

## RESEARCH ARTICLE

# Dynamic Decomposition of Service Function Chain Using a Deep Reinforcement Learning Approach

SWARNA B. CHETTY<sup>1</sup>, HAMED AHMADI<sup>2</sup>, (Senior Member, IEEE),  
MASSIMO TORNATORE<sup>3</sup>, (Senior Member, IEEE),  
AND AVISHEK NAG<sup>1</sup>, (Senior Member, IEEE)

<sup>1</sup>School of Electrical and Electronic Engineering, University College Dublin, Dublin 4, D04 V1W8 Ireland

<sup>2</sup>School of Physics, Engineering and Technology, University of York, YO10 5DD York, U.K.

<sup>3</sup>Department of Electronics and Information, Politecnico di Milano, 20133 Milan, Italy

Corresponding author: Avishek Nag (avishek.nag@ucd.ie)

This work was supported by the School of Electrical and Electronic Engineering, University College Dublin.

**ABSTRACT** The Internet of Things (IoT) universe will continue to expand with the advent of the sixth generation of mobile networks (6G), which is expected to support applications and services with higher data rates, ultra-reliability, and lower latency compared to the fifth generation of mobile networks (5G). These new demanding 6G applications will introduce heavy load and strict performance requirements on the network. Network Function Virtualization (NFV) is a promising approach to handling these challenging requirements, but it also poses significant Resource Allocation (RA) challenges. Especially since 6G network services will be highly complicated and comparatively short-lived, network operators will be compelled to deploy these services in a flexible, on-demand, and agile manner. To address the aforementioned issues, microservice approaches are being investigated, in which the services are decomposed and loosely coupled, resulting in increased deployment flexibility and modularity. This study investigates a new RA approach for microservices-based NFV for efficient deployment and decomposition of Virtual Network Function (VNF) onto substrate networks. The decomposition of VNFs involves additional overheads, which have a detrimental impact on network resources; hence, finding the right balance of when and how much decomposition to allow is critical. Thus, we develop a criterion for determining the potential/candidate VNFs for decomposition and also the granularity of such decomposition. The joint problem of decomposition and efficient embedding of microservices is challenging to model and solve using exact mathematical models. Therefore, we implemented a Reinforcement Learning (RL) model using Double Deep Q-Learning, which revealed an almost 50% more normalized Service Acceptance Rate (SAR) for the microservice approach over the monolithic deployment of VNFs.

**INDEX TERMS** 6G, machine learning, Internet of Everything, resource allocation, deep reinforcement learning.

## I. INTRODUCTION

IoT and Industry 4.0 are some of the main applications of 5G, and they will continue to be important as the technology progresses towards 6G. As IoT and Industry 4.0 become more mature, they drive the emergence of more modern applications like Digital Twin (DT), connected robotics, autonomous

The associate editor coordinating the review of this manuscript and approving it for publication was Frederico Guimarães<sup>1</sup>.

systems, Augmented Reality (AR)/ Virtual Reality (VR)/ Mixed Reality (MR), Blockchain and Trust technologies, and wireless brain-computer interfaces [1], [2]. This will further load the network and push for serving a variety of new services and applications instantaneously in addition to the existing ones. To support this evolution towards 6G, the NFV architecture, which was initiated in 2012 [3], is also evolving towards a microservices-based architecture [4], [5], [6], [7]. We assume that readers have basic knowledge

of NFV; however, for more details on the NFV framework, interested readers can refer to [8], [9], and [10]. *Microservices* are a standard software design paradigm that decomposes monolithic systems into several smaller (micro) individual and independent fragments. These micro-fragments are loosely coupled software codes, providing the advantage of easy maintenance like upgrading and scaling the micro-fragments. Due to their independence, the testing and migration become manageable and allow the re-use of the micro-fragments [11]. The microservice concept has proven to be an effective strategy in cloud-based applications such as Netflix [12] and Amazon [13], and since the telecom networks are becoming ‘cloud-native’, the microservices-based NFV framework promises to serve the future networks well. The virtualization approach allows various applications to coexist and simultaneously use the same resource infrastructure, which is at the core of the cloud computing architecture [14].

The NFV-based network provides considerably more freedom, agility and flexibility for the placement of online arriving services. But exploiting this flexibility mapping can cause a significant computational problem. One of the main challenges is ‘NFV-RA’, which is described as the provisioning of guaranteed resources by the network or the underlying infrastructure to the requested Service Function Chainings (SFCs) for effective deployment. NFV-RA, due to its requirements and affinity and anti-affinity constraints, normally becomes an *NP*-hard problem [14]. To efficiently address the NFV-RA, different heuristics and machine-learning-based approaches have been investigated in the literature (discussed in the next section in detail with appropriate references).

Now with proposals to decompose monolithic VNFs into microservices, the NFV-RA problem becomes more interesting, and the solution (theoretically) becomes closer to the optimum. Just to clarify and with a little misuse of the concept, imagine a knapsack problem in which the bigger objects can be decomposed into smaller objects which can have shared functionality/values. The loosely coupled micro-VNFs (*m*VNF) give the freedom of migration, scalability, maintenance, and software update without disrupting the neighbouring VNFs, which is a significant advantage over monolithic architectures. It also has a higher fault tolerance [4], [11]. However, these benefits come with new costs and constraints. Deployment and architectural complexity intensify when this microservice concept is implemented on the VNF-Forwarding Graph (FG) embedding problem [4] as the decomposed micro-functionalities seek more resources like bandwidth and latency. For example, Figure 1 illustrates an SFC composed of 5 VNFs and 7 Virtual Links (VLs) ((0,4), (1,4), (2,1), (2,3), (3,1), (4,0), (4,1)), based on the granularity criteria (which is explained later in Section IV-G) this SFC is further disintegrated at a finer-granular level, as shown in Figure 2. In the given example, each monolithic VNF (hereafter, the standard VNFs will be defined as monolithic VNFs) is decomposed into 5 independent and modular deployable micro-functionalities, generating a total

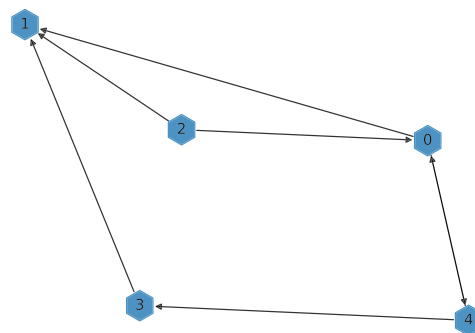


FIGURE 1. Monolithic SFC.

