Article:
Chetty, Swarna Bindu, Ahmadi, Hamed orcid.org/0000-0001-5508-8757 and Nag, Avishek (2024) A DDPG-based Zero-Touch Dynamic Prioritization to Address Starvation of Services for Deploying Microservices-based VNFs. IEEE Transactions on Machine Learning in Communications and Networking. pp. 526-545. ISSN: 2831-316X

https://doi.org/10.1109/TMLCN.2024.3386152

IEEE ComSoc IEEE Communications Society | IEEE COMPUTER SOCIETY | IEEE Signal Processing Society | VTS Connecting the Mobile World
IEEE Transactions on Machine Learning in Communications and Networking

Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version XX Month, XXXX.

Digital Object Identifier 10.1109/TMLCN.2022.1234567

A DDPG-based Zero-Touch Dynamic Prioritization to Address Starvation of Services for Deploying Microservices-based VNFs

Swarna B. Chetty[1], Hamed Ahmadi[1] *Senior Member, IEEE*, and Avishek Nag[2] *Senior Member, IEEE*

[1]School of Physics, Engineering and Technology, University of York, UK
[2]School of Computer Science, University College Dublin, Ireland

ABSTRACT The sixth generation of mobile networks (6G) promises applications and services with faster data rates, ultra-reliability, and lower latency compared to the fifth-generation mobile networks (5G). These highly demanding 6G applications will burden the network by imposing stringent performance requirements. Network Function Virtualization (NFV) reduces costs by running network functions as Virtual Network Functions (VNFs) on commodity hardware. While NFV is a promising solution, it poses Resource Allocation (RA) challenges. To enhance RA efficiency, we addressed two critical subproblems: the requirement of dynamic service priority and a low-priority service starvation problem. We introduce 'Dynamic Prioritization' (DyPr), employing an ML model to emphasize macro- and microlevel priority for unseen services and address the existing starvation problem in current solutions and their limitations. We present 'Adaptive Scheduling' (AdSch), a three-factor approach (priority, threshold waiting time, and reliability) that surpasses traditional priority-based methods. In this context, starvation refers to extended waiting times and the eventual rejection of low-priority services due to a 'delay. Also, to further investigate, a traffic-aware starvation and deployment problem is studied to enhance efficiency. We employed a Deep Deterministic Policy Gradient (DDPG) model for adaptive scheduling and an online Ridge Regression (RR) model for dynamic prioritization, creating a zero-touch solution. The DDPG model efficiently identified 'Beneficial and Starving' services, alleviating the starvation issue by deploying twice as many low-priority services. With an accuracy rate exceeding 80%, our online RR model quickly learns prioritization patterns in under 100 transitions. We categorized services as 'High-Demand' (HD) or 'Not So High Demand' (NHD) based on traffic volume, providing insight into high revenue-generating services. We achieved a nearly optimal resource allocation by balancing low-priority HD and low-priority NHD services, deploying twice as many low-priority HD services as a model without traffic awareness.

INDEX TERMS 6G, Machine Learning, Resource Allocation, Network Function Virtualization, Microservices

I. INTRODUCTION

ALTHOUGH the fifth generation of mobile networks (5G) is currently providing essential support for the Internet of Everything (IoE) and Ultra-Reliable Low Latency Communications (URLLC), it definitely raises the question of whether current 5G systems can handle advanced applications such as Digital Twin (DT), connected robotics, autonomous systems, Augmented Reality (AR)/Virtual Reality (VR)/Mixed Reality (MR), and Blockchain and Trust technologies [1], [2]. These emerging applications are expected to make demanding service requests, such as those for high reliability, ultralow latency, and substantial data rates [1]. As a result of this debate, significant research progress towards sixth generation of mobile networks (6G) has been made. 6G must be designed to support upcoming service types like Computation Oriented Communications (COC),

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

Contextually Agile eMBB Communications (CAeC), and Event Defined uRLLC (EDuRLLC) in addition to enhanced Mobile Broadband (eMBB), URLLC, and massive Machine Type Communications (mMTC) services [3]. The network architecture must be improved to provide adequate support to support these emerging and rapidly evolving services.

The Network Function Virtualization (NFV) framework virtualizes the Network Functions (NFs) by enabling them to run as softwarized NFs (i.e., Virtual Network Functions (VNFs)) on commodity hardware, releasing them from their dedicated proprietary substrate appliances. This gives VNFs the freedom, flexibility, and agility to switch between servers in response to dynamic variations in resource demand. Although NFV is a promising technology, it can be difficult when multiple applications coexist and the supporting infrastructure must guarantee resources for all Service Function Chainings (SFCs). This is the NFV-Resource Allocation (RA) problem, which is categorized as NP-Hard. Due to the affinity and anti-affinity[1] requirements, the complexity of this type of NFV-RA increases, making it more difficult to update software and perform routine maintenance. Despite the increased complexity of design and deployment, to facilitate the transition to 6G, a number of actions are being taken on the access and network sides. On the network side, the NFV architecture [4], introduced in 2012, is pushing towards a microservices-based architecture [5], [6]. NFV based on microservices [7], [8] is a standard software design paradigm that decomposes monolithic VNF into several individual microscale and independent VNFs. These micro-VNFs are loosely coupled software codes, providing the advantage of easy maintenance, such as upgrading and scaling of micro-VNF. In other words, dynamic scaling of VNFs is possible. Due to their independence, testing and migration become manageable and allow the reuse of the micro-VNFs, this plays a vital role for efficient resource allocation. embedding problem introduces notable advantages; however, it also brings about associated costs and constraints. In particular, the deployment and architectural complexity escalate as micro-functionalities, resulting from the deconstruction of monolithic VNFs, demand additional resources such as bandwidth and latency. The process of breaking down monolithic VNFs leads to the creation of redundant *m*VNFs, replicating standard functionalities. These redundant *m*VNFs contribute to increased processing overhead and unnecessary resource consumption, causing wasted CPU cycles. To address this limitation, a rearchitecture of the decomposed VNF-Forwarding Graph (FG) is essential. Criteria are required to identify the suitable VNFs for decomposition in each service. Furthermore, a granularity criterion is needed to break down identified VNFs into microservices based on network resources. This cutting-edge 'NFV-RA + Microservies' strategy promises to provide a solution that is (theoretically) closer to optimal, as demonstrated in [9]. Our work in [9] provided a detailed analysis of this approach, the criteria to identify the potential VNFs for dynamic decomposition and its granularity based on the conditions of the network. Section IV provides an overview of it.

This work performs a deeper analysis of the criteria for dynamically prioritizing online SFCs, the merits of embedding an SFC over others, and the importance of traffic and admission control. Until now, most of the research has focused on the deployment of the arrived SFCs and their VNFs according to predefined objectives. It was typically assumed that the scheduling of these online SFCs follows a First-in-First-out (FIFO) approach. In other words, various proposed models dynamically identify network placements for incoming SFCs as they arrive without rearranging the SFC waiting queue according to a rational strategy. Our main emphasis is on the significant impact of maintaining an efficient scheduling queue for incoming SFCs, an aspect that has been neglected in prior research. In reality, not all SFCs are of same type or belong to the same priority group. Therefore, it cannot be considered equally [10]. When practicing the traditional method, FIFO accepts the scheduling (waiting) queue as it is, neglecting to recognize and prioritize emergency or time-sensitive SFCs, which requires earlier deployments over non-emergency SFCs. This approach may lead to resource exhaustion on the network, as resources are already allocated to non-essential SFCs, resulting in the rejection of highly critical (emergency) SFCs. This demonstrates an inefficient resource allocation method for SFCs. Research work such as [11], [12] identified the problem and tried to solve the problem at a high level. In their approach, the SFC deployment was purely based on the priority of SFCs, assuming that this priority is already known. Previously, services were distinct, less complicated, and rarely evolving; therefore, having a rigid priority and manual classification was reasonable. Considering the future services' dynamically evolving service types and time-varying resource requirements, this traditional inflexible method is inefficient. To provide adequate support for assigning dynamic priority, one must discover the hidden pattern, which requires more information about the requested service. Dynamically predicting the type of SFCs and their priority level will be a challenge. Thus, we need a model to perform intelligent and dynamic prioritization based on the currently requested Quality of Service (QoS) and other attributes. From the literature, considering the static prioritization approach might diminish the drawback of the FIFO to an extent, it brings to light a few other essential factors to be considered that were ignored previously. To start with, the introduction of 'starvation for lower priority services' caused a significant bias in the system - an unfair design for users. Because high-priority services are always preferred, low-priority services must wait a longer period of time. This results in complete rejection of the low-priority service, either due to exhaustion of nodal resources or waiting in the queue beyond the upper limit [2]. To avoid this bias,

[1]Affinity and anti-affinity constraints refer to the ability to embed a VNF on a particular node (affinity) and the converse of it (anti-affinity)

[2]The threshold waiting time per service.

vital SFCs should be categorized according to other essential characteristics, rather than only priority levels, which many researchers have overlooked. This calls for an intelligent fair system that will take into account other factors (such as the wait time in the upper limit and the demand for reliability) and provide an optimal solution to the service schedule.

In 6G-and-beyond, considering the time-varying traffic, it is anticipated that the processing resource requested from a VNF will fluctuate based on the traffic arrival statistics and the QoS requirement [13]. The amount of processing traffic for each service changes regularly, i.e., the number of users or clients requesting the services changes, as does the produced traffic load. For a network to operate reliably and effectively, it is necessary to understand the time-varying network traffic. Network administrators can optimize network performance by ensuring that enough resources are available to meet user demands and reduce the risk of network failures by monitoring and managing network traffic. Thus, in this work, we have integrated the traffic metric to see the impact on the proposed priority, scheduling, and deployment model.

In general, in this work, we propose a zero-touch solution to provide seamless support for more realistic services for future networks. The online-Ridge Regression (RR) model and Deep Deterministic Policy Gradient (DDPG) approach, respectively, are applied to the proposed 'Dynamic Prioritization' (DyPr) and an 'Adaptive Scheduling' (AdSch) module to reduce the 'Starvation' situation and offer a fair scheduling principle. The starvation problem is further investigated on the basis of the traffic load generated by the deployed services. We trained the classifiers to identify the services as High-Demand (HD) and Not-so-High-Demand (NHD) based on the traffic load. The objective of our suggested DDPG-based algorithm is to understand the importance of embedding the 'Beneficial SFC' or 'Starving SFC' over others. Furthermore, the literature does not provide the criteria and procedures to determine the priority of the online service; instead, a static approach has been used. Therefore, using the ML method, we have first developed a standard to dynamically assess the priority of a SFC. Subsequently, our DDPG-based model was trained to rank the arriving SFCs according to urgency, reliability, achieved priority, traffic class, and other QoS attributes. The SFC's deployment is rescheduled based on the rank obtained. From the rearranged queue of SFCs, each deploying SFC must pass through the admission control module, which grants permission or denies deployment based on the amount of time SFCs waited in the queue. In summary, the goal of this work is to diminish the 'low starvation problem. To do so, our contributions are as follows:

- Developed an ML-based traffic-aware system to label the services based on the load generated. A thorough examination of various clustering and classifier models was conducted,
- Explained the existence of the starvation problem in the existing methods and their drawbacks,
- Established a dynamic priority estimation criteria for all the online services using the RR model,
- Developed a traffic-aware DDPG-based framework for recognizing and understanding the importance of preferring an SFC (say, 'Beneficial and Starving SFC') over others,
- Established an admission control approach, based on the SFC's waiting time in the queue before being deployed, and finally,
- Adopted the traffic-aware deep Q-Learning (QL) model along with microserives concept for VNF embedding.

II. Literature Review

This a gentle reminder to the reader that the main focus of this work is on service prioritization and the impact of traffic on the scheduling technique, which is integrated with the service deployment. As a result, the following literature focuses primarily on service prioritization, scheduling style, and the role of traffic in prioritization.

