = (1 - r_{res})/r_{res}$; where r_{res} is the remaining rate of the resources which is how much resource is remaining after the resources for a particular SFC request has been allocated. Since r_{res} can converge to zero after a particular allocation, it might lead to infinite delay. To avoid this, a lower limit for the remaining rate of resource had been set at 1×10^{-6}. The total end-to-end delay of a particular request is contributed to by three factors which are processing delays at each node, processing delays at each VNF, and the combined effect of both propagation and transmission delays at each link. The total delay from these various factors must be lesser than the total allowable delay. The different delay elements and the delay constraint can be formulated as:

$$D_{tx,i} = d_{drop,i} + d_{tx,i} + \frac{1 - r_{res,bw}}{r_{res,bw}} \times d_{tx,i}, \forall i \in \mathcal{L} \quad (4)$$

$$D_{proc,i} = \frac{1 - r_{res,mem}}{r_{res,mem}} \times d_{proc,i}, \forall i \in \mathcal{N} \quad (5)$$

$$D_{proc,ij} = \frac{1 - r_{res,cpu}}{r_{res,cpu}} \times d_{proc,ij}, \forall i \in \mathcal{N} and \forall j \in V_i \quad (6)$$

$$D_{tx,i} + D_{proc,i} + D_{proc,ij} \leq D_{max} \quad (7)$$

Algorithm 1: SFCs as **MLG - (Multi-layer Graph)**

$\mathbb{G}_k = (\mathcal{N}_k, \mathcal{L}_k, \mathcal{R}_k) \rightarrow$ Graphical Topology;
$\mathcal{V}, \mathcal{L}, \mathcal{I} \Rightarrow$ Nodes, links and VNF instances after the demand of SFC_t where $t \in T$;
$X \rightarrow src_{Li}, dst_{Li}, dist_{Li}, src_{Di}, dst_{Di} \Rightarrow$ Topology information for $\mathbb{G}_k$
$\mathcal{N}, \mathcal{L}, \mathcal{R} \rightarrow$ Pruning the nodes, links, and VNF instances with fewer resources than the demand of SFC requests SFC_i
$GRAPH_{multi} \rightarrow$ Construct MLG using adjacency matrix representation.
Apply Dijkstra's algorithm to find the shortest route from the source node to the target node of SFC request i.
$bw_{Di} \rightarrow$ Deduct demanded bandwidth from all links from source to target,
$mem_{Di} \rightarrow$ Deduct demanded memory from all nodes from source to target,
$cpu_{Di} \rightarrow$ Calculate CPU demand with help of Machine Learning technique,
$cpu_{Di} \leftarrow$ Deduct demanded CPU from all nodes for source to target with same VNF type $V_{type_{kj}}$ as SFC request 'i' and VNF instance 'j' and node 'k' in worst fit manner.
$d_{tx,i}, d_{proc,i}, d_{proc,ij}, d_{prop,i} \leftarrow$ compute delay components to calculate the total compounded delay with utility factor.
if *[constraints (1), (2), (3), (7) all satisfied]* **then**
    acquire lock variable to access shared variable and update success count.
    put resources demanded by SFC request 'i' in the priority queue 'PQ' according to SFC and times.
end
else
    acquire lock variable to access shared value to update failed count.
end
END

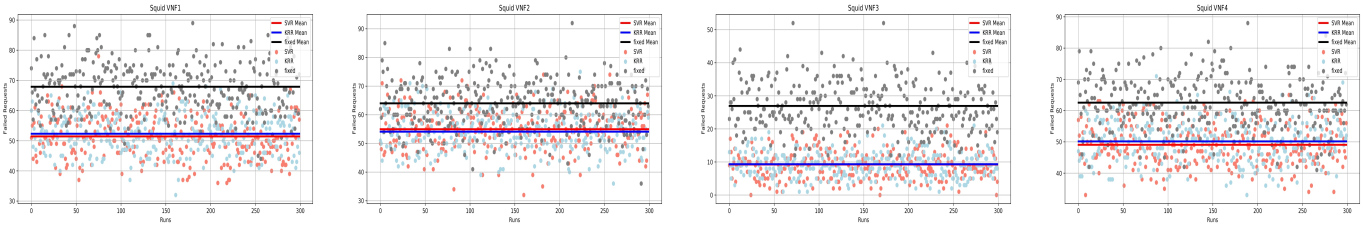
Algorithm 2: Releasing the VNF instances