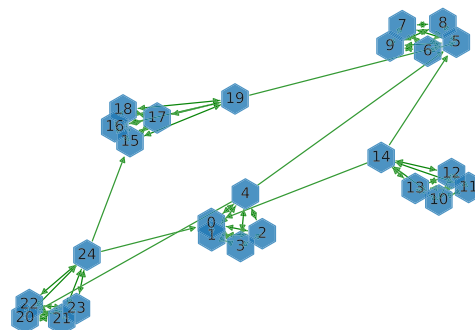


FIGURE 2. Decomposed SFC.

of 25 *m*VNFs, and 100 micro-VLs (*m*VLs) driving a higher amount of resource consumption. As a result, the decomposition of all VNFs is an ineffective approach; hence, identifying the best decomposing scenario for each SFC is vital.

During the deconstruction of monolithic VNFs, a substantial number of standard functionalities are replicated; they are referred to as redundant *m*VNFs. These redundant *m*VNFs will increase the processing overheads and will consume resources unnecessarily, wasting the CPU cycles. To eliminate this limitation, the decomposed VNF-FG needs to be re-architected [7] accordingly. Since the *m*VNFs are lightweight and reusable, i.e., self-contained by performing a specific task, they can be scalable and easily migratable.

In this paper, in the quest to have more agile and fast-orchestrated future networks, we solve a deep-reinforcement-learning-based NFV-RA problem allowing the VNFs to be decomposed to microservices dynamically when the substrate nodes have limited resources to embed and also re-architecting the decomposed VNF-FG before deploying. Since the VNFs are sequentially chained, further decomposition will make the VNF-FG even more complex as VNF-FG becomes denser in terms of connected nodes and links. Optimal embedding of such decomposed SFCs on the substrate network, keeping all physical resource constraints and the interdependencies of the microservices, is difficult to model as an exact mathematical optimization problem. In this paper, we have successfully addressed it using Deep Reinforcement Learning (DRL) for the first time, to the best

of our knowledge. We chose DRL as a solution tool because it provides the right balance between optimality, computational times, and the ability to decide a change in the problem settings (i.e., decomposition of monoliths to microservices in between episodes) and adapt to that change [15]. Study [15] has conducted a comparative analysis on Integer Linear Program (ILP), meta-heuristic, and Deep Reinforcement Learning (DRL)-based models for addressing an NP-hard problem. The authors have demonstrated how DRL-based models are better than others and that ILP works better for smaller topologies, such as those with 10 nodes. But, it is expensive and impractical for much denser topologies. Furthermore, since QoS metrics like latency have a non-linear relationship with traffic flow on the connections, the NFV-RA problem with strict Quality of Service (QoS) requirements cannot be solved by models like ILP [16].

In summary, the main contributions of this paper are:

- Develop a deep Q-learning-based framework for the optimal embedding of VNFs on a substrate network.
- Establish granularity criteria for the decomposition of monolithic VNFs into microservices based on network resources.
- Establish an approach to re-architect the decomposed microservices so that there are no redundant microservices.
- Adapting the deep Q-learning model for VNF embedding to optimally embed the microservices also (along with the microservices interdependencies) if there are decomposed microservices according to the granularity criteria.

The rest of the paper is organized as follows. Section II presents the related literature. Section III describes the formal problem definition, followed by the details of the solution methodologies in Section IV. The constructed Deep Neural Network (DNN) architecture is discussed in Section VI. Our results are presented in Section VI, and finally, we conclude in Section VII.

## II. LITERATURE REVIEW

The SFC's major goal is to offer users with guaranteed Service Level Agreements (SLAs), which is accomplished in three stages: VNF-Chain Composition (CC), VNF-FG,<sup>1</sup> and VNF-Scheduling (SCH). Our work is restricted to stages 2 and 3 only; we consider stage 1 as a predetermined factor provided by the network operator. But the authors, like [17], have suggested that the service chain be constructed adaptively in accordance with the flow management and prioritization strategy.

Traditionally deployment of network services was considered a Bin-Packing problem, which has been thoroughly investigated [16]. Based on the characteristics of the problem, multiple approaches like ILP and Mixed-Integer

Linear Program (MILP) have been implemented to address the online VNF-FG Embedding (FGE) problem.

Like, the optimization in [18] is based on local and global fitness values, which are estimated on the rational rule, where only a subset of the substrate network is considered to reduce execution time, limiting the model's performance. The authors in [19] used a queuing-based approach that decouples the VNF-FGE problem and implements it in a sequential system. The authors of [20] adopted the MILP model to produce a joint optimization for VNF mapping and scheduling for denser and more complicated networks. The authors of [21] have established a strong foundation for the VNF's online mapping and scheduling based on meta-heuristic algorithms. However, due to the repetitive procedure, the technique is non-scalable for larger networks and moreover, [21] overlooked the virtual link mapping and its associated delays. In addition to a heuristic- or meta-heuristic-based approaches or linear formulations, machine learning was frequently recommended as a powerful technique for solving such complex issues. Like, the authors of [22] have suggested a genetic algorithm named GASVIT (Genetic Algorithm for Service mapping with Virtual Topology design) for optical networks to solve the VNF-placement problems. The authors of [23] suggested an architecture based on the Experiential Networked Intelligence (ENI) framework, i.e., a Machine Learning (ML)-assisted architectural solution for mapping the VNFs and VFs in NFV architecture. These ML-assisted solutions can be supervised, unsupervised or Reinforcement Learning. The authors, however, omitted to highlight the benefits of one machine learning model over another or the criteria by which the suggested architecture will choose these models. Authors of [24] and [25] presented a model based on Q-Learning (QL). There are a few other works on applying different variants of RL and DRL. For example, the authors of [16] implemented an Enhanced Exploration Deep Deterministic Policy ( $E^2D^2PG$ ) is based on DRL, but they did not explore the microservices concept.

A majority of the researchers have preferred to solve the VNF-FGE problem based on heuristic or meta-heuristic methods, which resulted in a sub-optimal solution. This indicates a definite trade-off between solution quality and computation time. Even the RL model could not deliver the optimal solution because of the complexity of the problem. As a result, solving the VNF-FGE problem effectively is still an open issue. In this study, to unravel this open issue, we investigate the advanced RL model, Double Deep Q Learning (DDQL), to improve these sub-optimal solutions, where learning from prior experience by upgraded models can be advantageous.