A. Prioritization & Scheduling Style

NFV has received a lot of attention for meeting Service Level Agreements (SLAs) for next-generation networks. Most researchers [14]–[19] have proposed models based on Machine Learning (ML), heuristic or metaheuristic for dynamic deployment of arriving services. These works concentrated exclusively on the FIFO scheduling scheme for the deployment of SFCs. However, they ignored the importance of understanding that not all SFCs fall under the same priority group and cannot be addressed equally [10]. The main limitation of traditional method, FIFO is it failed to recognize and give emergency SFCs higher preference than the non-emergency ones (such as time-sensitive services), causing rejection of highly critical SFCs. This pattern has been demonstrated in Figure 5. Therefore, following the FIFO technique can cause failure in the deployment of emergency services over those that are able to do the best effort when needed. To overcome the limitation, the authors in [11] and [12] classified the SFCs mainly as Premium (Pr) or Best-Effort (BE), yielding only two cardinalities. Methods that classify SFCs as either Pr or BE are constantly prioritizing Pr services over the BE ones, resulting in severe service deprivation for low-priority services [11]. Work in [20] and [21] demonstrated the effectiveness of mapping services according to the priority of the VNFs. The authors of [20] discuss three types of priority: per-service, per-VNF, and per-flow, where service prioritization is based only on the delay attribute only and per-VNF priority is randomly assigned. This is an iterative process that adjusts the solution after each iteration, which is not scalable considering the requirements 6G-and-beyond requirements. Furthermore, the predefined number of flows that a VNF can handle ignores the operational dynamics. The authors of [22] described three priority levels (categories A, B, and C) in descending order according to commercial value or

related SLA. In [23], the authors concentrated on each SFC's QoS or considered the selection of fixed priority-based SFCs, whereas in [24], the placement of SFCs occurred, based on the high-to-low service priority. So far, the authors have not considered rational strategies for service prioritization, instead assuming that the services arrived with the pre-allocated values. However, the authors of [25] proposed priority-aware criteria, classifying the priority based on the VNF's resource demand and assigning it to the highest or lowest priority list, which helps sort the batch requests. The method of scheduling/ordering of deploying the services or serving the traffic according to the priority is called the 'Priority-based' method. The authors in [26] calculated the priority based on the CPU and bandwidth requirements of the services as well as the percentage of node and link resources that are affected by node failure. The authors mainly focused on the VNF migration assuming that SFCs have already been deployed. As noted in [14]–[19], despite the deployment model relying on machine learning or heuristics, the scheduling of SFCs adheres to a FIFO approach. Attempting to tackle this limitation, earlier studies like [11], [12] , and [20]–[25] have put forth priority-based approaches, often resulting in a starvation problem for low-priority services. Nevertheless, none of these studies have presented an end-to-end model incorporating dynamic prioritization and an adaptive scheduling scheme suitable for dynamic network conditions and evolving service demands. This sets it apart from static or rule-based scheduling approaches which are considered as baseline models.

Our proposed algorithm integrate adaptive scheduling, dynamic prioritization, and various ML models like Supervised, Deep Reinforcement Learning (DRL), setting it apart from conventional approaches:

- **Adaptive Scheduling and Dynamic Prioritization Integration:** Unlike traditional DRL methods focusing solely on optimizing the deployment of SFCs in the FIFO method, our algorithm incorporates adaptive scheduling and dynamic prioritization mechanisms. By considering the evolving priorities of services and adapting scheduling decisions based on real-time changes in network conditions, our approach ensures efficient resource allocation while addressing the dynamic nature of service demands. This integration enables our algorithm to make more informed decisions that reflect the varying importance and urgency of different services over time.
- **Fine-Grained Service Differentiation:** While conventional DRL-based or other models may treat all services as equal entities, our algorithm introduces a multilevel multiclass priority framework, allowing for fine-grained differentiation between services. By categorizing services into multiple priority levels and classes, based on factors such as criticality, waiting time, and traffic load, our algorithm captures the nuanced requirements of diverse applications. This granularity enables more precise and context-aware decision-making, ensuring optimal resource utilization and service quality.
- **Consideration of Operator Benefits:** In addition to optimizing resource allocation for service quality, our algorithm incorporates the strategic objectives of service operators. By factoring in the anticipated revenue potential associated with different services, our approach aims to maximize operator benefits while maintaining network performance. This consideration goes beyond traditional model objectives and aligns with the broader interests of stakeholders, contributing to more sustainable and economically viable network management strategies.
- **Dynamic Adaptability and Learning:** Our algorithm's incorporation of DRL techniques enables dynamic adaptability and continuous learning from interactions with the network environment. By iteratively updating policies based on feedback from ongoing operations, our approach can effectively respond to changing network conditions and evolving service requirements in real time. This dynamic adaptability enhances the algorithm's robustness and resilience in dynamic network environments, distinguishing it from static or rule-based scheduling approaches.

B. Role of Traffic

In networking, the amount of data or information travelled from the source to the destination node within a network is referred to as traffic/workload. Depending upon the applications, various network traffic exists, such as data packets, voice calls, video streams, and so on, which are transmitted among the connected devices. That is, each network service (like web browsing) has its own communication patterns and data requirements, resulting in diverse traffic characteristics. Web browsing traffic, for example, consists of requests for online pages, whereas file transfer traffic consists of the transport of large files between devices. Voice and video traffic require real-time transmission to enable smooth collaboration. A network's load might change depending on variables, including the time of day, typical network traffic patterns, and the kinds of applications or services used. According to Ericsson's 'Mobile data traffic outlook' report [27], by the end of 2022, overall mobile network traffic will reach about 115 EB per month and 453 EB per month by the end of 2028 when FWA is included. In the forecasted traffic growth up to 2028, it is assumed that Extended Reality (XR)-type services, such as AR, VR, and MR, will begin to be used. Considering this prediction, one can articulate the significance of understanding the traffic. Thus, network administrators should be aware of the network's workload since it may be used to plan for network capacity, assign resources, and ensure the network can support its traffic demands. This forecasted unprecedented growth in network traffic had put tremendous pressure on telecom operators to manage and optimize their networks in order to

meet the growing demand for ultra-high-speed and reliable connectivity.

In order to increase network effectiveness, network quality, load balancing, and energy conservation in networks, traffic clustering, forecasting, and maintenance are essential criteria [28]. This traffic complication further increases for the denser heterogeneous architecture, particularly in 6G-and-beyond networks. When deploying VNFs, it is beneficial to consider traffic characteristics. For example, deploying VNFs corresponding to the traffic load can prevent processing and transmission resources from being over-or underutilized. The imbalanced network load can degrade service and network performance. Thus consideration of traffic load for VNF placement is a must.

As mentioned, various types of network traffic exist, and identifying and categorizing network traffic accurately is vital [29]. Researchers have considered many classification approaches. A fundamental strategy is port-based. It is a technique for classifying network traffic based on the port numbers in the packets' communication protocols. Each application or service is assigned a special port number, and network administrators can identify the type of traffic by inspecting these port numbers in packet headers. Nevertheless, as newer applications run on dynamic ports, a port-based strategy is no longer used due to inadequate classification results [30]. Deep Packet Inspection (DPI) is also a widely used approach; regardless, it has some drawbacks. For example, only applications with available patterns are the only ones that DPI can detect; it is computationally highly expensive and does not perform adequately for encrypted data. ML-based strategies have been implemented to address these shortcomings. The authors of [31] stated, "Traffic classification is an important network function, which provides a way to perform fine-grained network management by identifying different traffic flow types.". This will aid in effective resource allocation for diverse services. ML-based traffic classification for resource allocation is mentioned in [32]–[35]. In [29], the authors adopted K-means and DBSCAN algorithms for clustering the traffic, assuming four classes exist.

An effective way of deploying the VNFs over the infrastructure is to consider an elastic allocation (dynamic provisioning and scaling) of the resources according to the VNF service requests (their requirements) and the workload/traffic. This elasticity is achieved by Horizontal scaling and Vertical scaling of the resources. Installation and removal of VNF instances are considered horizontal scaling, whilst allocating and releasing host and bandwidth resources to and from a VNF instance are considered vertical scaling [36]. Actively migrating the VNFs or re-distributing the workload among the VNFs also achieves elasticity. Though the introduction of elasticity will benefit future networks, however, there are expenses associated with releasing and allocating resources elastically [37]. Thus, the main challenge is determining which elasticity approach to adopt according to the workload or the demand. Below are some approaches adopted by the researchers.

In [38], the researchers proposed the Least-First-Greatest-Last (LFGL) approach to solving the traffic-aware deployment of middle-boxes, assuming there is only one flow at a time. The authors did not acknowledge the interdependence of the middleboxes, which is one of the critical constraints. To overcome this, the authors of [39] integrated interdependency relations among the middle-boxes or VNFs, where different scenarios have been considered like the placement of middle-boxes when the flow path is predetermined or not if the middleboxes are non-ordered/partially ordered/totally ordered. The problem has been solved using the heuristics models. The traffic not only aids in middlebox resource allocation but also in the scaling and migration of VNFs. The authors in [36] defined an Elastic Virtual Network Function Placement (EVNFP) problem and proposed a Simple Lazy Facility Location (SLFL) method for optimizing VNF placement according to an on-demand workload; similarly, the authors in [40] considered traffic load along with the QoS of the service to deploy the VNFs on the network. Over here, the authors define a traffic engineering framework, a multi-objective mixed integer optimisation problem. A time-efficient heuristic model is proposed to solve the problem with feasible computation expense. The authors suggested a change-point-driven traffic parameter learning and resource demand prediction method [13]. Based on this information, a deep Q-learning technique is used to create dynamic VNF migrations. The authors proposed a traffic prediction model based on deep learning for scaling the geo-distributed VNF chains based on the discovered patterns by Deep Learning (DL). And for the deployment of the VNFs, a DRL model has been suggested [41].

Most of the researchers followed the same steps. Initially, the deployment of VNFs is performed based on the QoS by heuristic or ML-powered approaches. Later, based on the flow rate/traffic, the VNF migration or horizontal or vertical scaling is triggered. This approach is attainable for smaller action spaces; with the increase in complexity, the section of VNF/Virtual Machines (VMs) for scaling generates high complexity, resulting in high computational time and poor coverage performance. Thus, this is an open issue. To address this, we proposed a model that classifies whether online services are HD or NHD, providing sensitive information if future migration/scaling or duplication is required. This classified information is provided to the embedding model for effective service placement, which will benefit in reducing the scaling/migration computational time and also diminish the starvation problem (detailed explanation is provided in Section A.

III. System Modeling and Technical Background

The physical topology is modelled as a directed graph $\mathbf{G} = (\mathbf{H}, \mathbf{N})$, where $\mathbf{H} = [\mathbf{h}_0, \ldots, \mathbf{h}_{|H|-1}]$ and $\mathbf{N} = [\mathbf{n}_0, \ldots, \mathbf{n}_{|N|-1}]$ represent the set of physical nodes (say

high-volume servers) and physical links of the topology, respectively. $|H|$ and $|N|$ represents the total number of physical nodes and physical links of a topology. Each physical node is represented as $h_{x'}$, where x' ranges from 0 to $|H|$. $\mathbf{h}_{x'} = [h_{x',0}, \ldots, h_{x',J_{node}-1}]$ can be expressed as an array of amount of available nodal resources like CPU core, RAM, etc, for each physical node $h_{x'}$. For example, $h_{0,0}$ provides information about the CPU cores availability on physical node 0. J_{node} is the number of nodal resource types indexed from $0, 1, \ldots, J_{node} - 1$. Similarly, $n_{y'}$ represents the physical link on the topology, which ranges from 0 to $|N|$. Each physical link can be expressed in an array indicating the availability of the link resources, i.e., $\mathbf{n}_{y'} = [n_{y',0}, \ldots, n_{y',J_{link}-1}]$ signifies the link resources such as bandwidth, latency, etc. For example, $n_{y',0}$ is available bandwidth resource for link $n_{y'}$. J_{link} stands for the number of link resource types, the resource available for physical links.

The VNF-Forwarding Graph (VNF-FG) or SFC (Ψ) is represented as a directed graph $\mathbf{G}'_{\Psi} = (\mathbf{V}_{\Psi}, \mathbf{B}_{\Psi})$ with the set of VNFs ($\mathbf{V}_{\Psi} = [\mathbf{v}_{\Psi,0}, \ldots, \mathbf{v}_{\Psi,|V|-1}]$) and Virtual Links (VLs) ($\mathbf{B}_{\Psi} = [\mathbf{b}_{\Psi,0}, \ldots, \mathbf{b}_{\Psi,|b|-1}]$) for each SFC Ψ that provides an end-to-end service. Deploying these graphs onto the physical topology is termed the VNF-FG Embedding (VNF-FGE) problem. The total number of VNFs and VLs in an SFC is denoted as $|V|$ and $|B|$, respectively. Each VNF and VL comprises a set of requested resources such as the CPU core, delay, bandwidth which is represented as $\mathbf{v}_{\Psi,x} = [r_{\Psi,x,0}, \ldots, r_{\Psi,x,J_{node}-1}]$ and $\mathbf{b}_{\Psi,y} = [l_{\Psi,y,0}, \ldots, l_{\Psi,y,J_{link}-1}]$, respectively. The computing resource initialization in most related works is done in a random manner, disregarding the high correlation between the CPU core and RAM. This work outlines the link between the CPU core and RAM as in [42]. In addition, we considered delay, jitters, packet loss, reliability, and threshold waiting time as some of the QoS attributes. Moreover, we also included the predicted epoch-based aggregated traffic load per SFC (by different users) and the total load of arriving SFC for its entire life cycle. Thus making the $J_{node} = 8$ in our case. Similarly, $J_{link} = 2$ in our studies, considering latency and bandwidth. All the notation for this proposed work is described in Table 1.

A. Traffic-aware Service

Traffic patterns and characteristics play a vital role in organizing SFC scheduling queues. When the incoming traffic load surpasses the link bandwidth or exceeds the processing resources allocated in Commercial-Off-The-Shelf (COTS) computing hardware, it necessitates scaling or migration, incurs additional elasticity costs.

One method for lowering this cost is by predicting arriving workload of the services and deploying them to the hosts accordingly in such a way that it prepares the network to satisfy future demand with lesser migration and scaling costs. Given that the traffic and the VNF's processing resources are both time-varying, learning from historical data will be advantageous in this scenario.