$\mathbb{G}_k = (\mathcal{N}_k, \mathcal{L}_k, \mathcal{R}_k) \rightarrow$ Graphical Topology;
$\mathcal{V}, \mathcal{L}, \mathcal{I} \Rightarrow$ Nodes, links and VNF instances after the demand of SFC_t where $t \in T$;
$X \rightarrow src_{Li}, dst_{Li}, dist_{Li}, src_{Di}, dst_{Di} \Rightarrow$ Topology information for $\mathbb{G}_k$
$te_{Di,top} \Leftarrow$ earliest ending time of currently running SFC requests from priority queue **PQ**,
$t \geq te_{Di,top}$, Release resources held by the SFC request at the top of **PQ**
Forward the function to check the next status.
END

V. Performance Evaluation

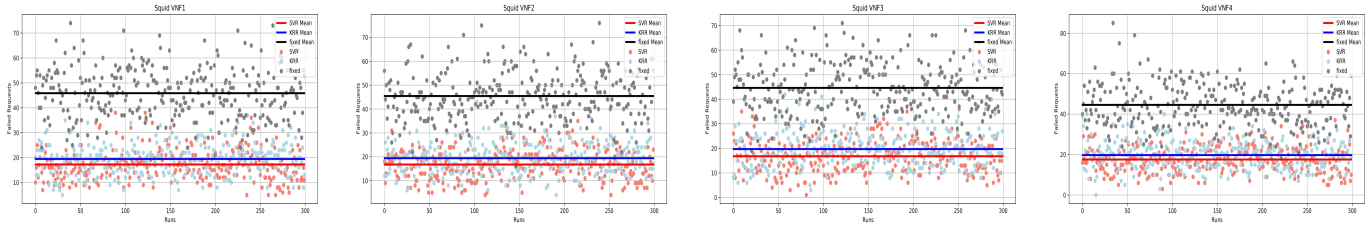
1) Refinements in the Virtual Plane operations: In Fig. 1 we have shown the impact of SFC provisioning for multi-layered network slices. Finding optimal paths to route a network function belonging to a service chain brings about gains in

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3235382



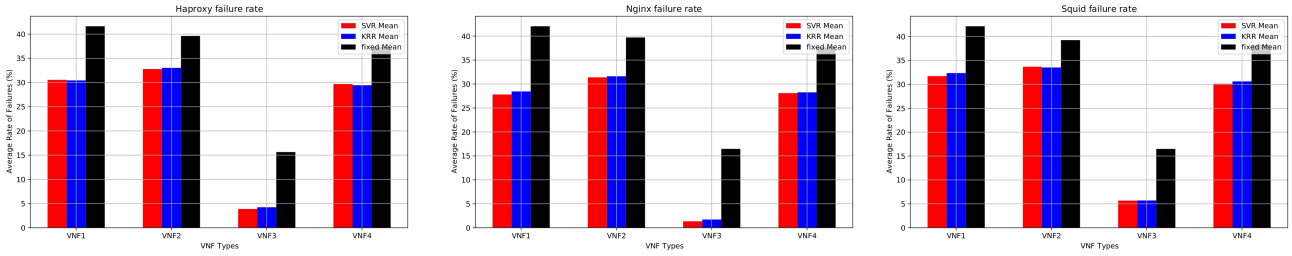
(a) VNF type-1 SFC failure rate (b) VNF type-2 SFC failure rate (c) VNF type-3 SFC failure rate (d) VNF type-4 SFC failure rate

Fig. 5: Distribution of failed SFC requests for uniform VNF distribution for every VNF type on Squid supporting applications