In the context of microservices-based NFV, the studies in [5], [6], [26] and [27] investigated the deployment of decomposed VNF-Forwarding Graphs (VNF-FGs). Based on the traffic demands, the authors in [6] suggested a MILP model to determine the best decomposition for each SFC from the pool of already decomposed SFCs. The authors of [27] used the DRL model for the embedding of

<sup>1</sup>It is a graphical representation of VNFs that are sequentially chained and deliver an end-to-end network service.

TABLE 1. Summary of related work.

Ref	Model	Resource Type	Rel. CPU and RAM	Decomposition	Online Decomposition
[5]	M/M/I	CPU	No	Yes	No
[6]	MILP	CPU, BW, Traffic	No	Yes	No
[16]	DRL	CPU, RAM, BW, Latency, Loss Rate	No	No	No
[28]	ILP	CPU, BW, and Delay	No	No	No
[29]	ILP	CPU, BW	No	No	No
[18]	MILP	CPU, RAM, BW, and PD	No	No	No
[19]	Heuristic	CPU, RAM, and BW	No	No	No
[21]	Meta-heuristic	Storage	No	No	No
[22]	Genetic Algorithm	CPU, RAM and Storage	No	No	No
[24]	QL	CPU, BW	No	No	No
[25]	QL	CPU, RAM, Storage	No	No	No
[8]	QL	CPU, RAM, BW, Latency	Yes	No	No
[9]	DQL	CPU, RAM, BW, Latency	Yes	No	No
[27]	DQL	CPU, RAM, BW, Latency	No	Yes	No
Our Model	DDQL	CPU, RAM, BW, Latency	Yes	Yes	Yes

Rel. CPU and RAM, BW, and PD represent Relationship between the CPU and RAM, Bandwidth and Propagation Delay, respectively

decomposed VNFs but did not explain the micro-segmentation criteria for each VNF. Similarly, the authors of [26] also assumed that the VNF provider had already acquired the details of the decomposed VNFs. Thus, in the above works, the authors' have pre-acquired knowledge on the structure of the decomposed VNF or *mVNF*-FGs; nevertheless, this is not a viable option for online VNF mapping when the arrival of VNF or the SFC type is unknown to the network. As a result, with respect to these works in this study, we propose a dynamic decomposition method based on network availability; the detailed description is presented in the later section.

Furthermore, in the literature, most authors employed the shortest-distance technique to discover the path between the substrate nodes to deploy the VNFs and *mVNFs*, resulting in a highly uneven load on certain links. Thus, to maintain equilibrium across the topology, our studies [8] and [9] opt for a practical link-selection technique and consider a more realistic relationship between the CPU and RAM (nodal resource), which is embraced in this work also. According to [8] (our initial study), over here, the services are deployed using QL models to account for various nodal failures and communication delays between the two VNFs, such as 30 ms, 50 ms, and 100 ms and network density. From this experiment, we observed as the network density and complexity increased, the model was affected by the curse of dimensionality, which caused inadequate performance. In contrast, the [9] is formulated from the drawbacks of the [8] model. In [9], the Deep Q Learning (DQL) model performs the deployment, and the model's performance is examined for a) various nodal capacity, b) VNF complexity, c) nodal outage and d) network density. During this analysis, we came across the overestimation problem caused by the DQL model. Solving this would improve service acceptability. Thus, this work adopts the advanced DRL model, DDQL, which overcomes the drawbacks given by the DQL model in [9].

The primary goal of this study is to improve model learning by implementing an enhanced DQL version called DDQL that eliminates the over-estimation problem caused by QL and DQL models. Our decomposition model first uses analytical criteria for discovering the potential/candidate VNFs

for decomposition from the arriving network services and later determines the degree of micro-segmentation. The current availability of the network and requested resources by VNFs define these analytical criteria, making it suitable for an independent and intelligent zero-touch system. Thus, our decomposition occurs online (say dynamically), unlike in the literature, where the structure of the decompositions are static or pre-defined by the engineers. However, our model does not have any prior information on the decomposition structure for the potential/candidate VNF, making this model more pragmatic. Table 1 summarizes related works.

### III. PROBLEM STATEMENT

This section will introduce the NFV-RA optimization problem. The arrival of SFCs is assured in a discrete time-step (time-slotted) fashion, i.e., at each time-step, one SFC is scheduled for deployment over the substrate network. The requested SFC is discarded if the substrate network cannot deploy the SFC by the end of the time-step. In the case of decomposed VNFs, the candidate VNF is decomposed into numerous *mVNFs* based on the granularity criteria (described later). These decomposed *mVNFs* and *mVLs* are represented graphically as micro-VNF-FG (*mVNF-FG*). For a successful *mVNF-FG* deployment, the model must adhere to the same constraints as the VNF-FG. As a result, the primary goal is to promote service acceptance by effectively allocating resources to the SFCs.

#### A. OBJECTIVE

The objective of our proposed model is to maximize the placement of the online SFCs or SAR in the network so that all the VNFs and VLs of an SFC are embedded in accordance with the promised QoS.

#### B. CONSTRAINTS

For a successful deployment, the SFC must meet all defined constraints. These constraints are as follows.

##### 1) VN MAPPING

The requirements for each VNF differ depending upon the SFC type [16]. Some VNFs, like Firewall and Deep Packet

Inspection, may require more CPU and RAM resources, while others require less. According to the requested resources, the provisioning of appropriate resources by the substrate nodes is essential for a successful VNF deployment.

Similarly, each  $m$ VNF demands for a comparatively smaller amount of resources, which will be embedded onto the substrate node. Again, for a successful deployment, the substrate node should satisfy the requested resources.

### 2) VL MAPPING

During the deployment, it is critical to analyze the availability of the links to provide the promised QoS to the users. Inappropriate VL/ $m$ VL mapping can lead to inadequate network performance. Thus, the VL/ $m$ VL is successfully deployed if the substrate link meets the requested link requirements, such as bandwidth and latency.

### 3) SEAMLESS PATH

According to the promised QoS, these VLs/ $m$ VLs should form a continuous path between the head VNF/ $m$ VNF to the end VNF/ $m$ VNF. This discovered path should not have loops. For example, if head VNF/ $m$ VNF and end VNF/ $m$ VNF are placed on the same substrate node, there is a high possibility of unnecessary loop creation. To avoid such instances, we used constraint 7 from [16]. The VNFs/ $m$ VNFs have the freedom to be deployed on the same substrate node, and hence the intra-communication between them on the same substrate node happens through a vSwitch or bridge.