1) Overview of Traffic-aware Service Classification

To do that, first, we need to observed the network dynamics and gathered valuable information, like the rate of SFC arrival (its type and QoS), deployment of VNFs, the network traffic generated by the successfully deployed SFCs and the time-varying processing of resources. This process happens in 'Traffic Gen.' repository module as shown in Figure 1. Using these collected observations, the classifier model is later trained to distinguish between highly demanding (popular) and not-so-demanding services among users or clients. Figure 1 illustrates proposed approach for clustering and classification of the online services based on the traffic load. This classified information will be fed to the scheduler and deployment modules to provide an optimal placement and scheduling solution to reduce migration or scaling costs.

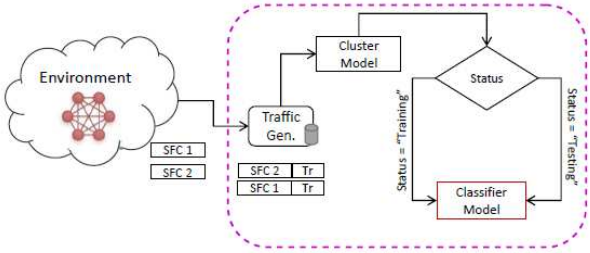

FIGURE 1. Process for Training the Classifier

2) Clustering and Classification Models

In our proposed work, the arriving SFCs are classified into two classes: 'HD' services and 'NHD' services, by a binary classifier developed using supervised learning. The main limitations are the constant change in load and the unavailability of labelled data to train the supervised models. To overcome that, we clustered the collected dataset into two classes using models like K-means, Agglomerative clustering, Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH), and Gaussian Mixture (GM), turning the unlabelled data into labelled. Later, these labelled datasets are provided to the supervised models, considering the labels as the target value, which helps train the models. We have opted for supervised learning methods like Logistic Regression (LR), Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), and K-Nearest Neighbour (KNN) to perform a deep analysis of the models. Accuracy, Recall, Precision, Baseline accuracy (Random Rate classifier), and Runtime are some metrics used to investigate the model's performance. Algorithm 1 describes the classification of traffic-aware services. Line 1-2 defines the collection of the

Notations	Descriptions
G	Directed graph for topology
H	Set of substrate nodes
N	Set of substrate links
J_{node}	Total nodal resource types
J_{link}	Total link resource types
G'	Directed graph for VNF-FG
V	Set of VNFs
B	Set of VLs
ε	Epoch
H	Hurst exponent
$\boldsymbol{Dpd}$	Array of deployed SFCs
$\boldsymbol{AgtL}$	Epoch-based Aggregated Traffic Load
$L_{\Psi,u'}$	Traffic load produced by each user or client (u') per SFC (Ψ)
$totLoad_{\Psi}$	Total traffic load by an SFC Ψ for its life-cycle (b)
Ξ'	Admission Controller
Υ	Beneficial-cost
Φ	Starvation-cost
$n(\cdot)$	Normalized function
ζ_{Ψ}	Starvation factor
Γ_{Ψ}	Reliability for SFC Ψ
P_{Ψ}	Priority for SFC Ψ
T_{Ψ}	Threshold Waiting Time for SFC Ψ
$R(\Psi)$	Reward function for SFC Ψ
Δ	Waiting Period
Ar_y	Available resources in the node
Ir_y	Initialized resources of a node
Cl_y	Classified SFC
R_{rel}	Rewards based on the reliability of the SFC
$R_{priority}$	rewards based on priority of the SFC
$R_{placement}$	Reward based on the placement of the SFC
$R_{quality,y}$	Reward based on quality of the selected node (y)

TABLE 1. Notations

datasets (SFC and their traffic load) from the environment until the desired datasets are collected for better training. Once the enough samples are collected in the repository the datasets are ready for ML models (Lines 3-4). Lines 5-10 initialization of clustering and classifier models. The entire data set is clustered, and the unlabelled services are labelled as in lines 11-15. When the clustering is performed successfully, the status is modified to classification (Line 17), where the entire labelled dataset is split into testing and training (Line 18-20). Once the training is over, the classifier models are tested (Lines 21). These models are trained and tested offline by collecting the data from an environment without disrupting the ongoing deployment process. These trained models were then placed in the network in such a way that based on the online SFC and its features and the traffic loads; the model will classify them in either the HD or NHD classes. This achieved information is fed to the subsequent modules (i.e., adaptive scheduling and dynamic deployment) for optimal resource allocation on the network.

3) Non-stationary Traffic Generation method

The clustering or/and classification of SFCs depends not only on the QoS but also on the total load that an SFC can bring over the course of its life-cycle and the number of packets that arrive during a specific time epoch. The network traffic is considered non-stationary due to the constant fluctuation in the workload. A non-stationary traffic trace is assumed to be a concoction of various series of stationary traffic segments with unknown change points. And each stationary traffic trace represents the aggregated traffic load induced by a deployed SFC requested by various users. Figure 2 shows an example of a non-stationary traffic trace for an epoch (considering the epoch is of 6-time units), where at different time-steps, each deployed SFC produced some aggregated traffic (packets per time-unit).

For example, an SFC 30 is under deployment process (we assume the time taken by an SFC to be deployed as an epoch) while SFCs 1, 5, 32, and 67 have been previously deployed on the network. The deployed SFCs (1, 5, 32, and

Algorithm 1 Traffic-aware service classifier

Store the arrived SFCs and its traffic load from the Environment in the repository till the threshold limit is reached
Datasets (Dt): Retrieve deployed services from the repository for training
Initialize clustering (K-means, Agglomerative clustering, BIRCH, and GM)
Initialize classifier models (LR, SVM, Decision Tree (DT), RF, and KNN)
Initialize the number of cluster, i.e., 2
Initialize the number of class sizes, i.e., 2
if *Status = 'Clustering'* **then**
  Unlabelled Dt is given to clustering models
  Using k-mean, Agglomerative, BIRCH, GM
  Achieve labels (L)
  Allocate labels to corresponding transitions ($(sfc : L)$)
else
  Status = 'Classification'
  Labelled Dt is split into $Dt_{Training}$ and $Dt_{Testing}$
end
Using $Dt_{Training}$ train Decision Tree, Random Forest, Support Vectors, K-nearest neighbor model
Using $Dt_{Testing}$ tested the models
Estimate Accuracy, Precision, Recall, Baseline accuracy

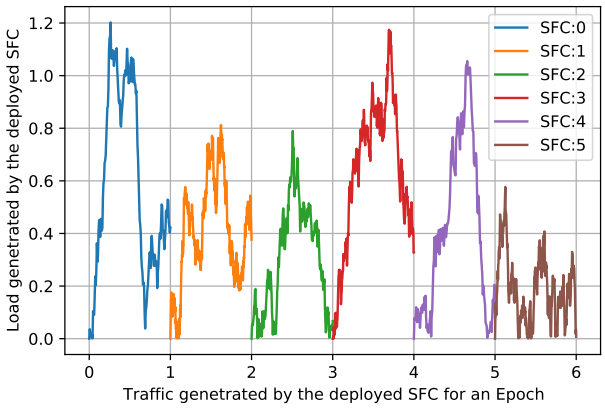

FIGURE 2. An example of a non-stationary network traffic generated during an epoch

67) will start generating the traffic load on the network as per the users' requests until SFC 30's placement process is over. Each deployed SFC's traffic trace will be identified as stationary traffic, and the combination of all (1, 5, 32, and 67) SFCs for that epoch is recognized as a non-stationary traffic trace.

Equation (1) represents the array of deployed SFCs ($\boldsymbol{Dpd}$) for an epoch (ε) and (2) describes an array of epoch-based aggregated traffic load ($\boldsymbol{AgtL}$) by the deployed SFCs. For example, SFC $\mathbf{G}_0$ has generated $AgtL_0$ amount of traffic load for an epoch, this provides information on how resource demanding this SFC can be for a period of time. This epoch-based aggregated traffic load ($AgtL_\Psi$) for an SFC ($\mathbf{G}'_\Psi$) is the sum of traffic load ($L_{\Psi,u'}$) produced by each user or client (u'), as in (3). The epoch-based aggregated traffic load ($AgtL_\Psi$) observes a Gaussian distribution, and the overall total load achieved by an SFC $\mathbf{G}'_\Psi$ during its life-cycle (b) is represented as ($totLoad_\Psi$) as in (4).

$$\boldsymbol{Dpd}(\boldsymbol{\varepsilon}) = [\mathbf{G}_0, \ldots\ldots, \mathbf{G}_i] \tag{1}$$

$$\boldsymbol{AgtL}(\boldsymbol{\varepsilon}) = [AgtL_0, \ldots\ldots, AgtL_i] \tag{2}$$

$$AgtL_\Psi(\varepsilon) = \sum_0^{u'} L_\Psi^{u'} \tag{3}$$

$$totLoad_\Psi = \sum_0^{b} AgtL_\Psi^{b} \tag{4}$$

$$E = E_t, t \geq 0 \tag{5}$$

$$\Gamma_E(t,s) = \frac{1}{2}(t^{(2H)} + s^{(2H)} - |t-s|^{(2H)} \tag{6}$$

To model network traffic for each SFC, we used the Fractional Brownian motion (fBm) (E) technique. E_t represents the fBm at time-step (t), as in (5). fBm is a continuous-time Gaussian process with self-similarity and Long-Range Dependence (LRD) stochastic properties, which correlate to real network traffic characteristics [43], [44]. In (6) H is a Hurst exponent, which has values ranging from 0 to 1, controls the self-similarity attribute of fBm, or, say, the degree of correlation of the process. When $H = 0.5$, the long-term correlation in the process is absent, which acts like a classic Brownian motion, the process is highly correlated if $H > 0.5$, and vice versa for $H < 0.5$. Thus, the Hurst parameter is vital in defining each SFC's network traffic trace (process). For our experiment, due to the high complexity and immense amount of constantly changing variables, we have adopted $H = 0.7$.

B. Dynamic Prioritization

A major drawback of commonly-used methods of binary classification of services into Pr and BE is that the predetermined priority levels for the arriving services are randomly allocated, and the co-relation between the service requirements and priority is ignored. In a realistic scenario, the allocation of service priority should be based on various factors of QoS, flow type, etc. Determining the priority levels for the existing or well-aware services is trivial. The complexity induces when future communications systems expect frequent unseen 'short-lived' services with instant placement requirements. To fulfill this need, an intelligent DyPr model is anticipated rather than a static or conventional method.

In the DyPr model, we investigate the relationship between the requested QoS factors of the arriving service and, based on it, a dynamic priority level is assigned. This problem

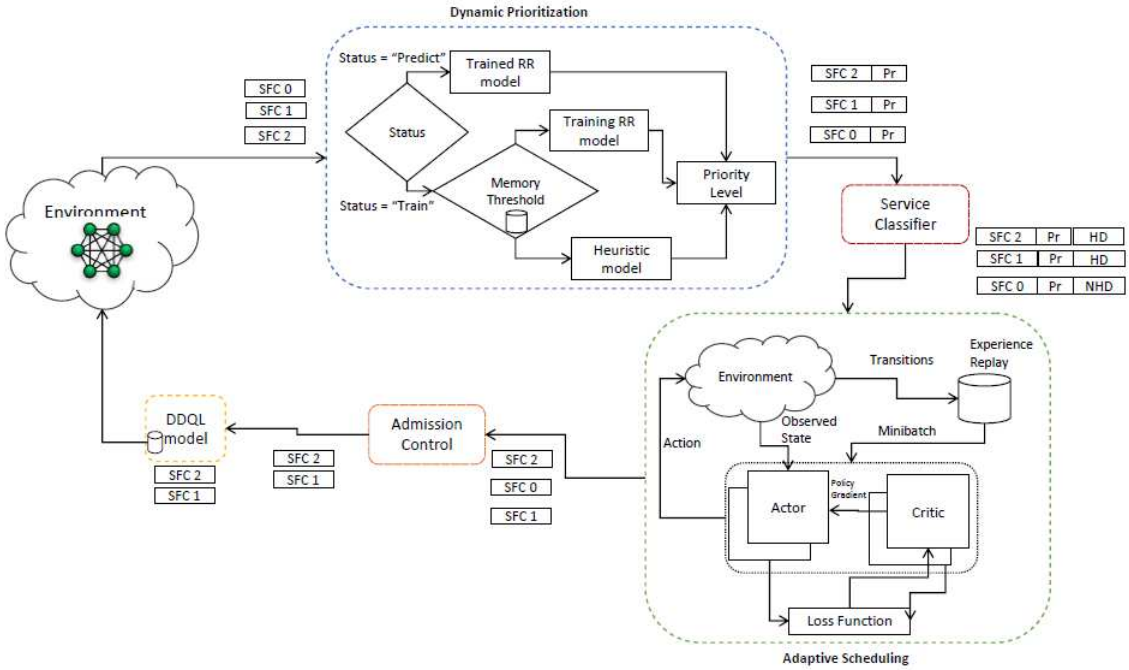

FIGURE 3. Overview of the Proposed model where the services are dynamically prioritized, classified as HD or NHD, based on the metrics adaptive scheduling occurs for efficient deployment

is viewed as a multiple regression task with various independent variables $A[0] \dots A[p]$ (like QoS factors with p as the number of factors), to predict a continuous dependent variable Y' as referred to in (7). Here W and B are the regression coefficient and residual term, respectively, which are learned during the training. The adopted model discovers the linearity between the dependent and independent variables. In our case, these independent variables are threshold jitters, delay, and packet loss, which a service can tolerate, and the dependent variable denotes the appropriate priority level. However, the multiple regression data suffer from multi-colinearity, which causes the estimation of regression coefficients to be inaccurate. As a result, its existence reduces the model's performance by causing the predicted value to differ significantly from the actual value. One way to diminish this is by adopting the RR model, which performs L2 regularization, due to which the model is more restricted and causes less over-fitting [45]. The objective is to minimize the cost function mentioned in (8), where M is the number of samples (services per run), Y_Ψ is the actual value. The λ (penalty term) regularizes the coefficients in such a way that the optimisation function is penalized if the coefficients assume high values.

$$Y'_\Psi = W[0] \times A_\Psi[0] + \cdots + W[p] \times A_\Psi[p] + B \tag{7}$$

$$\sum_{\Psi=0}^{M} (Y_\Psi - Y'_\Psi)^2 = \sum_{\Psi=1}^{M} (Y_\Psi - \sum_{j=0}^{p} W_j \times A_{\Psi,j})^2 + \lambda \sum_{j=0}^{p} W_j^2 \tag{8}$$

The RR model is supervised learning, which uses pre-existing datasets for training purposes. Our study evaluates the model's performance for future unseen services; hence, we assume that no helpful service information is available to us. The traditional RR model would not provide enough support due to the lack of sufficient datasets. To support the dynamism, we have modified it by adding an observation phase before the learning phase.