(a) VNF type-1 SFC failure rate (b) VNF type-2 SFC failure rate (c) VNF type-3 SFC failure rate (d) VNF type-4 SFC failure rate

Fig. 6: Distribution of failed SFC requests for random VNF distribution for every VNF type on Squid supporting applications



(a) VNF failure rate on Haproxy (b) VNF failure rate on Nginx (c) VNF failure rate on Squid

Fig. 7: Avg. rate of failed requests for different applications running on different datasets for *static* VNF distribution

Table III: Analysis on shared VNF utilization

Uniform VNF type - resource distribution		Percentage Improvement for VNF types (Absolute percentages)			
Data Set	**Comparative Analysis**	**VNF Type I**	**VNF Type II**	**VNF Type III**	**VNF Type IV**
Haproxy	**Fixed Allocation – to SVR**	19.92%	20.02%	20.31%	20.42%
	Fixed Allocation to KRR	20.56%	20.34%	20.91%	21.06%
	KRR to SVR & vice-versa	0.64%	0.32%	0.60%	0.64%
Nginx	**Fixed Allocation to SVR**	25.70%	25.75%	25.67%	24.86%
	Fixed Allocation to KRR	25.38%	25.39%	25.15%	24.49%
	KRR to SVR & vice-versa	0.32%	0.36%	0.52%	0.38%
Squid	**Fixed Allocation to SVR**	17.47%	17.48%	17.24%	16.59%
	Fixed Allocation to KRR	16.10%	15.87%	15.48%	15.30%
	KRR to SVR & vice-versa	1.37%	1.61%	1.76%	1.30%

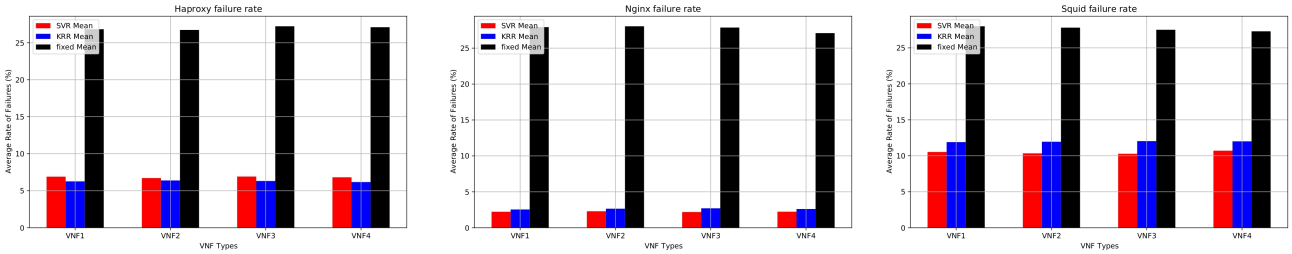
Table IV: Analysis on dynamically shared VNF utilization

Random VNF type - resource distribution		Percentage Improvement for VNF types (Absolute percentages)			
Data Set	**Comparative Analysis**	**VNF Type I**	**VNF Type II**	**VNF Type III**	**VNF Type IV**
Haproxy	**Random Allocation to SVR**	11.07%	6.83%	11.76%	7.70%
	Random Allocation to KRR	11.18%	6.58%	11.41%	7.95%
	KRR to SVR & vice-versa	0.11%	0.24%	0.35%	0.25%
Nginx	**Random Allocation to SVR**	14.24%	8.34%	15.12%	9.64%
	Random Allocation to KRR	13.60%	8.11%	14.75%	9.48%
	KRR to SVR & vice-versa	0.64%	0.24%	0.37%	0.17%
Squid	**Random Allocation to SVR**	10.42%	5.57%	10.81%	8.13%
	Random Allocation to KRR	9.79%	5.72%	10.79%	7.62%
	KRR to SVR & vice-versa	0.63%	0.15%	0.02%	0.51%

the total bandwidth utilization of the links which reduces the energy consumption in the network. Both, the physical as well as virtual links have been considered for this bandwidth calculation. Also, since a demand needs to traverse through all the VNFs present in the service chain, following optimal paths takes lesser time and an overall reduction in the latency of operation in the entire SDN and NFV-enabled network framework can be observed.