### 4) LATENCY

Our study focuses on latency-sensitive applications, where a candidate path for a VL is chosen according to latency requirements. The selection of the optimal path from all the possible paths is performed based on a present latency limit. In the case of  $m$ VNF-FGs, we assumed the upper bound latency from the head  $m$ VNF to the tail  $m$ VNF to be 50 ms and 100 ms for the case of monolithic VNF-FGs. Instead of concentrating on a specific type of communication like enhanced Mobile Broadband (eMBB), Ultra-Reliable Low Latency Communications (URLLC), or massive Machine-Type Communications (mMTC), our proposed model addresses services generated by all types of communications. These upper bound parameters were arbitrarily chosen for experimental purposes since our proposed model is versatile and can be modified accordingly, providing a degree of freedom for future communication requirements. This is a general representative problem formulation for NFV-RA, which we investigated in [9]. In this study, we have adapted the framework of [9] for microservices-based VNF deployment and used the advanced DRL method to solve it.

## IV. DEEP RL SOLUTION FOR MICROSERVICE DECOMPOSITION

### A. REINFORCEMENT LEARNING

RL is a self-learning process in which the agent (decision-maker) observes a state  $s_t \in S$  from the environment at each

iteration (time-step  $t$ ) and responds with an appropriate action  $a_t \in A$  by applying a  $\pi_t$  policy. Based on the selected action, the environment provides numerical reward  $r_t \in R \subset \mathbb{R}$  and causes a new state  $s_{t+1}$ . The agent's primary objective is to maximize the total rewards (expected discounted return  $R_t$ ), which promote the model's training even during the network dynamism. This expected discounted return is defined as the sum of the received discounted rewards:

$$R_t = \sum_{k=t}^T \gamma^{k-t} r_k \quad (1)$$

where  $T$  is the terminal state and  $\gamma$  is the discount rate that defines the current value of the future rewards. The discount rate, which ranges from  $0 \leq \gamma \leq 1$ , is a critical learning parameter since it influences the agent's behaviour. When  $\gamma$  is at zero, the agent focuses on maximizing the immediate rewards for choosing the action, i.e., the agent acts as myopic. As  $\gamma$  approaches 1, the agent becomes more farsighted, considering the importance of future rewards [30]. In our model, the agent's current activity impacts the future; as a result, we have chosen a 0.99 discount rate for our studies. In this work, the optimization problem is formulated as a Markov Decision Process (MDP), and MDP establishes a mathematical formation (functions) for exploring the decision-making problems. This decision-making process is partially random and partially controlled by the agent. The agent aims to improve the decision strategy by achieving an optimal policy  $\pi^*$  to attain an optimal solution based on the received observations (i.e., state, action, rewards, and next state), i.e.,  $\pi^* = S \rightarrow A$ , where  $A$  is the action taken for the state  $S$ . This optimal policy is estimated by maximizing the optimal action-value function, i.e.,

$$\pi^*(s_t) = \arg \max_a Q^*(s_t, a_t) \quad (2)$$

and this action-value function is expressed using the Bellman Equation, i.e.,

$$Q^\pi(s_t, a_t) = \sum_{s_{t+1}} P(s_{t+1}, r_t | s_t, a_t) (r_t(s_t, a_t) + \gamma \sum_{a_{t+1}} \pi(a_{t+1} | s_{t+1}) (Q^\pi(s_{t+1}, a_{t+1}))) \quad (3)$$

where,  $s_t$ ,  $a_t$ , and  $r_t$  are the state, action, and reward achieved at time-step  $t$ , respectively.  $s_{t+1}$  and  $a_{t+1}$  are the state and action for the next time-step  $t + 1$ , whereas  $P(s_{t+1}, r_t | s_t, a_t)$  is the probability of the next state for a given current state and action. In addition,  $\pi(a_{t+1} | s_{t+1})$  is the probability of selecting the next action under a  $\pi$  policy with the  $\gamma$  discounting factor.

Because the environment's behavior is unpredictable in our study, measuring the transition probability will be challenging; thus, learning from previous as well as current experiences will be beneficial. As a result, we are considering Q-Learning, a model-free off-policy method, where the Target policy<sup>2</sup> learns from the Behaviour policy.<sup>3</sup> From the

<sup>2</sup>The agent estimates the action-value using Target policy.

<sup>3</sup>The agent uses this policy to decide which action to take.

literature, RL has been proven to be the most efficient approach for NP-Hard problems [15], [31]. As a result, Equation 3 is changed to,

$$Q^\mu(s_t, a_t) = r_t(s_t, a_t) + \gamma \max_a Q^\mu(s_{t+1}, a; \theta_t) \quad (4)$$

The above equation depicts that the most competent action has the maximum estimated action-value (Q-value) using the  $\mu$  policy; this is the fundamental principle behind the basic QL. Since the agent is unaware of the environment, it learns from each iteration and updates the knowledge accordingly.

In Equation 4, the maximization function (single estimator with  $\theta$  parameters, that is, weights and biases) selects and evaluates actions using the same parameters, which creates a lot of noise and divergence in Q-values. In other words, it chooses an action with an over-optimistic Q-value. This is called the Overestimation problem. Thus, the achieved action may not be ideal for the observing state as the agent has selected a non-optimal action. The considerable noise in Q-value leads to significant positive biases, which will disrupt the updating and learning procedures.

We have opted for the Double QL [32] method to remove this overestimation. The Double QL decouples the single estimator into two different estimators;  $Q, Q'$ . As in basic QL, the  $Q$  estimator is utilized to discover the optimal action by applying the max operator with the online parameters  $\theta_t$ . At the same time, the  $Q'$  estimator uses a new set of parameters  $\theta'_t$  to calculate the estimated Q-value for the selected action, i.e., evaluating the current policy. Equations 5 and 7 are the target estimate for the basic QL and the Double QL, respectively and Equations 6 and 9 are the Q-value updating procedure for QL and Double QL, respectively. To achieve the goal, the convergence of the Q-value to the true action-value is a must. The mean squared error loss function (Equation 10) is used to acquire convergence by minimizing the distance between the target and estimated Q-values.