The observation phase plays a crucial role in accumulating valuable information about online services over time, even without prior knowledge. This data is vital for training the model, enabling it to make online decisions. Over here, the model saves (observes) the online services in a memory buffer until minimal transitions are achieved to begin the training phase. During this time, the allocation of priority is carried out uniformly. This helps in constructing our training datasets and making an 'Online-RR' learning model rather than based on offline learning. Later, once the saved transition threshold is surpassed, the learning phase starts, which uses the current and saved transitions for model training. These saved transitions (batch transitions) are selected randomly from the memory to avoid any chance of over-fitting (co-relation between the transitions). The model's accuracy is checked periodically. Once the model reaches the desired accuracy, the trained model state changes from 'Train' to 'Predict'. The overview of DyPr is shown in Figure 3. Moreover, this model was chosen after thoroughly evaluating model performances using several methodologies, including Artificial Neural Network (ANN) and Lasso. Of all the models, the online-RR model outperformed expectations. This model is constructed with an envision to support

future unseen services by introducing a zero-touch cognitive system.

Traditionally, the services are categorized into two cardinalities, which diminishes the value of services relative to one another within a class. Most research has ignored the intra-class priority aspect; however, given the future demands, it should be considered. To overcome this, a service priority is ranked between 0.0 to 1.0, with 0.0 representing the least essential service, and 1.0 being the most crucial. Each priority level is expressed in macro-class priority and micro-class priority. While the micro-class priority establishes the priority status inside a class, the macro-class priority categorizes the class to which it belongs. For example, the priority of SFC A and SFC B is 0.79 and 0.72, respectively. Though both the SFCs belong to the same class (i.e., macro-class is 7), SFC A with a micro-class as 9 will be prioritized over the SFC B with a micro-class as 2, due to the higher value. This delivers more details about the importance of services within the class, which is beneficial, especially for time-sensitive or critical applications and for the adaptive scheduling phase. Therefore, in our work, the DyPr is considered as a regression model rather than a classification model.

C. Adaptive Scheduling

Conventionally, high-priority services are preferred over others, leaving the low-priority services to wait for extended periods of time, resulting in their starvation. This highlights the second drawback of the conventional binary classification of services. A biased scheduling system raises questions like, 'Can the priority attribute be the most effective scheduling decision-making factor?' 'Will the decision be more optimal by considering additional factors, such as service waiting time or reliability?' In a search for an unbiased scheduling system, we have proposed an intelligent Adaptive Scheduling module using the DDPG approach. This approach weighs more than one factor, resulting in a more efficient solution compared to traditional methods. Let us say that a high-priority SFC A has a higher waiting threshold than a lower-priority SFC B (which is about to expire soon). According to the traditional model, SFC A will be selected, ignoring the SFC B to expire, affecting the Quality of Experience (QoE). However, our model, which considers more than one factor, prefers SFC B over SFC A, understanding that there will not be any negative impact on SFC A if it waits in a queue a little longer, providing scheduling optimality.

DDPG is an actor-critic Reinforcement Learning (RL) model, trained to identify 'Beneficial and Starving' services based on 4-factors: Threshold Waiting Time (TWT)(T), Reliability (Γ), Priority (P) and Traffic classifier (Cl_Ψ) of the service. The state-space for the requested service Ψ is represented as $(T_\Psi, \Gamma_\Psi, P_\Psi, Cl_\Psi)$. Based on these factors, the DDPG agent determines a rank for each service which is in range of 0 to 1 (i.e., action-space (A_Ψ)), indicating the significance of deploying the service. With a higher rank, the necessity to deploy the service is significant, resulting in a rank-based scheduling method. In order to train the DDPG model, an appropriate reward function is essential since it provides feedback to the agent based on the given action's effectiveness.

Equation (9) describes the reward function $R(\cdot)$, which comprises two parts: Beneficial-cost (Υ) and Starvation-cost (Φ), providing a trade-off between the high-priority, starvation, and highly-reliable services. The $R(\cdot)$ is a point-based function; depending upon the significance of factors, the reward points (θ_{pts}) are scaled, as in (10) and (11), where $n(\cdot)$ is the normalized function.

$$R(\Psi) = \Upsilon_\Psi + \Phi_\Psi, \tag{9}$$

$$\Upsilon_\Psi = [(1-n(T_\Psi)\times\theta_{pts})+(\Gamma_\Psi\times\theta_{pts})+(P_\Psi\times\theta_{pts})+(Cl_\Psi\times\theta_{pts})], \tag{10}$$

$$\Phi_\Psi = \zeta_\Psi \times \theta_{pts}, \tag{11}$$

$$\zeta_\Psi = \begin{cases} \alpha(1-\epsilon)^\kappa, & \text{if } Z_\Psi = 1 \\ 0, & \text{otherwise,} \end{cases} \tag{12}$$

$$Z_\Psi = \begin{cases} 1, & \text{if } P_\Psi \leq 0.2 \\ 0, & \text{otherwise,} \end{cases} \tag{13}$$

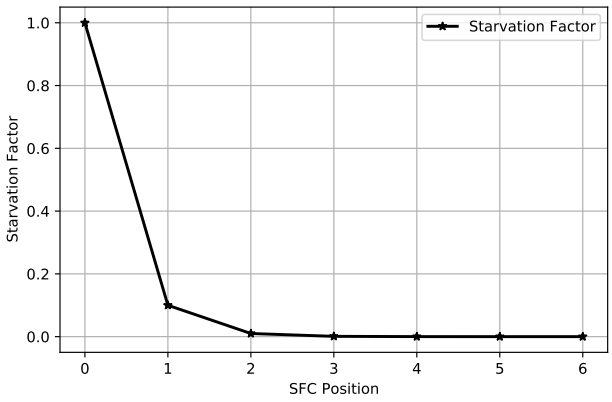

FIGURE 4. Decay Rate Graph Representing the Starvation Rate According to SFC Position

Equation (11) introduces the biasness to diminish the starvation issue, where ζ_Ψ represents the starvation factor, which decays exponentially with the SFC's rank. According to (13), our algorithm determines whether the arriving service qualifies as a 'Potential Starving' service. In the affirmative, the algorithm checks its placement position κ in the adaptive scheduling queue, which impacts the starvation cost. κ is determined using the exponential decay formula, with α as 1 and decay rate $(1-\epsilon)$ as 0.1, as in Figure 4. The decay rate induces greediness in the system when the service is positioned at the start (i.e., SFC position 0) by giving higher rewards than others (i.e., SFC position 1, 2,..., and so on). Thus, making the agent position the starving services at the beginning. To assess the performance of our proposed model, we have constructed traditional FIFO and Priority-based approaches, as well as Weighted Fair Queuing (WFQ).

D. Admission Control

After achieving an idea scheduling queue, we constructed an admission-control model to evaluate the extent to which the 'Potential Starving' and 'High-priority' services are deployed before expiration. This determines the trade-off between starvation and traditionally-preferred services. When a service is under placement, the waiting period (Δ) for the reminder services is recorded, which is illustrated in a 2-D matrix as below. A scheduling queue, for instance, is $[SFC_0, SFC_1, SFC_2, SFC_3]$. When SFC_0 is placed, the remainder services' waiting span is x_0, which is the deployment time of SFC_0. With the deployment of SFC_1, the waiting time for SFC_2 and SFC_3 is increased by x_1, and so on. $\Delta_{\Psi,\Psi}$ depicts the total waited time by the service Ψ in the queue.

$$\Delta = \begin{array}{c} \\ SFC_0 \\ SFC_1 \\ SFC_2 \\ SFC_3 \end{array} \begin{array}{c} \begin{array}{cccc} SFC_0 & SFC_1 & SFC_2 & SFC_3 \end{array} \\ \begin{pmatrix} \Delta_{0,0} & \Delta_{0,1} & \Delta_{0,2} & \Delta_{0,3} \\ \Delta_{1,0} & \Delta_{1,1} & \Delta_{1,2} & \Delta_{1,3} \\ \Delta_{2,0} & \Delta_{2,1} & \Delta_{2,2} & \Delta_{2,3} \\ \Delta_{3,0} & \Delta_{3,1} & \Delta_{3,2} & \Delta_{3,3} \end{pmatrix} \end{array}$$

$$\Delta = \begin{array}{c} \\ SFC_0 \\ SFC_1 \\ SFC_2 \\ SFC_3 \end{array} \begin{array}{c} \begin{array}{cccc} SFC_0 & SFC_1 & SFC_2 & SFC_3 \end{array} \\ \begin{pmatrix} 0 & x_0 & x_0 & x_0 \\ 0 & x_0 & x_0+x_1 & x_0+x_1 \\ 0 & x_0 & x_0+x_1 & x_0+x_1+x_2 \\ 0 & x_0 & x_0+x_1 & x_0+x_1+x_2 \end{pmatrix} \end{array}$$

To initiate the placement, the service must satisfy the requirement as in (14).

$$\Xi'_{\Psi} = \begin{cases} 1, & \text{if } \Delta_{\Psi,\Psi} \leq T_{\Psi} \\ 0, & \text{otherwise,} \end{cases} \tag{14}$$

E. VNF-FGE Problem Formulation

It is important to note that this work does not provide an exact mathematical formulation with decision variables and objectives for the VNF-FG problem. Instead, it represents a RL adaptation of an exact formulation outlined in our previous work [9]. Results from our previous work shows that the Double Deep Q Learning (DDQL) model solves the NFV-RA problem efficiently, intending to embed maximum SFCs onto the substrate network under certain defined constraints. Like, for a successful deployment, it is essential to ensure that the demanded resources by VNFs, micro-VNF, VLs and micro-VLs are adequately met by the underlying substrate nodes and links within the topology. In line with the promised QoS, it is crucial that these VLs/micro-VLs establish a seamless path from the head VNF/micro-VNF to the end VNF/micro-VNF. This path should be devoid of loops to maintain the desired QoS. For instance, if both the head VNF/micro-VNF and end VNF/micro-VNF are located on the same substrate node, there's a significant risk of creating unnecessary loops. The determination of an optimal link path for an SFC relies on latency considerations.