2) Service Flow Considerations on shared VNFs: It must be noted that we have shown the variation of latency of the network with the SDN density instead of the network slice density. This is because of the fact that any user demand is generated in the Control Plane and the Virtual Plane is only used to provide different network services. So regardless of how many network slices are embedded, if the functional policy being requested is embedded in a particular slice, then

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3235382



(a) Failure rate for Haproxy dataset (b) Failure rate for Nginx dataset (c) Failure rate for Squid dataset

Fig. 8: Avg. rate of failed requests for different applications running on different datasets for random VNF distribution

that would be used for demand fulfilment. This independence between Slice density and latency of network operation has been shown in Fig. 2. Keeping the SDN density constant we varied the NS density and calculated the corresponding latency values. A slight variation was seen due to averaging of values obtained from 100 model runs but it can be observed clearly that in general, the latency remained constant for a given SDN density. Algorithms (1) and (2) work on the same backbone model with a fixed amount of network resources in uniform and dynamic environments consecutively.

The MLG as shown in Fig. 1, is formed following a logical slicing method on a shared physical backbone network. Individual slices are responsible to serve dedicated network services. The intelligent approach of MLG includes simultaneous distributions of Random and Uniform traffic distributions over the topology under observation. SDN controllers are responsible for the network softwarization and the NFV hypervisor is responsible for the network virtualization phenomenons. Both of these processes together generate multiple graph-like slices where the OpenFlow data-plane switches are controlled by the nodes (SDN controller and NFV hypervisor) units. The wired connection links are the optical fibre connections which are represented as the graphical edges. The modelling of the graph is explained in Section III using the parameters from table (I).

3) System Specification and Complexity Analysis: For VNF placement (Algorithm 1), the XML parsing and demand generation are done in linear time. So the time complexity of these events will be $\mathcal{O}(\mathcal{L})+\mathcal{O}(\mathcal{N})+\mathcal{O}(D)$, where $\mathcal{L}$, $\mathcal{N}$ and D are the total number of links, the total number of nodes and delay per node, respectively. We see that the thread *T1* creates the pruned graph in $\mathcal{O}(\mathcal{L})$ time as only the links' data is required in this step. To find the shortest distance from the source to the target node, we applied Dijkstra's shortest path algorithm that has a complexity of $O(\mathcal{N}^2)$. After finding the shortest path, resource allocation and VNF embedding are done with the help of Algorithm 1, which has the complexity of $\mathcal{O}(\mathcal{N}\times V_i)$, where V_i is the total number of VNF instances at the i^{th} node. A priority queue is maintained for thread *T2* to release resources that have been allocated by thread *T1*. Here insertion of the VNF instances into the priority queue is done as shown in Algorithm 2 that has a complexity of $\mathcal{O}(log[D])$. We find that thread *T2* is only responsible for releasing resources and it is done in the $\mathcal{O}(\mathcal{N}+\mathcal{L})$ time. Therefore, we see that the total complexity of the entire process is $\mathcal{O}(\mathcal{N}^2)$. The simulation has been executed on Google Colab with 12.7 GB of RAM.