Basic QL:

$$y_t = R_t + \gamma \max_a Q(s_{t+1}, a; \theta_t) \quad (5)$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(y_t - Q(s_t, a_t)) \quad (6)$$

Double QL:

$$y_t = R_t + \gamma Q'(s_{t+1}, \mathbf{a}; \theta'_t) \quad (7)$$

$$\mathbf{a} = \max_a Q(s_{t+1}, a; \theta_t) \quad (8)$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(y_t - Q(s_t, a_t)) \quad (9)$$

$$Loss = (R_t + \gamma Q'(s_{t+1}, \mathbf{a}; \theta'_t) - Q(s_t, a_t; \theta_t))^2 \quad (10)$$

To solve this NFV-RA problem, we consider the DDQL model with two different DNNs; Primary DNN ( $Q$  estimator) and Target DNN ( $Q'$  estimator). The Target DNN has the same architecture as the Primary DNN, but the parameters (such as weights and bias) are updated smoothly at every  $\tau$  iterations. This reduces the correlation between the DNNs and achieves a stable learning. To further improvise our learning, we adopt the Experience Replay process, where all

the observed transitions like state, action, rewards, and next state are stored in a memory buffer. These stored transitions are uniformly sampled and given to the model for training purposes.

The DDQL process, including decomposition and re-architecture, is depicted in Figure 3, and each component of the process is described in detail below.

## B. ENVIRONMENT

The environment has been defined as the physical topology on which the VNFs/mVNFs will be deployed; this topology comprises high-volume servers with assigned processing resources. We modelled the topology as a directed graph  $G = (H, N)$ , where  $H$  and  $N$  represent the set of physical nodes and physical links of the topology, respectively. Let  $h$  denotes the amount of nodal resources like CPU core, RAM, memory, etc, and  $J_{node}$  denotes the number of nodal resource types which is indexed from 0, 1, ...,  $J_{node} - 1$ . Thus the available resource on the physical node  $y$  is denoted as  $\mathbf{h}_y = [h_{y,0}, \dots, h_{y,J_{node}-1}]$ , where  $h_{y,w}$  represents the amount of resource type  $w$  in the physical node  $y$ . Similarly, each  $n$  signifies the amount of link resources like bandwidth, latency, packet loss rate, jitters, etc.  $J_{link}$  stands for the number of link resource types, the available resource for physical link  $u$  is  $\mathbf{n}_u = [n_{u,0}, \dots, n_{u,J_{link}-1}]$ , where  $n_{u,w'}$  is the amount of available  $w'$  resource type on a physical link  $u$ . All the notations are described in Tables 5 and 6.

## C. STATE SPACE

Each SFC is represented in a graphical form (VNF-FG) as a directed graph  $G' = (V, B)$ , where  $V$  and  $B$  are the set of VNFs and VLs, respectively, with their corresponding resource requirements. Each  $v$  and  $b$  denote the requested nodal resources (CPU core, RAM) and link resources (latency, bandwidth), respectively. Similarly, in the case of decomposition, each potential VNF ( $\varrho$ ) candidate is decomposed and represented as a directed graph  $G''_\varrho = (K_\varrho, D_\varrho)$ , where  $K_\varrho$  and  $D_\varrho$  are the set of mVNFs and mVLs, respectively. The state-space for the requested SFC  $\Psi$  is represented as  $(\mathbf{V}_\Psi, \mathbf{R}_\Psi, \mathbf{K}_\Psi, \mathbf{L}_\Psi)$ .  $\mathbf{V}_\Psi = [v_{\Psi,0}, \dots, v_{\Psi,|V|-1}]$  denotes the sequentially ordered VNFs and  $\mathbf{R}_\Psi = [r_{\Psi,0}, \dots, r_{\Psi,|V|-1}]$  indicates the vector of computing resources requested by the corresponding VNFs, where  $|V|$  is the length of SFC  $\Psi$ . The  $r_{\Psi,x}$  represents the details of requested resources by the VNF  $x$  of the SFC  $\Psi$ . Further,  $r_{\Psi,x}$  is expressed as  $r_{\Psi,x} = [r_{\Psi,x,0}, \dots, r_{\Psi,x,J_{node}-1}]$ , where  $r_{\Psi,x,w}$  describes the amount of type  $w$  resource requested by the VNF  $x$  of SFC  $\Psi$ .  $\mathbf{K}_\Psi$  and  $\mathbf{L}_\Psi$  describe the state of VNFs in terms of microservices (mVNFs) and their demanded resources, respectively.  $\mathbf{K}_\Psi = [k_{\Psi,0}, \dots, k_{\Psi,|V|-1}]$ , each  $k_{\Psi,\varrho}$  determines the vector of finely-decomposed VNF ( $\varrho$ ) sequences, that is.,  $k_{\Psi,\varrho} = [k_{\Psi,\varrho,0}, \dots, k_{\Psi,\varrho,|K_\varrho|-1}]$ , where  $|K_\varrho|$  is the length of decomposed mVNFs.  $\mathbf{L}_\Psi = [l_{\Psi,0}, \dots, l_{\Psi,|V|-1}]$ , again,  $l_{\Psi,\varrho} = [l_{\Psi,\varrho,0}, \dots, l_{\Psi,\varrho,|K_\varrho|-1}]$ , depicts set of the requested resource by the mVNFs of the VNF ( $\varrho$ ). For example, for the

$m$ VNF ( $z$ ), the  $\mathbf{l}_{\Psi,\rho,z} = [l_{\Psi,\rho,z,0}, \dots, l_{\Psi,\rho,z,J_{node}-1}]$  provides detailed information on the requested resource amount, where  $l_{\Psi,\rho,z,w}$  defines the amount of requested resource of type  $w$  by the  $m$ VNF  $z$  of VNF  $\rho$ .

Most related works have allocated computing resources randomly when it comes to resource initialization, neglecting the reality that resources like CPU core and RAM are highly correlated. As a result, in our work, we employ the relationship between CPU core and RAM as described in [33]. In this work, the  $J_{node}$  is 1, slightly reducing the state-space complexity without overly simplifying the system model and maintaining the integral correlation between the resources. And  $J_{link} = 2$ , i.e., Latency and Bandwidth. We will study the constructed model's performance for other resource types in our future work.