In [9] we considered the physical topology as a DDQL environment comprised of high-volume servers. Each VNF-FG and its resource requirements are represented as state space (S). The state-space for the requested SFC Ψ is represented as $(\mathbf{V}_{\Psi}, \mathbf{mV}_{\Psi})$, where $\mathbf{V}_{\Psi}$ is the set of VNFs and their resource requirements. $\mathbf{mV}_{\Psi}$) represents the micro-VNFs and their resource requirements. The amount of physical nodes/servers present in the topology is described as the action space (A). For a SFC (Ψ) the action-space is defined as $\mathbf{A}_{\Psi} = [\mathbf{A}_{\Psi}^{vnf}, \mathbf{A}_{\Psi}^{mvnf}]$. $\mathbf{A}_{\Psi}^{vnf}$ and $\mathbf{A}_{\Psi}^{mvnf}$ represents the action-space for monolithic VNFs and micro-VNFs, respectively. $\mathbf{A}_{\Psi}^{vnf}$ is represented as $[\mathbf{a}_{\Psi,0}^{vnf}, \ldots, \mathbf{a}_{\Psi,|V|-1}^{vnf}]$, where for each VNF ($x$) the action-space is $\mathbf{a}_{\Psi,x}^{vnf}$ is $[a_{\Psi,x,0}^{vnf}, \ldots a_{\Psi,x,|H|-1}^{vnf}]$. The expansion of microservices introduces a rise in the dimensional complexity of the action-space size, as mentioned in our work [9]. The Local reward function (i.e., the Equation (13) in [9]) has been modified as (15), which is constructed based on the four attributes:

1) $R_{quality,y}$, the quality of the selected node (y) for deploying the VNF(x), is the ratio of available resources (Ar_y) in the node to the initialized resources (Ir_y), as in (17). The availability of resources in a physical node determines its quality.

$$L_{reward}(x) = \begin{cases} \boldsymbol{R_{vnf}}, & \text{if } \Phi_x^y = 1 \\ P_{vnf}^{pt}, & \text{otherwise} \end{cases} \tag{15}$$

$$\boldsymbol{R_{vnf}} = R_{quality,y} + R_{priority} + R_{rel} + R_{placement} + R_{Tload} \tag{16}$$

$$R_{quality,y} = \frac{Ar_y}{Ir_y} \times R_{vnf}^{pt} \tag{17}$$

$$R_{priority} = P_{\Psi} \times R_{vnf}^{pt} \tag{18}$$

$$R_{rel} = \Gamma_{\Psi} \times R_{vnf}^{pt} \tag{19}$$

$$R_{placement} = J_y^x \times R_{vnf}^{pt} \tag{20}$$

$$R_{Tload} = (AgtL_{\Psi}(\varepsilon) \times Cl_{\Psi}) \times R_{vnf}^{pt} \tag{21}$$

$$Cl_{\Psi} = \begin{cases} \mathbf{1}, & \text{if } SFC = `High - Demand' \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

2) Based on the service priority ($R_{priority}$), as in (18).
3) Based on the requested service reliability (R_{rel}), as (19).
4) Time taken by the DDQL model to find an appropriate node for the placement (J_y^x), (20) represents the placement reward function ($R_{placement}$).
5) Based on the traffic load R_{Tload} induced by the SFC, where $AgtL_{\Psi}(\varepsilon)$ is the aggregated traffic and Cl_y is the classified class, as in (21) and (22), respectively.

IV. Overview of the Proposed model

Algorithm 2 provides a summary of the proposed models. Lines 1 to 3 initialize the parameters for DDQL, DDPG, and RR models. The trained classifier from Algorithm 1 is initially implemented in the system to classify the arriving services. At the start of each episode, the environment is restored to its original state with abundant resources in Lines 5-7. For each episode, T services arrive, and the

traffic load is initialized for them (Lines 8-10). The dynamic priority for each service is assigned using the Online-RR approach, as shown in Lines 11-20, where the model is trained from the online services until it reaches a specific threshold transition. When the model is trained, the status changes from 'Train' to 'Predict'. Along with the QoS, the achieved priority value is fed into the classifier, which predicts whether the service is HD or NHD (Line 21). After that, all service information is given to the DDPG model in order to achieve a rank for rescheduling or rearranging the deployment queue (Lines 22-25). The DDQL begins the placement process based on the newly acquired deployment queue (Lines 26-29). The primary innovation in this paper is the introduction of an end-to-end deployment process for incoming SFCs, incorporating ML-based dynamic prioritization and adaptive scheduling. Unlike existing works, which mainly rely on static or deterministic prioritization approaches, our approach adopts the advanced approach. Furthermore, previous research primarily focused on only priority for adaptive scheduling, leaving room for exploration. To the best of our knowledge, this marks the first instance of introducing an end-to-end ML-based algorithm for deployment, prioritization, and scheduling.

Several strategies have been explored for the reliability and performance enhancement of the training process.Firstly, within the DRL model framework, the experimentation involves adjusting learning rates and employing step-decaying epsilon values to balance exploration and exploitation in the action space. Secondly, regularization techniques such as L1 and L2 regularization and batch normalization (as exemplified in the 'Online-RR' model) are applied to mitigate overfitting and enhance the model's reliability during training. Additionally, the study delves into the impact of varying batch sizes and sampling strategies, recognizing that smaller batches may introduce more noise but potentially lead to faster convergence. In comparison, larger batches offer a more stable gradient estimate at the potential expense of slower convergence. Lastly, hyperparameter tuning involves adjusting parameters such as the number of layers, hidden units, dropout rates, and optimisation algorithms to identify configurations that enhance training stability and overall performance. Collectively, these strategies aimed to improve the robustness and effectiveness of the training process.

Algorithm 2 DyPr and AdSch

```
Initialize DDPG Model: Critic, Actor, Target Critic, Target Actor Networks, DDPG Replay Buffer B_DDPG
Initialize DDQL model
Initialize RR Memory Buffer B_RR
Trained Classifier using Algorithm 1
foreach episode i = 1... epi do
    Reset the Environment
    Initialize substrate node resource R_H, substrate link R_N
    Received arriving T services
    while for all T services do
        Initialize traffic load for deploying service; Agt_load
        Using DyPr method; online-RR model
        if status = "Train" then
            if Transition < Threshold Transition then
                Priority is selected randomly
            else
                Online-RR model gets trained
            end
        else
            Prediction: using Trained model
        end
        Using trained classifier: predict Cl_y
        Using AdSch method; DDPG model
        Achieve the Rank for each services
    end
    Sort the T service in ascending order (T'), according to achieved rank
    foreach time-step t' = 1...T' do
        Admission Control using (14)
        Deployment initiated
    end
end
```

A. Complexity Analysis

Analyzing the time complexity of the provided algorithm entails breaking down the steps and estimating the computational cost associated with each part as follows:

1) The time complexity of the Ridge Regression algorithm depends on the number of services and features, i.e., $O(|T| \times P)$ for convergence, where $|T|$ represents the number of services and P is the features per service.
2) The time complexity of the DDPG algorithm can be analyzed based on its key components. DDPG is an actor-critic algorithm designed for continuous action spaces. The critic network is updated by minimizing the temporal difference error. The time complexity of a single critic network update is typically $O(\varrho_{critic})$, where ϱ_{critic} is the number of parameters in the critic network. The actor-network is updated to maximize the expected return. Similar to the critic, the time complexity of a single actor network update is generally $O(\varrho_{actor})$, where ϱ_{actor} is the number of parameters (weights and biases) in the actor-network. DDPG uses an experience replay buffer to store and sample experiences for training. The time complexity of inserting an experience into the buffer is $O(1)$, and the time complexity of sampling a batch of experiences is $O(\varrho_{batch})$, where ϱ_{batch} is the batch size. Putting these components together, the overall time complexity per training iteration for DDPG can be expressed as $O(|T| \times (\varrho_{critic} + \varrho_{actor}) + \varrho_{batch})$.

3) The DDQL algorithm's time complexity depends on the Neural Network (NN)'s training, which involves forward and backward passes for each service. The time complexity of a forward and backward pass through the neural network is often $O(\varrho_{NN})$, where ϱ_{NN} is the number of parameters (weights and biases) in the network. The number of service can be denoted as $|T|$. Therefore, the overall time complexity for neural network training is $O(|T| \times \varrho_{NN})$. The time complexity of inserting an experience into the buffer is $O(1)$, and the time complexity of sampling a batch of experiences is $O(\varrho_{batch})$, where ϱ_{batch} is the batch size. Considering these factors, the overall time complexity of DDQL can be expressed as $O(|T|_{NN} + \varrho_{batch})$.

In summary, the overall time complexity of the provided algorithm can be expressed as: $O((|T| \times P) + (|T| \times (\varrho_{critic} + \varrho_{actor}) + (\varrho_{batch} + |T|_{NN} + \varrho_{batch}))$.

V. Simulation results

The simulation results are explained in two parts; a) The effectiveness of the DDPG and RR models for AdSch and DyPr are examined under various conditions, and b) The impact of the traffic-aware VNF deployment using DDQL model and AdSch is investigated. These models' performances have been studied for Netrail (7 nodes, 10 links) and BtEurope (24 nodes, 37 links) topologies, under diverse substrate nodal and link capacities (i.e., from highly available resource topology to easily exhausted). The online services are constructed using the Erdős–Rényi model with different structural complexity and resource requirements. Each run consists of 2000 episodes, and each episode is expected to have a maximum of 100 services. The DDPG and Ridge model is designed in Python language using the PyTorch library, and the simulations are run on an Intel Core i7 processor with 64 GB RAM. Table 2 lists the parameters applied to develop the models and services.

TABLE 2. DDQL+Adsch+DyPr Model Parameters

DDPG Model	
Alpha	0.0001
Beta	0.001
Gamma	0.99
Batch Size	64
Optimizer	Adam
Memory Size	50000
Hidden Layers	6
Neurons per Layer	300
Neural Network	2
Activation function	Sigmoid

In this work, we are considering a 'worse-case' scenario, where a maximum of 100 services arrive at once, imposing a significant load and high variation on the topology. Moreover, most arriving services need to be deployed sooner, as their threshold waiting time is considerably less, which adds to the system's complexity. Our DyPr and AdSch model's efficiency is compared with traditional queuing models like FIFO, WFQ, and High-Priority-based scheduling. The training dataset for the simulation results has been synthetically generated. This artificial dataset was explicitly created to train the models. Each quality of service (QoS) parameter for the service function chains (SFCs), such as delay, jitter, packet loss, and other relevant metrics, was generated using a Gaussian distribution with varying mean and standard deviation values, as shown in the Table 3.

QoS attributes	Distribution value
Flow Behaviour	μ = 0.5, σ = 0.1
Delay	μ = 4, σ = 10
Jitters	μ = 4, σ = 10
Packet Loss	μ = 2, σ = 1.0
Waiting Time	μ = 5, σ = 10
Computing Resources	μ = 3, σ = 0.4
SFC Arrival	Poisson Distribution μ = 100
Traffic Generated	Factorial Brownian Motion Method = ' daviesharte' Hurst = 0.7

TABLE 3. Model Configuration

A. Need for Priority

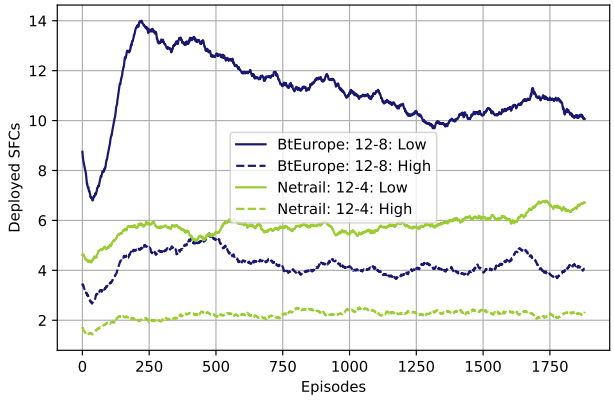

FIGURE 5. Performance of FIFO for Netrail and BtEurope Topology

Figure 5 represents the deployment of high and low-priority SFCs in a FIFO manner for Netrail and BtEurope. The model deploys the services as it comes, unable to distinguish between emergency and non-emergency services as the deployment occurs without any guidelines or prior knowledge of the service. As a result, less than 6% of urgent/emergency services are preferred, causing considerable rejection of them. This shows the need for priority which is predicted by the online-RR model.

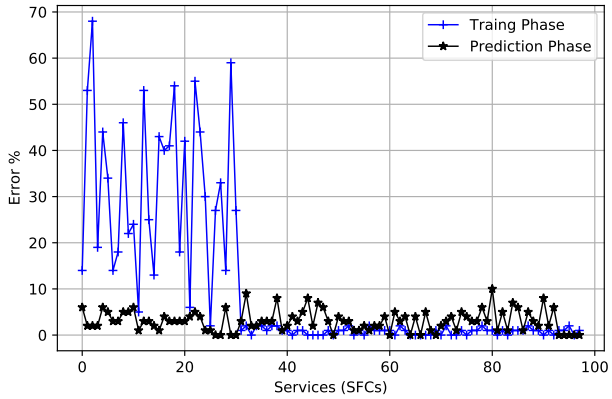

FIGURE 6. **Performance of Online Ridge Regression Model**

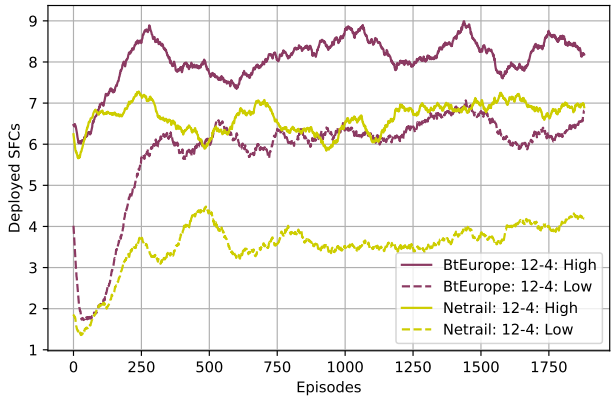

FIGURE 7. **Performance of Priority Algorithm for Netrail SAR: 12-4 scenario**

Figure 6 shows the training phase and prediction phase of the online-RR model. The model gets trained until its accuracy exceeds 80%, however we only displayed for episode 0 for training phase. Initially, the model observed the SFC till 32 iterations; later, it commenced learning by discovering a logistical approach. Figure 6 depicts the prediction for the last episode of a run. It is evident that the model performed well in predicting the priority, as the error% between the predicted and target priority values are less than 10%. Thus, the predicted model's accuracy was also above 80%.