VI. Results and Discussions

In the overall result analysis via exhaustive simulations, it is found that our proposed solution gives better QoS under both ideal uniformly distributed VNF types as well as more realistic random VNF type allocation for real network topology. We have taken 300 runs of our VNF placement algorithm each time with randomized SFC requests. From the distribution diagrams in Fig. 5, it is clear that the number of failed SFC requests follows a constant trend throughout the 300 runs of the algorithm for all scenarios that have been considered. Figures (5a) through (5b) show the average failure rate for SFC requests over 300 runs for different VNF types and different applications. In every possible scenario, resource allocation done with the help of machine learning over fixed allocation performs better. In ideal conditions where different types of VNFs are distributed uniformly among all available VNFs at a node, intelligent resource allocation can achieve 15%, 17%, and 20% fewer failed requests than fixed resource allocation in the Indian topology for Squid, Haproxy, and Nginx applications respectively. For the US topology and uniform VNF distribution, our proposed solution achieves 16%, 20%, and 25% fewer failed requests than fixed resource allocation for Squid, Haproxy, and Nginx applications respectively. Fig. 6 shows the case of random VNF distribution over the US topology (AT&T Topology) for *Squid* environment. Figures (7) and (8) show the average rate of failed requests for different applications running on different datasets for static and dynamic VNF distributions respectively. Even under more realistic conditions, where the distribution of different types of VNFs at a particular node is random, our solution still provides 8%, 9%, and 7% better QoS for Haproxy, Nginx, and Squid applications respectively in the Indian topology. On the US topology, for the applications Haproxy, Nginx, and Squid, we achieved respectively an improvement of 11%, 12%, and 8% in QoS. Table (5) summarizes the percentage improvements achieved with our algorithm. While there is a significant improvement in QoS for intelligent resource allocation as compared to fixed allocation, there isn't a significant difference between the performance of the two machine learning models that we have trained for this purpose. While KRR outperforms SVR during the training of the models, the final result of QoS is not significantly affected by that difference. From the above figures and results, it is clear that fixed resource allocation can cause quite a drop in QoS as a significant number of SFC requests to fail due to a lack of resources. This happens because a significant amount of resources is allocated

to requests irrespective of their actual need which results in the consequent failure of further requests as they find themselves without any available resources.

VII. Conclusion and Future Work

In this work, we proposed two novel algorithms (VNF-READ and SFC-DRIVE) to realize suitable solutions for VNF-EAP (VNF-Embedding and Allocation Problem) on real network topology. Service-type specific VNFs are chosen and allocated to the requested SFCs by following static and dynamic methods respectively. The process uses Machine Learning techniques to predict the VNF instances with the minimum errors. The VNF instances are limited, hence the distribution of all such instances must be done intelligibly. The aim is to save the network resources and minimize the false resource claims by the network applications. Our exhaustive result analysis shows significant improvement in VNF classifications over state-of-the-art techniques. We have shown the SFC and VNF failure analysis of the *Squid* dataset. A similar analysis has been conducted for the other two datasets (*Nginz, and Haproxy*). Both uniform and random VNF distributions for all three datasets for a variety of networking conditions are examined parallelly. The potential future work could be further improvements in the accuracy of resource prediction and latency of VNF embedding.

VIII. Acknowledgement

This work was supported by the National Science Foundation, under award number 2219741.

References

[1] J. Hasneen and K. M. Sadique, "A survey on 5g architecture and security scopes in sdn and nfv," in *Applied Information Processing Systems*. Springer, 2022, pp. 447–460.

[2] X. Cheng, Z. Huang, and L. Bai, "Channel nonstationarity and consistency for beyond 5g and 6g: A survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 3, pp. 1634–1669, 2022.

[3] A. A. Gebremariam, M. Usman, and M. Qaraqe, "Applications of artificial intelligence and machine learning in the area of sdn and nfv: A survey," in *2019 16th International Multi-Conference on Systems, Signals & Devices (SSD)*. IEEE, 2019, pp. 545–549.

[4] A. Mughees, M. Tahir, M. A. Sheikh, and A. Ahad, "Towards energy efficient 5g networks using machine learning: Taxonomy, research challenges, and future research directions," *IEEE Access*, vol. 8, pp. 187 498–187 522, 2020.

[5] S. Parui, D. Basu, U. Ghosh, and R. Datta, "A brain to uav communication model using stacked ensemble csp algorithm based on motor imagery eeg signal," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 1–6.

[6] D. Basu, R. Datta, and U. Ghosh, "Softwarized network function virtualization for 5g: Challenges and opportunities," *Internet of Things and Secure Smart Environments*, pp. 147–192, 2020.