### D. ACTION SPACE

The action space ( $\mathbf{A}$ ) is defined as the total number of physical nodes/servers present in the topology. For a SFC ( $\Psi$ ) the action-space is defined as  $\mathbf{A}_{\Psi} = [\mathbf{A}_{\Psi}^{vnf}, \mathbf{A}_{\Psi}^{mvnf}]$ .  $\mathbf{A}_{\Psi}^{vnf} = [a_{\Psi,0}^{vnf}, \dots, a_{\Psi,|V|-1}^{vnf}]$  represents the action-space for monolithic VNFs, where for each VNF ( $x$ ) the action-space is  $\mathbf{a}_{\Psi,x}^{vnf} = [a_{\Psi,x,0}^{vnf}, \dots, a_{\Psi,x,|H|-1}^{vnf}]$ .

The dimensional complexity of the action-space size increases with the introduction of microservices. Let us consider, an overall action-space vector for an SFC, represented as  $\mathbf{A}_{\Psi}^{mvnf} = [a_{\Psi,0}^{mvnf}, \dots, a_{\Psi,|V|-1}^{mvnf}]$ ,  $|\mathbf{A}_{\Psi}^{mvnf}|$  is the service-chaining length.  $\mathbf{a}_{\Psi,x}^{mvnf} = [a_{\Psi,x,0}^{mvnf}, \dots, a_{\Psi,x,|K_x|-1}^{mvnf}]$  vector describes the sequentially decomposed  $m$ VNFs per VNF,  $|\mathbf{a}_{\Psi,x}^{mvnf}|$  is the fine-granularity of  $m$ VNFs. For each  $m$ VNF ( $f$ ), the action-space will be  $\mathbf{a}_{\Psi,x,f}^{mvnf} = [a_{\Psi,x,f,0}^{mvnf}, \dots, a_{\Psi,x,f,|H|-1}^{mvnf}]$ .

### E. REWARD FUNCTION

The environment rewards the agent only if the chosen actions allow the VNFs/ $m$ VNFs and VLs/ $m$ VLs to be successfully deployed on them; otherwise, the agent receives a strict penalty. As mentioned earlier, the reward function is a crucial part of learning; since the environment provides feedback at each movement. To further describe, for each SFC, the reward function ( $R(\Psi)$ ) is comprised of two sub-reward functions: Local ( $L_{reward}(\Psi_{vnf/mvnf})$ ) and Global ( $G_{reward}(\Psi)$ ) reward function as in Equation 11. These sub-reward functions indicate the model's performance for each VNF/ $m$ VNF and for the overall VNF-FG/ $m$ VNF-FG.

$$R(\Psi) = L_{reward}(\Psi_{vnf/mvnf}) + G_{reward}(\Psi_{VNF-FG/mVNF-FG}) \quad (11)$$

The quantity of the generated local rewards are based on the agent satisfying the constraints in III-B, which indicates the quality of the agent's decision. These local reward functions for VNFs/ $m$ VNFs and VLs/ $m$ VLs are mentioned in Equations 12, 14, 19, and 24.  $\Phi_x^y$  and  $\Phi_i^w$  are the binary variables which represent the deployment status of VNF/ $m$ VNF ( $x$ ) onto the substrate node ( $y$ ) and VL/ $m$ VL ( $i$ ) onto the substrate link ( $w$ ), respectively. On successful placement

(i.e., satisfying the constraints III-B1-III-B4) the decision  $\Phi$  variable is 1, else 0.

Furthermore, the global reward for the VNF-FG/ $m$ VNF-FG is based on the definite continuous path between the VNFs and subject to latency, and other mentioned constraints in Section III-B. The successful deployment of a VNF-FG/ $m$ VNF-FG is checked using the conditions represented in the Equations 17, 18, 22, and 23. The constructed reward functions are expressed based on the points system notion, in which the environment rewards the agent with high positive points for each successful action and negative points (penalty) for each failed action. This technique assists the agent in selecting a better solution.

Local reward function for VNF:

$$L_{reward}(\Psi_{vnf}) = \sum_{x=0}^{|V|-1} L_{reward}(x) \quad (12)$$

$$L_{reward}(x) = \begin{cases} R_{vnf}^{pt}, & \text{if } \Phi_x^y = 1 \\ P_{vnf}^{pt}, & \text{otherwise} \end{cases} \quad (13)$$

Similarly, local reward function for  $m$ VNF:

$$L_{reward}(\Psi_{mvnf}) = \sum_{x=0}^{|K|-1} L_{reward}(x) \quad (14)$$

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

Global reward for VNF-FG ( $G'$ ):

$$G_{reward}(\Psi_{G'}) = \begin{cases} GR_{G'}^{pt}, & \text{if } C1 \ \& \ C2 \ \text{are satisfied} \\ 0, & \text{otherwise} \end{cases} \quad (16)$$

$$C1 : |V| \times R_{vnf}^{pt} = L_{reward}(\Psi_{vnf}) \quad (17)$$

$$C2 : |B| \times R_{vl}^{pt} = L_{reward}(\Psi_{vl}) \quad (18)$$

$$L_{reward}(\Psi_{vl}) = \sum_{i=0}^{|B|-1} L_{reward}(i) \quad (19)$$

$$L_{reward}(i) = \begin{cases} R_{vl}^{pt}, & \text{if } \Phi_i^w = 1 \\ P_{vl}^{pt}, & \text{otherwise} \end{cases} \quad (20)$$

Similarly, Global reward for  $m$ VNF-FG ( $G''$ ):

$$G_{reward}(\Psi_{G''}) = \begin{cases} GR_{G''}^{pt}, & \text{if } C3 \ \& \ C4 \ \text{are satisfied} \\ 0, & \text{otherwise} \end{cases} \quad (21)$$

$$C3 : |K| \times R_{mvnf}^{pt} = L_{reward}(\Psi_{mvnf}) \quad (22)$$

$$C4 : |D| \times R_{mvl}^{pt} = L_{reward}(\Psi_{mvl}) \quad (23)$$

$$L_{reward}(\Psi_{mvl}) = \sum_{i=0}^{|D|-1} L_{reward}(i) \quad (24)$$

$$L_{reward}(i) = \begin{cases} R_{mvl}^{pt}, & \text{if } \Phi_i^w = 1 \\ P_{mvl}^{pt}, & \text{otherwise} \end{cases} \quad (25)$$

As mentioned previously, the Shortest-Distance algorithm is adopted in most works to determine the path between the VNFs. This technique induces biases leading to significant congestion on a few connections resulting in a prolonged link delay. Especially for delay-sensitive applications, this technique will not be practical. To address this, we looked into a way of determining the path per VNF-FG based on the latency's upper bound. We set this limit to 100 ms for VNF-FGs and 50 ms for  $m$ VNF-FGs for our investigation; however, the constructed model can be trained for different latency values. The reward is delivered to the agent when the end-to-end path is built under the latency constraint for VL/ $m$ VL. As a result, the reward function is built based on the placement of the VNFs and VLs on the appropriate substrate nodes and links.