B. Existence of Starvation

The effectiveness of the high-priority-based scheduling paradigm is seen in Fig. 7. Here, the model favours high-priority SFCs above others to avoid the problems caused by FIFO. This triggers a prolonged wait for low-priority SFCs to be deployed, creating a 'Starvation' experience and a low acceptance rate. Despite the large nodal resources or higher density topology, as seen in Figure 7, starvation still exists. This starvation gap might slightly get reduced over time for much denser topologies with ample available resources. However, this is an unrealistic topology due to the dynamism, where plentiful resources are not always available.

C. Reduction of Starvation

Figure 8 represents the Service Acceptance Rate (SAR) (contains all priority levels, excluding lower-priority SFCs), and Figure 9 depicts the SAR exclusive for low-priority SFCs for Netrail and BtEurope topologies with 12-4 and 12-8 CPU cores. In Figure 8, the WFQ and high-priority-based models embed a large number of high-priority SFCs to establish a deploying rule. This pattern is repeated when topological density or nodal capacity increases. From the figures, the WFQ and high-priority-based models established a deploying rule by embedding only beneficial SFCs. This pattern is repeated when topological density or nodal capacity increases. The DDPG model, on the other hand, discovered a trade-off between high and low-priority SFCs

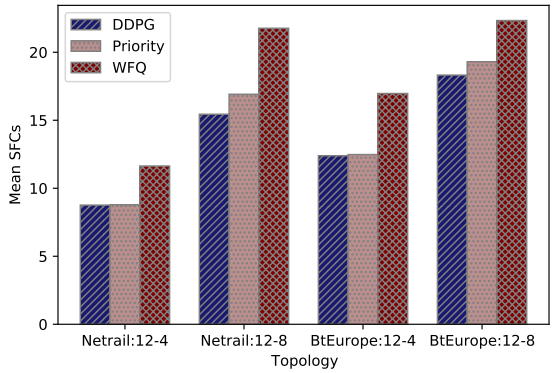

FIGURE 8. **SAR: Deploying Beneficial Services**

by identifying 'Beneficial and Starving' SFCs, resulting in a higher rate of deployment of low-priority SFCs than other models. DDPG, like Netrail 12-4 CPU cores, was able to deploy five times more low-priority SFCs than WFQ at the expense of three less high-priority SFCs. However, in the remaining scenarios (Netrail 12-8; BtEurope 12-4, and BtEurope 12-8), there is a 100% increase in the deployment of low-priority services (Fig. 9). This affected high-priority service deployment, with a 30% reduction in Netrail and a 25% reduction in BtEurope (Fig. 8) respectively.

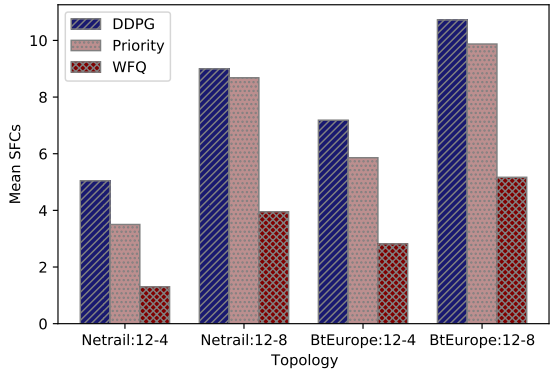

FIGURE 9. **SAR: Deploying Starving Services**

D. Impact of Traffic

From the above study, AdSch using DDPG was able to diminish the starvation of the services by providing a more efficient solution compared to conventional methods. To enhance resource allocation efficiency, traffic inclusion is highly recommended. Thus, for this part of the study, we considered online-RR as the DyPr model, DDPG for AdSch, and DDQL for solving the VNF-FGE problem as a foundation.

1) Accuracy, Precision, and Recall

The accuracy, precision, and recall results are achieved for BtEurope 12-4 scenario, where more than 500 distinct SFCs were generated with various aggregated traffic loads from different users. Each transition is a combination of an SFC with its traffic load at various epochs, making a significantly extensive dataset for training and testing. We have split the dataset into 70/30 ratios for defining the training and testing data. Figures 10, 11, and 12 indicate the performance of the clustering and classifier models in terms of accuracy, precision, and recall metrics. This section will answer the question "Which combination of 'cluster-classifier's model is appropriate for our problem?" Considering Figure 10, to identify the best combination of cluster-classifier models, we started with estimating the baseline accuracy. The phrase "baseline accuracy" describes the minimal level of accuracy that a predictive model must attain in order to be considered functional, which is above 55% in our case.

From the figure, besides SVMs, most classifier models are considered valuable using this metric. However, classifier models like LR and DT needed to perform more effectively for the GM model. The accuracy of models like LR, RF, Decision Tress (DTs) and KNN has shown consistent performances with K-means, Agglomerative clustering and BIRCH classifiers.

Figure 11 illustrates the reliability of the model using a precision metric. From this, the model BIRCH-SVM and all classifier models with GM are expressed as unreliable, i.e., zero or limited positive samples, were correctly categorized among the total classifier positive samples. Classifier models with k-means and agglomerative showed impressive performance as compared to others. The model's sensitivity, or its capacity to identify positive samples accurately, is examined using the recall metric, as demonstrated in Figure 12. Based on this metric, SVM performed poorly overall and models in relation to GM as well. Whereas, especially models RF, LR, and DT with K-means, Agglomerative, and BIRCH performed adequately.

Furthermore, we have analyzed the model's performance in terms of runtime, as shown in Figures 13 and 14. Among the clustering model, our hierarchical model; Agglomerative, outperformed by taking approx. 100 ns to cluster the large datasets, whereas K-means took almost ten times more than Agglomerative, though both the models are an iterative process. From Figure 14, RF, which was one of the highly performed models, consumed a high amount of computational time (almost 8 times more) compared to other models like DTs, KNN.

Overall, comparing the performance of the DTs and RF classifiers, both performed extraordinarily in accuracy, precision, and recall. However, considering the runtime, DTs outperformed RF. Thus, from the above analysis, we have considered agglomerative-DTs as our primary cluster-classifier model, and kmean-LR is our comparative model.

2) Deployment of High-Demand Services

Figures 15 and 16 show the performance of the cluster+classifier model for online SFCs. We chose three model combinations: Agglomerative+DTs, K-means+LR, and BIRCH+SVM, and the performances of these models were investigated for the Netrail and BtEurope topologies with 12-8 and 12-4 nodal capacity, respectively. In both figures, Aggo+DT and Kmean+LR model was able to identify the HD and NHD services, whereas BIRCH+SVM failed to distinguish between the services. The efficiency of SVM is hampered by the 'curse of dimensionality', which occurs when data dimensionality increases, so is in our case. Additionally, SVM performs better with balanced data when there are roughly equal numbers of samples in each class. However, taking into account our situation, our training data is imbalanced for pragmatic reasons. SVM, therefore, has trouble performing well. Figure 15 depicts that the aggo+DT and kmean+LR scenarios classified the services nearly identically, whereas, for BtEurope 12-4 case (Figure 16), the models delivered a different classification pattern. Since the K-means algorithm can converge to various local minima depending on the starting points, the initial placement of centroids in K-means can significantly impact the clustering outcomes. As a result, the algorithm becomes less reliable due to the clustering outcomes depending on the initial centroid placement. Figure 17 portrays the deployment of the HD and all priority services for Netrail 12-4, Netrail 12-8, BtEurope 12-4 and BtEurope 12-8 scenarios. Based on the above findings, we examined the effect of traffic-aware HD services deployment for two models: Aggo+DTs and Kmean+LR. To analyze the performance of these models in terms of deployment rate, we have indicated the total amount of classified HD services arrived per scenario as 'Arrived SFC' in the figure. A significant discrepancy exists between the arrived HD SFCs and the deployed ones in Netrail. The Netrail topology has just seven nodes; even with expanded nodal capacity, the resources tend to be exhausted at a higher rate. Thus, it can only deploy a certain amount of arriving services, establishing an upper limit, as explained in our previous work [9]. Overall, Aggo+DT-based deployment and scheduling outperformed kmean+LR for all scenarios (Netrail 12-4, 12-8; BtEurope 12-4, 12-8).

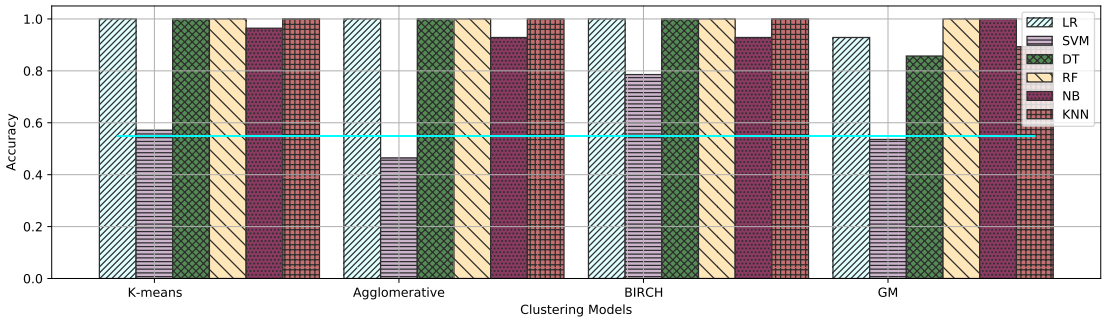

FIGURE 10. **Performance of Classifier Models in terms of Accuracy**

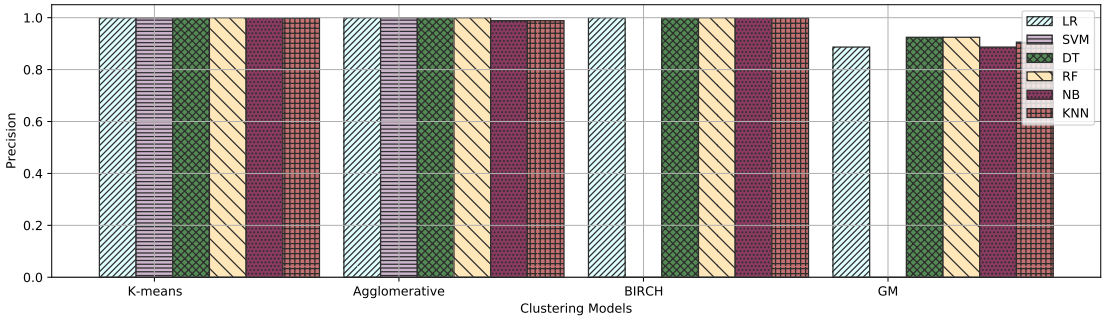

FIGURE 11. **Performance of Classifier Models in terms of Precision**

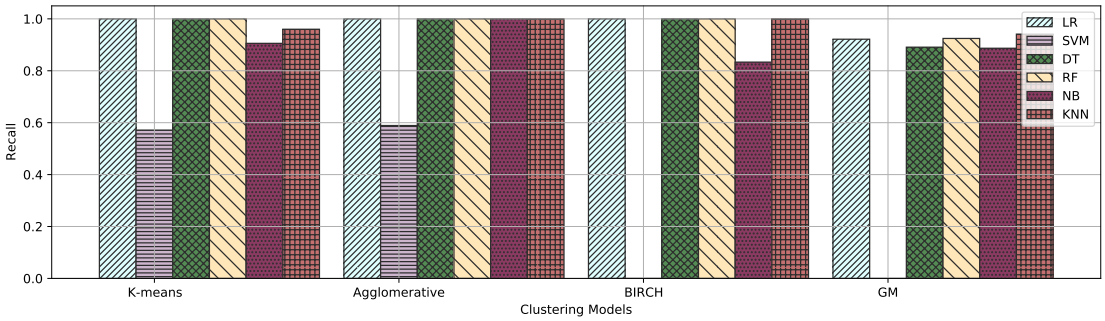

FIGURE 12. **Performance of Classifier Models in terms of Recall**

The analysis of this graph in terms of the rejection rate for both cluster+classifier models is shown in Figure 19. Considering the BtEurope 12-4 scenario, the rejection rate by Aggo+DT is approx. 17%, whereas Kmean+LR is almost 50%, giving 33% of more HD SFCs being deployed by the Aggo+DT model. For BtEurope 12-8, the rejection rate between the cluster+classifier models is more than 25%. Considering the Netrail topology, the rejection rate between the models is less than 20%. For an extensive topology like BtEurope, the Aggo+DT model gave an outstanding performance compared to kmean+LR. On comparing the overall performance of the models in different scenarios, we conclude that Aggo+DT was a clear winner by providing better results than kmean+LR.

3) Deployment of High-Demand Low-Priority Services

Our main goal is to diminish the starvation problem, which was accomplished using our proposed model AdSch. Over here, we incorporated a new feature to the AdSch model to investigate its impact on scheduling and deployment. AdSch model has proven to provide an opportunity to deploy the low-priority service without disrupting the critical SFCs. But some low-priority services are revenue-generating; those are identified as low-priority HD services. Deploying these low-priority HD services over low-priority NHD services can be beneficial to the network operator.