[7] N. Fatima, P. Saxena, and M. Gupta, "Integration of multi access edge computing with unmanned aerial vehicles: Current techniques, open issues and research directions," *Physical Communication*, vol. 52, p. 101641, 2022.

[8] D. Qi, S. Shen, and G. Wang, "Towards an efficient vnf placement in network function virtualization," *Computer Communications*, vol. 138, pp. 81–89, 2019.

[9] I. Tomkos, D. Klonidis, E. Pikasis, and S. Theodoridis, "Toward the 6g network era: Opportunities and challenges," *IT Professional*, vol. 22, no. 1, pp. 34–38, 2020.

[10] D. Basu, A. Jain, U. Ghosh, and R. Datta, "Flexarch: Flexible controller placement architecture for hypervisor assisted vsdn-enabled 5g networks," in *2020 IEEE Globecom Workshops (GC Wkshps*. IEEE, 2020, pp. 1–6.

[11] S. O. Oladejo and O. E. Falowo, "Latency-aware dynamic resource allocation scheme for 5g heterogeneous network: A network slicing-multitenancy scenario," in *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2019, pp. 1–7.

[12] W. Wu, C. Zhou, M. Li, H. Wu, H. Zhou, N. Zhang, X. S. Shen, and W. Zhuang, "Ai-native network slicing for 6g networks," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 96–103, 2022.

[13] H. Chergui, L. Blanco, L. A. Garrido, K. Ramantas, S. Kukliński, A. Ksentini, and C. Verikoukis, "Zero-touch ai-driven distributed management for energy-efficient 6g massive network slicing," *IEEE Network*, vol. 35, no. 6, pp. 43–49, 2021.

[14] H. Ko, J. Lee, and S. Pack, "Priority-based dynamic resource allocation scheme in network slicing," in *2021 International Conference on Information Networking (ICOIN)*. IEEE, 2021, pp. 62–64.

[15] S. Schneider, N. P. Satheeschandran, M. Peuster, and H. Karl, "Machine learning for dynamic resource allocation in network function virtualization," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2020, pp. 122–130.

[16] S. Dräxler, S. Schneider, and H. Karl, "Scaling and placing bidirectional services with stateful virtual and physical network functions," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 2018, pp. 123–131.

[17] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2179–2192, 2018.

[18] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari, "Joint energy efficient and qos-aware path allocation and vnf placement for service function chaining," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 374–388, 2018.

[19] F. Malandrino, C. F. Chiasserini, G. Einziger, and G. Scalosub, "Reducing service deployment cost through vnf sharing," *IEEE/ACM Transactions on Networking*, vol. 27, no. 6, pp. 2363–2376, 2019.

[20] Y. Cui, X. Huang, D. Wu, and H. Zheng, "Machine learning based resource allocation strategy for network slicing in vehicular networks," in *2020 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, 2020, pp. 454–459.

[21] Y. Hua, R. Li, Z. Zhao, X. Chen, and H. Zhang, "Gan-powered deep distributional reinforcement learning for resource management in network slicing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 334–349, 2019.

[22] A. Mohamad and H. S. Hassanein, "On demonstrating the gain of sfc placement with vnf sharing at the edge," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[23] R. Behravesh, D. Harutyunyan, E. Coronado, and R. Riggio, "Time-sensitive mobile user association and sfc placement in mec-enabled 5g networks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3006–3020, 2021.

[24] Y.-T. Chen and W. Liao, "Mobility-aware service function chaining in 5g wireless networks with mobile edge computing," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[25] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, April 2007, http://sndlib.zib.de, extended version accepted in Networks, 2009. [Online]. Available: http://www.zib.de/orlowski/Paper/OrlowskiPioroTomaszewskiWessaely2007-SNDlib-INOC.pdf.gz

[26] D. Basu, R. Datta, U. Ghosh, and A. S. Rao, "Load and latency aware cost optimal controller placement in 5g network using snfv," in *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications*, 2020, pp. 106–106.

[27] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2179–2192, 2019.