The placement of VLs is summarized in Algorithm 2. Following the deployment of VNFs and  $m$ VNFs, the algorithm, as shown in line 4, determines all feasible paths between the source VNF/ $m$ VNF and the destination (end) VNF/ $m$ VNF of an SFC. Later, each path is iterated to discover the constrained fulfilling path; on success, incentives are given; otherwise, penalties are applied, as specified from lines 5 to 17. Further, these transitions are stored in the memory buffer for better learning (line 18).

#### F. DECOMPOSITION IDENTIFIER

The benefits of moving from a monolithic system to decomposed VNFs were qualitatively described in the prior sections. The performance of the microservice-based approach would not be profitable if all the arriving VNFs were decomposed constantly since it will increase the architectural and computational complexity by demand more link resources like bandwidth and latency than expected, as presented in Figure 2. Therefore, the main goal is to find that potential VNF (say candidate VNF) for decomposition, which would benefit the optimization problem by increasing the service acceptance quality rather than degrading the performance. To do so, each VNF is verified against a decomposition identification module to examine if it qualifies as a prospective candidate for dynamic decomposition. One of the considered criteria is when the agent ( $\Xi$ ) is unable to find a suitable/satisfying substrate node. This can be due to low network resource availability as compared to the demanded resource  $r_{\psi,x}$ , as in Equation 27.  $\mathcal{R}(e_{\psi,x}^{vnf}) = [e_{\psi,x,0}^{vnf}, \dots, e_{\psi,x,J_{node}-1}^{vnf}]$  is the current resource availability of the action ( $e_{\psi,x}^{vnf}$ ), where  $e_{\psi,x}^{vnf}$  represents the action (substrate node) achieved from the agent for VNF ( $x$ ) for all resource type.  $E_{\psi}^{vnf}$  and  $E_{\psi}^{mvnf}$  are the achieved action vectors from the agent for the monolithic VNFs, and decomposed VNFs, respectively, for an SFC ( $\Psi$ ), as in Equation 26.  $E_{\psi}^{vnf} = [e_{\psi,0}^{vnf}, \dots, e_{\psi,|V|-1}^{vnf}]$ , where  $e_{\psi,x}^{vnf}$  is the obtained action for VNF ( $x$ ).  $E_{\psi}^{mvnf} = [e_{\psi,0}^{mvnf}, \dots, e_{\psi,|V|-1}^{mvnf}]$ , where  $e_{\psi,\varrho}^{mvnf}$  describes the VNF's ( $\varrho$ ) decomposition status.  $e_{\psi,\varrho}^{mvnf} = [e_{\psi,\varrho,0}^{mvnf}, \dots, e_{\psi,\varrho,|K_{\varrho}|-1}^{mvnf}]$ , where  $e_{\psi,\varrho,f}^{mvnf}$  is the action selected by the agent for  $m$ VNF ( $f$ ) of potential VNF

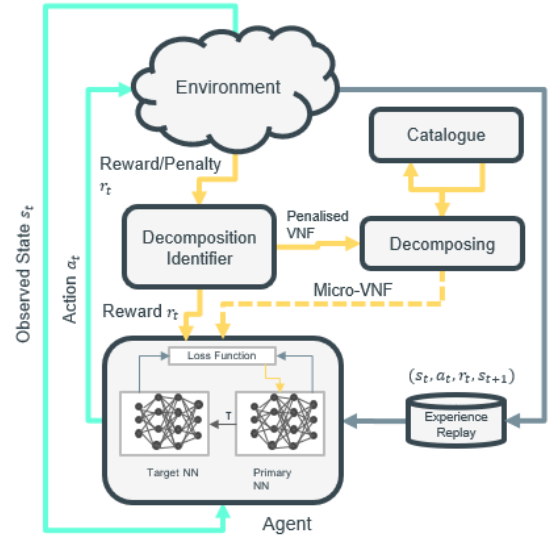


FIGURE 3. RL-based NFV-RA with microservices decomposition.

candidate ( $\varrho$ ). This way, only essential VNFs are decomposed, optimizing the resource allocation efficiently.

$$\Xi(V_{\psi}, R_{\psi}, K_{\psi}, L_{\psi}) = [E_{\psi}^{vnf}, E_{\psi}^{mvnf}] \quad (26)$$

$$r_{\psi,x} > \mathcal{R}(e_{\psi,x}^{vnf}) \quad (27)$$

#### G. GRANULARITY CRITERIA

Following the identification of the potential VNF, the next consideration is how finely these VNFs should be decomposed? Instead of assuming the operator's static (pre-defined) decomposition model (as in literature), we propose a dynamic granularity criterion. This granularity criterion regulates the magnitude of micro-segmentation of the potential VNF, which is expressed in terms of the Granularity Index (GI). The estimation of the GI is based on two parameters: 1) the current VNF's requested processing resource (CPU core); 2) the Network Availability Index (NAI). To perform the dynamic decomposition, we require actively updated network conditions, i.e., the current network resource availability, NAI ( $\Delta$ ). The NAI is the ratio of currently available processing resource in a topology  $\varphi_{topology,t}$  at each time-step to total initial processing resource in a network, i.e., maximum nodal capacity  $\zeta_{topology}$ , which the model estimates at each time step, as defined in Equation 28. The NAI ranges between 0 and 1.

$$\Delta_t = \frac{\varphi_{topology,t}}{\zeta_{topology}} \quad (28)$$

With 0 being an utterly exhausted network and 1 depicting a fully available network (a plethora of resources).

The GI ( $\Gamma_t^V$ ) is represented as the product of the currently requested processing resource (CPU cores) by the VNF ( $R_V$ ) to the NAI ( $\Delta_t$ ), as in Equation 29.

$$\Gamma_t^V = R_V \times \Delta_t \quad (29)$$



NAI	1.0	0.9	0.8	....	0.2	0.1	0
GI for 4 CPU cores	4.0	3.6	3.2	....	0.8	0.4	0
GI for 5 CPU cores	5.0	4.5	4.0	....	1.0	0.5	0




FIGURE 4. Granularity criteria.