Figure 18 illustrates the deployment of classified low-priority HD services by various cluster+classifier models in Netrail and BtEurope topology, and the performance is compared without the traffic-aware model (i.e., when DDQL and AdSch model are not aware of high revenue generating services). In scenarios like Netrail 12-4, 12-8 and BtEurope

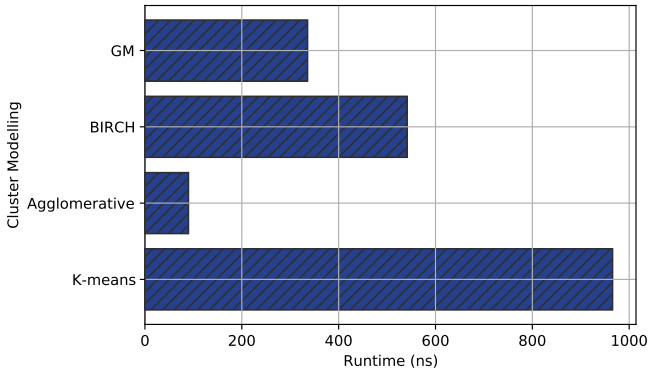

FIGURE 13. Performance of Clustering Models in terms of Runtime

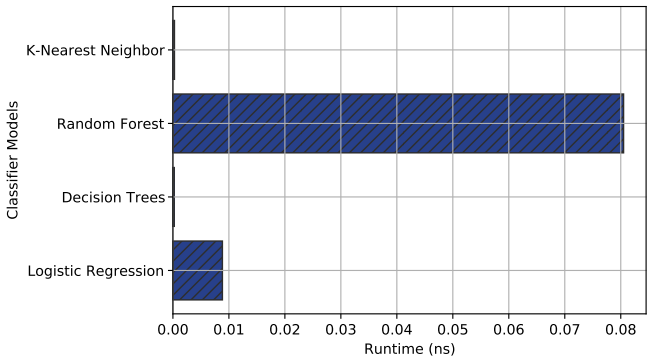

FIGURE 14. Performance of Classifier Models in terms of Runtime

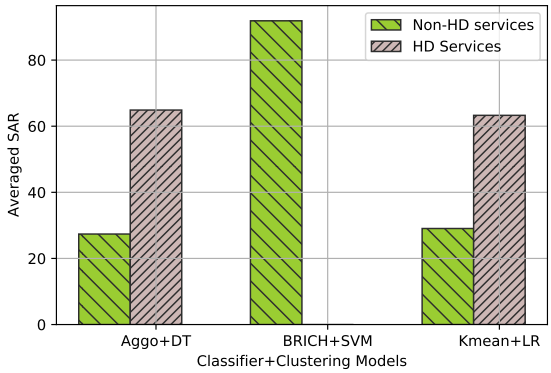

FIGURE 15. Netrail 12-8: HD and NHD classification

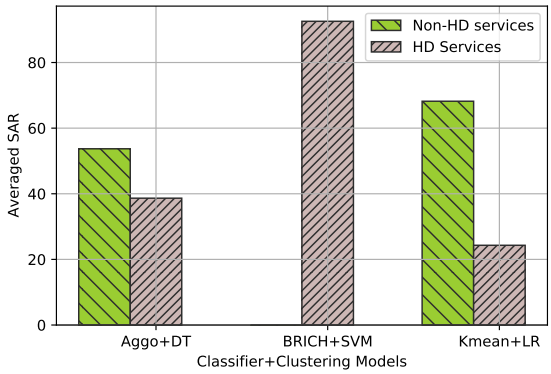

FIGURE 16. BtEurope 12-4: HD and NHD classification

12-4, 12-8, Aggo+DT surpasses the deployment of low-priority HD services, compared to the "without traffic-aware" and kmean+LR models. Like in Netrail 12-8 scenario, with Aggo+DT, our deploying model was able to deploy twice the low-priority SFCs by the "without traffic-aware" model. This depicts that our AdSch and DDQL models understand the importance of low-priority HD services and prefer to deploy them over the network. Again, considering the rejection rate aspect, the high rejection rate is due to the scant resources in the topology. In our experiment, we have demanded extensively high resources; for example, the Netrail 12-4 scenario demanded four times more resources than initialized. Overall, the Aggo+DT models performed better than the remaining models in enhancing the deployment of low-priority HD services.

E. Convergence Analysis

To evaluate the overall performance of our proposed model, we integrate the DDQL model for SFC deployment with microservices, enabling dynamic decomposition of SFCs. This approach is enhanced by dynamic prioritization through an Online-Ridge Regression model, adaptive scheduling using DDPG, and an Aggo+DT model for efficient traffic classification. This comprehensive strategy is assessed against traditional scheduling models such as FIFO and a multi-class Priority-based model, showcasing its advanced capability in managing SFC deployment effectively, as shown in Figures 21 and 22.

In the previously discussed section, the evaluation of our model centered on SFC deployments. This segment shifts focus to assess model performance via residual resources per iteration to understand convergence behaviors. This evaluation employs the Netrail topology, exploring diverse resource capacities to ensure depth in our analysis. Figures 21 and 22 delineate the comparison between our innovative model, DDPG+DDQN+Microservices, and traditional scheduling approaches, namely FIFO+DDQN+Microservices and Pr+DDQN+Microservices. To maintain fairness in evaluation, we preserved the DDQN+microservices deployment model as this study's foundation, allowing us to scrutinize the differential impact of various scheduling methodologies under similar conditions.

Figure 21, depicting the Netrail 12-4 scenario, demonstrates our model's proficiency in interpreting the optimisation function, thereby maximizing SFC deployments. Notably, post-500 episodes, our model showcased convergence, with fewer than 10 CPU cores remaining unallocated, highlighting its efficiency. In contrast, the traditional models

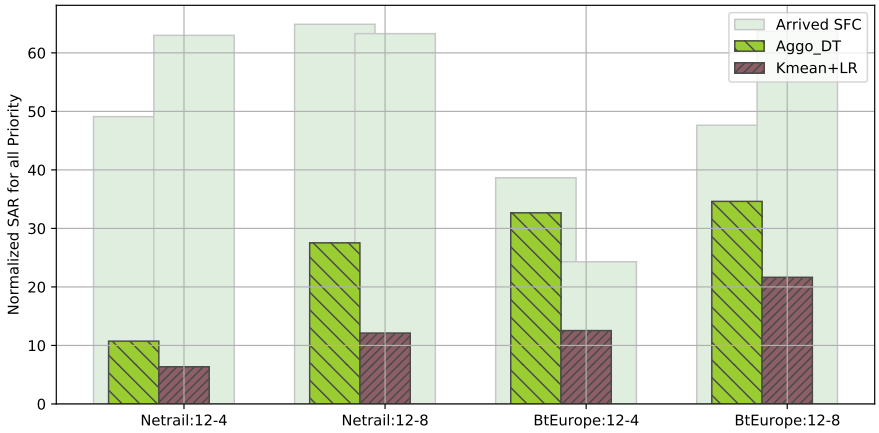

FIGURE 17. **Deployment of All-Priority HD Services**

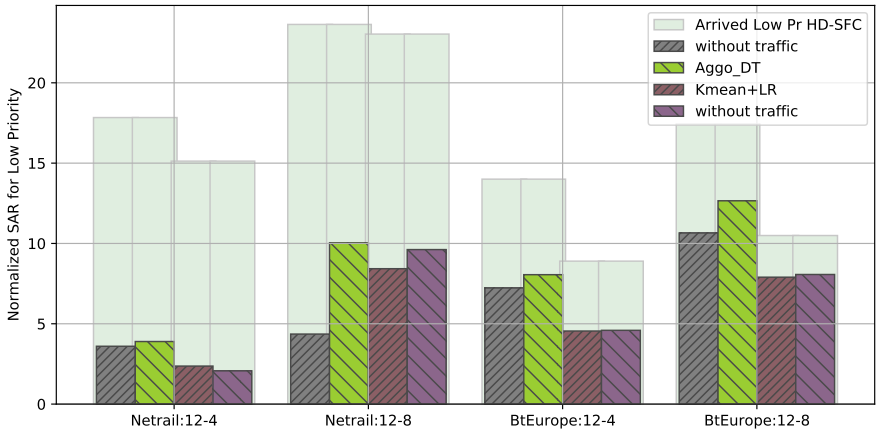

FIGURE 18. **Deployment of Low-Priority HD Services**

(Pr+DDQN and FIFO+DDQN) exhibited a delayed optimisation response. Figure 22 further solidifies our model's superiority, especially in scenarios with enhanced nodal capacities, facilitating a greater volume of SFC deployments. The proposed model consistently outperformed traditional methods, showcasing lower residual computing resources and quicker optimisation convergence. This disparity was particularly evident with FIFO's unstable learning curve and Pr+DDQN's prolonged optimisation period.

Through this comprehensive analysis, it becomes evident that our adaptive scheduling approach significantly influences SFC deployments' efficacy. The superior performance of the DDPG+DDQN+Microservices model, especially in terms of rapid convergence and efficient resource utilization, underscores its potential to revolutionize SFC deployment strategies within complex network topologies.

VI. Conclusions

The main aim of this study was to showcase the existence of the starvation problem, which the researchers have neglected. In this work, we have explained the existence of the starvation problem in the current scheduling methods and how the scheduling process should not be based only on priority but also on other important factors like threshold waiting time and reliability. With a motive to propose an intelligent scheduling scheme, our model DDPG has performed efficiently by deploying twice as many low-priority services as others. The DDPG agent successfully identified the 'Beneficial and Starving' services, which caused a reduction in the starvation of low-priority services. We have proposed a method to define dynamic priority for the upcoming services without hindrance. Our online-RR model learns the pattern within 100 transitions, and the accuracy rate for the prediction model is above 80%. Moreover,

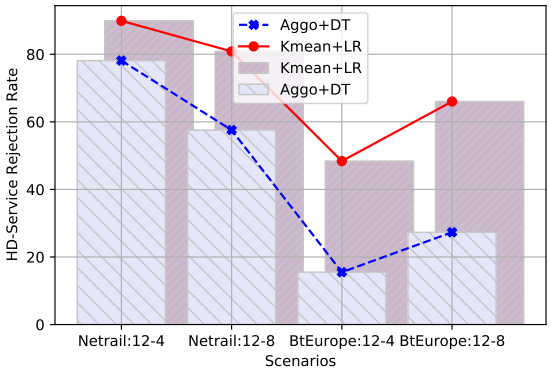

FIGURE 19. **Rejection Rate: All-Priority HD Service**

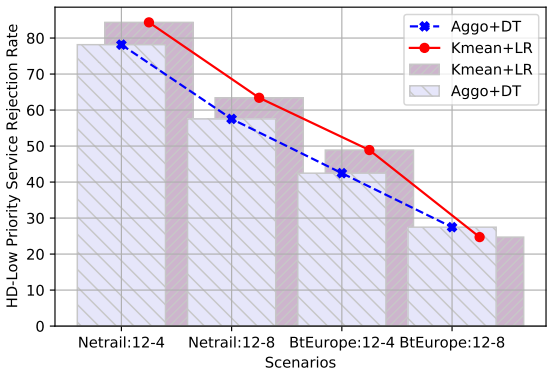

FIGURE 20. **Rejection Rate: Low-Priority HD Service**

we have examined the performance of the scheduling and placement models in the presence of traffic, where the trained classifier Aggo+DT model categorizes the online SFC into HD or NHD services. Giving the awareness of high revenue-generating services to the scheduling and placement model. In such a way, a trade-off between the low-priority HD SFCs and low-priority NHD SFCs occurs, to provide an efficient resource-allocation solution. This traffic-aware model was able to deploy twice the low-priority HD services than without the traffic-awareness one. The presence of these problems can have a negative impact in the future if not addressed correctly. An in-depth examination of the convergence properties and an exploration of upper bounds for the proposed algorithms will be undertaken as part of future research efforts.

Acknowledgment

This work was supported in part by the Engineering and Physical Sciences Research Council United Kingdom (EPSRC), Impact Acceleration Accounts (IAA) (Green Secure and Privacy Aware Wireless Networks for Sustainable Future Connected and Autonomous Systems) under Grant EP/X525856/1.

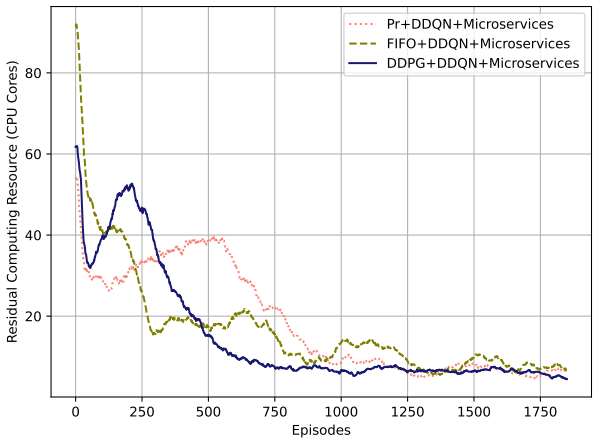

FIGURE 21. **Proposed model performance for Netrail 12-4 scenario**

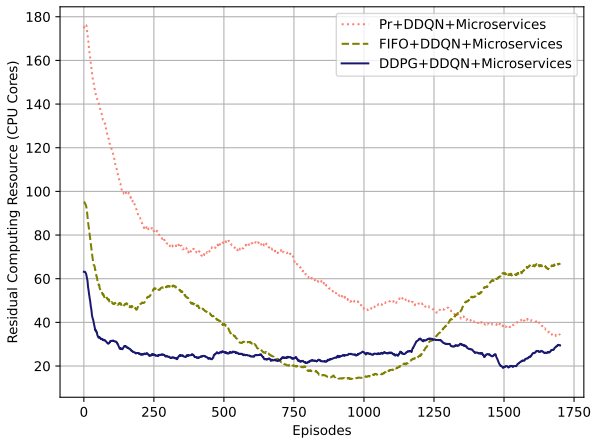

FIGURE 22. **Proposed model performance for Netrail 12-8 scenario**

References

[1] W. Saad, M. Bennis, and M. Chen, "A Vision of 6G Wireless Systems: Applications, Trends, Technologies, and Open Research Problems," *IEEE netw.*, 2019.

[2] H. Ahmadi, A. Nag, Z. Khan, K. Sayrafian, and S. Rahardja, "Networked Twins and Twins of Networks: An Overview on the Relationship Between Digital Twins and 6G," *IEEE Commun. Stan. Mag.*, vol. 5, no. 4, pp. 154–160, 2021. DOI: 10.1109/MCOMSTD.0001.2000041.

[3] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The Roadmap to 6G: AI Empowered Wireless Networks," *IEEE Commun. Mag.*, vol. 57, no. 8, pp. 84–90, 2019.

[4] "Network Functions Virtualisation – Introductory White Paper." (), [Online]. Available: https://portal.etsi.org/NFV/NFV_White_Paper.pdf. (accessed: 22.08.2022).

[5] M. Nekovee, S. Sharma, N. Uniyal, A. Nag, R. Nejabati, and D. Simeonidou, "Towards AI-enabled Microservice Architecture for Network Function Vir-

tualization," in *IEEE ComNet*, 2020, pp. 1–8. DOI: 10.1109/ComNet47917.2020.9306098.

[6] S. R. Chowdhury, M. A. Salahuddin, N. Limam, and R. Boutaba, "Re-architecting NFV Ecosystem with Microservices: State of the Art and Research Challenges," *IEEE Netw.*, vol. 33, no. 3, pp. 168–176, 2019.

[7] N. Dragoni, S. Giallorenzo, A. L. Lafuente, *et al.*, "Microservices: Yesterday, Today, and Tomorrow," *Present and ulterior software engineering*, pp. 195–216, 2017.

[8] J. Thönes, "Microservices," *IEEE Software*, vol. 32, no. 1, pp. 116–116, 2015. DOI: 10.1109/MS.2015.11.

[9] S. B. Chetty, H. Ahmadi, M. Tornatore, and A. Nag, "Dynamic Decomposition of Service Function Chain Using a Deep Reinforcement Learning Approach," *IEEE Access*, pp. 1–1, 2022. DOI: 10.1109/ACCESS.2022.3215744.

[10] A. Rahman and X. de Foy, "Network Slicing - 3GPP Use Case," 2017.

[11] P. Cappanera, F. Paganelli, and F. Paradiso, "VNF Placement for Service Chaining in a Distributed Cloud Environment with Multiple Stakeholders," *Computer Communications*, vol. 133, pp. 24–40, 2019.

[12] A. Mohamad and H. S. Hassanein, "PSVShare: A Priority-based SFC Placement with VNF Sharing," in *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, IEEE, 2020, pp. 25–30.

[13] K. Qu, W. Zhuang, X. Shen, X. Li, and J. Rao, "Dynamic Resource Scaling for VNF Over Nonstationary Traffic: A Learning Approach," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 2, pp. 648–662, 2020.

[14] S. Park, H.-G. Kim, J. Hong, S. Lange, J.-H. Yoo, and J. W.-K. Hong, "Machine learning-based optimal vnf deployment," in *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2020, pp. 67–72. DOI: 10.23919/APNOMS50412.2020.9236970.

[15] T. Subramanya and R. Riggio, "Machine learning-driven scaling and placement of virtual network functions at the network edges," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, IEEE, 2019, pp. 414–422.

[16] Y. Yuan, Z. Tian, C. Wang, F. Zheng, and Y. Lv, "A Q-Learning-based Approach for Virtual Network Embedding in Data Center," *Neural Comput. & Appl.*, vol. 32, no. 7, pp. 1995–2004, 2020.

[17] V. Sciancalepore, F. Z. Yousaf, and X. Costa-Perez, "z-TORCH: An Automated NFV Orchestration and Monitoring Solution," *IEEE Trans. Netw. & Serv. Manag.*, vol. 15, no. 4, pp. 1292–1306, 2018.

[18] P. T. A. Quang, Y. Hadjadj-Aoul, and A. Outtagarts, "A Deep Reinforcement Learning Approach for VNF Forwarding Graph Embedding," *IEEE Trans. Netw. & Serv. Manag.*, vol. 16, no. 4, pp. 1318–1331, 2019.

[19] Y. Mu, L. Wang, and J. Zhao, "Energy-efficient and interference-aware vnf placement with deep reinforcement learning," in *2021 IFIP Networking Conference (IFIP Networking)*, 2021, pp. 1–9. DOI: 10.23919/IFIPNetworking52078.2021.9472805.

[20] 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.

[21] M. Jalalitabar, G. Luo, C. Kong, and X. Cao, "Service Function Graph Design and Mapping for NFV with Priority Dependence," in *2016 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2016, pp. 1–5.

[22] R. Li, Z. Zhao, Q. Sun, *et al.*, "Deep Reinforcement Learning for Resource Management in Network Slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018. DOI: 10.1109/ACCESS.2018.2881964.

[23] X. Chen, Z. Li, Y. Zhang, *et al.*, *Reinforcement Learning based QoS/QoE-aware Service Function Chaining in Software-Driven 5G Slices*, 2018. arXiv: 1804.02099 [cs.NI].

[24] G. Li, B. Feng, H. Zhou, Y. Zhang, K. Sood, and S. Yu, "Adaptive Service Function Chaining Mappings in 5G Using Deep Q-Learning," *Computer Communications*, vol. 152, pp. 305–315, 2020.

[25] N. Siasi and A. Jaesim, "Priority-Aware SFC Provisioning in Fog Computing," in *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*, 2020, pp. 1–6. DOI: 10.1109/CCNC46108.2020.9045275.

[26] H. Qu, K. Wang, and J. Zhao, "Priority-awareness VNF Migration Method based on Deep Reinforcement Learning," *Computer Networks*, vol. 208, p. 108 866, 2022.

[27] "Mobile Data Traffic Outlook." (), [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-traffic-forecast.

[28] L.-V. Le, D. Sinh, B.-S. P. Lin, and L.-P. Tung, "Applying Big Data, Machine Learning, and SDN/NFV to 5G Traffic Clustering, Forecasting, and Management," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, IEEE, 2018, pp. 168–176.

[29] J. Erman, M. Arlitt, and A. Mahanti, "Traffic Classification Using Clustering Algorithms," in *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*, ser. MineNet '06, Pisa, Italy: Association for Computing Machinery, 2006, pp. 281–286, ISBN: 159593569X. DOI: 10.1145/1162678.1162679. [Online]. Available: https://doi.org/10.1145/1162678.1162679.

[30] A. R. Mohammed, S. A. Mohammed, and S. Shirmohammadi, "Machine Learning and Deep Learning based Traffic Classification and Prediction in Software Defined Networking," in *2019 IEEE International Symposium on Measurements & Networking (M&N)*, IEEE, 2019, pp. 1–6.

[31] J. Xie, F. R. Yu, T. Huang, *et al.*, "A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 393–430, 2019. DOI: 10.1109/COMST.2018.2866942.

[32] P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares, and H. S. Mamede, "Machine Learning in Software Defined Networks: Data Collection and Traffic Classification," in *2016 IEEE 24th International conference on network protocols (ICNP)*, IEEE, 2016, pp. 1–5.

[33] Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, and G. Noubir, "Application-awareness in SDN," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, 2013, pp. 487–488.

[34] P. Wang, S.-C. Lin, and M. Luo, "A Framework for QoS-aware Traffic Classification Using Semi-Supervised Machine Learning in SDNs," in *2016 IEEE international conference on services computing (SCC)*, IEEE, 2016, pp. 760–765.

[35] S. Rezaei and X. Liu, "Deep Learning for Encrypted Traffic Classification: An Overview," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 76–81, 2019. DOI: 10.1109/MCOM.2019.1800819.

[36] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic Virtual Network Function Placement," in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, 2015, pp. 255–260. DOI: 10.1109/CloudNet.2015.7335318.

[37] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A Cost-Aware Elasticity Provisioning System for the Cloud," in *2011 31st International Conference on Distributed Computing Systems*, 2011, pp. 559–570. DOI: 10.1109/ICDCS.2011.59.

[38] W. Ma, C. Medina, and D. Pan, "Traffic-aware Placement of NFV Middleboxes," in *2015 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2015, pp. 1–6.

[39] W. Ma, O. Sandoval, J. Beltran, D. Pan, and N. Pissinou, "Traffic Aware Placement of Interdependent NFV Middleboxes," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, IEEE, 2017, pp. 1–9.

[40] K. Qu, W. Zhuang, Q. Ye, X. Shen, X. Li, and J. Rao, "Traffic Engineering for Service-Oriented 5G Networks with SDN-NFV Integration," *IEEE Network*, vol. 34, no. 4, pp. 234–241, 2020. DOI: 10.1109/MNET.001.1900508.

[41] Z. Luo, C. Wu, Z. Li, and W. Zhou, "Scaling Geo-Distributed Network Function Chains: A Prediction and Learning Framework," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1838–1850, 2019. DOI: 10.1109/JSAC.2019.2927068.

[42] 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," *Comput. Netw.*, vol. 133, pp. 1–16, 2018.

[43] J. Kim and G. Hwang, "Adaptive Bandwidth Allocation Based on Sample Path Prediction With Gaussian Process Regression," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4983–4996, 2019. DOI: 10.1109/TWC.2019.2931570.

[44] K. Qu, W. Zhuang, X. Shen, X. Li, and J. Rao, "Dynamic Resource Scaling for VNF Over Nonstationary Traffic: A Learning Approach," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 2, pp. 648–662, 2021. DOI: 10.1109/TCCN.2020.3018157.

[45] A. C. Müller and S. Guido, *Introduction to Machine Learning with Python*. O'Reilly Media, Inc., 2016, ISBN: 9781449369415.

Swarna B. Chetty, having successfully defended her PhD at University College Dublin in Ireland in 2023, is currently a Research Associate at the University of York, actively contributing to the YO-RAN project. She received her B.E degree in Electronics and Communication Engineering from Sathyabama University, Chennai, India, in 2014. Later earned an M.S degree in Mobile Communication Systems from the University of Surrey, UK, in 2016. Prior to her doctoral studies, she gained professional expertise as a software developer. Her research interests include network virtualization, resource allocations, microservices, machine learning (especially reinforcement and deep learning), Open Radio Network Access, 5G, and beyond communications.

Hamed Ahmadiis a Reader in Digital Engineering, at the School of Physics, Engineering and Technology, University of York, UK. He is also an adjunct academic at the school of Electrical and Electronic Engineering, University College Dublin, Ireland. He received his Ph.D. from National University of Singapore in 2012 where he was a SINGA PhD scholar at Institute for Infocomm Research, A-STAR. Since then he worked at different academic and industrial positions in the Republic of Ireland and UK. Dr. Ahmadi has published more than 90 peer reviewed book chapters, journal and conference papers. He is a member of editorial board of IEEE Communication Standards magazine, IEEE Systems and Springer Wireless Networks. He is a senior member of IEEE, and Fellow of UK Higher Education Academy. He has been the Networks working group chair of COST Actions CA15104 (IRACON) and CA20120 (INTERACT). His current research interests include design, analysis, and optimisation of wireless communications networks, the application of machine learning in wireless networks, Open Radio Access and Networking, green networks, airborne networks, Digital twins of networks, and Internet-of-Things.

Dr. Avishek Nag is currently an Assistant Professor in the School of Computer Science at University College Dublin in Ireland. Dr. Nag received the BE (Honours) degree from Jadavpur University, Kolkata, India, in 2005, the MTech degree from the Indian Institute of Technology, Kharagpur, India, in 2007, and the Ph.D. degree from the University of California, Davis in 2012. He worked as a research associate at the CONNECT center for future networks and communication in Trinity College Dublin, before joining University College Dublin. Dr Nag is the recipient of the Best Paper Award at the 2nd IEEE Advanced Networks and Telecommunication Symposium in 2008 and has published over 100 publications including journals, conference proceedings, and book chapters. His research interests include, but are not limited to Cross-layer optimisation in Wired and Wireless Networks, Network Reliability, Mathematics of Networks (Optimisation, Graph Theory), Network Virtualisation, Software-Defined Networks, Machine Learning, Data Analytics, Blockchain, and the Internet of Things. Dr. Nag is a senior member of the Institute of Electronics and electrical engineers (IEEE) and also the outreach lead for Ireland for the IEEE UK and Ireland Blockchain Group.