Figure 4 shows the GI for a VNF with 5 CPU cores resources. When the NAI is 0.8 (i.e., 20% of the network's processing resource is utilized, and only 80% is remaining), and the GI is 4 ( $0.8 \times 5$  CPU cores), the micro-segmentation more or less will be coarse, perhaps the VNF will be split into 2, or 3 VNFs with (2,3) or (4,1), or (3,1,1) CPU core(s) as requested processing resources. As the NAI reduces, the GI decreases, too, instructing the model towards finer-granularity. For example, when NAI is 0.2, the GI is 1 ( $0.2 \times 5$  CPU cores), implying that VNF can be split into 5 *m*VNFs with 1 CPU core each. The GI thus determines the granularity of decomposition. With the decrement of GI, the granularity of the VNFs increases, indicating that finer micro-segmentation is required when network resources are scarce.

#### H. RE-ARCHITECTURE OF VNF-FG

Each decomposed *m*VNF has an independent functionality, and these micro-functionalities can be repeated over a decomposed SFC. From the example given in Figure 5, the SFC comprises five VNFs (presented vertically): WAN optimizer, Edge Firewall, Monitoring Functions, Application Firewall, and Load Balancer [7] chained in a specific order. Further, each VNF, such as WAN Optimizer, is decomposed into *m*VNFs like 'Read from NIC', 'Parse Header', 'Classify on L7 Type', 'Decompressed HTTP Payload', and 'Send to NIC', as shown (horizontally in the figure).

From Figure 5, the decomposed SFC indicates that several functionalities are repeated throughout the SFC, such as 'Read from NIC', 'Parse Header', 'Classify on L7 type', and so on. These repeated micro-functionalities are overlapping/redundant functions; deploying such functions will prompt the consumption of extra overheads/CPU cycles. Instead, sharing these redundant functions can eliminate unnecessary resource usage. As shown in the example, overlapping *m*VNFs like 'Read from NIC' are prevalent and embedding this *m*VNF frequently will consume additional resources; sharing such *m*VNFs will be beneficial. Identifying and eliminating such redundant functions is advantageous in decreasing resource consumption; thus, constructing such models without these redundant functions is highly desirable. Figure 6 shows an example of re-architecture of the above mentioned SFC, in which overlapping functions are detected and removed, and accordingly, the decomposed SFC is re-structured without any disturbance in the flow.

Therefore, in our research, we develop an identification and re-architecture model. After the completion of

decomposition, the identification model inspects the existence of this newly created *m*VNFs in the network. On success, the model retrieves the desired information, such as its deployed location (substrate node) from the repository and the re-architecture model is triggered, indicating the need for re-architecture of *m*VNF-FG. Rather than placing the identical *m*VNF onto the network, the present *m*VNF is connected to the precedent *m*VNF to construct a *m*VNF-FG that meets the promised QoS requirements. On an unsuccessful attempt, the newly created *m*VNF details (resource requirements) are fed to the DDQL agent to achieve a suitable substrate node for successful deployment. These details are saved in a catalogue for future reference. This optimizes the nodal and link resources, as will be evident in the simulated results.

#### I. OVERVIEW OF THE PROPOSED MODEL

Algorithm 1 describes an overview of the proposed model. Lines 1 to 4 defines the initialization of the necessary attributes for constructing the Primary and Target Neural Network (NN) and replay memory for stable learning. At the start of each episode, the environment is restored to its original state with abundant resources, as seen in lines 5-7. Once the agent observes the state based on it, the action is determined using the  $\epsilon$ -Greedy Exploration-Exploitation strategy (lines 9-18). The agent earns rewards or penalties depending on whether the chosen action satisfies or dissatisfies the demanded resources (lines 19-31). On an unsuccessful attempt, the model inspects for decomposition opportunities. If possible, the model initiates the decomposition phase; otherwise, a penalty is applied (lines 21-29). All these transitions are stored in the replay buffer (line 33). Later, target Q values and loss are estimated (lines 38-39), and finally, the primary and target NN are updated at the defined rate (lines 40-42). Our model's performance is not confined to the parameters provided, but it can also be trained for other use-cases such as URLLC or eMBB.

#### V. DEEP NEURAL NETWORK ARCHITECTURE

A proper NN architecture is just as critical as designing the model's other attributes like states, actions, environment, and reward functions. The NN architecture must be built based on the dynamicity and complexity of the problem in order to extract all of the characteristics (features) from the input to deliver an optimum solution. The quantity of feature extraction is controlled by two hyper-parameters: (1) the depth of the NNs, i.e., the presence of hidden layers in a NN, and (2) the width of NN, i.e., the number of neurons in each layer. A methodical experiment is performed because the depth of the NNs cannot be determined through an analytical calculation. Hence, this section explains the influence of the fully connected hidden layers in our model.

For our investigation, we used the DDQL model for deploying 100 SFCs on the BtEurope topology, with hidden layers ranging from 2 to 10, as shown in Figure 7. Further, each SFC is comprised of 5 VNFs with various resource requirements.

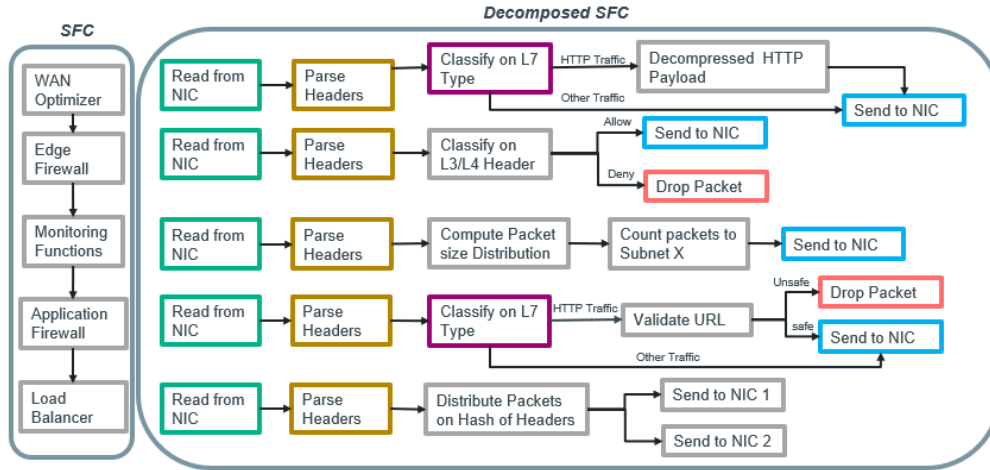


FIGURE 5. Decomposed VNF-FG.

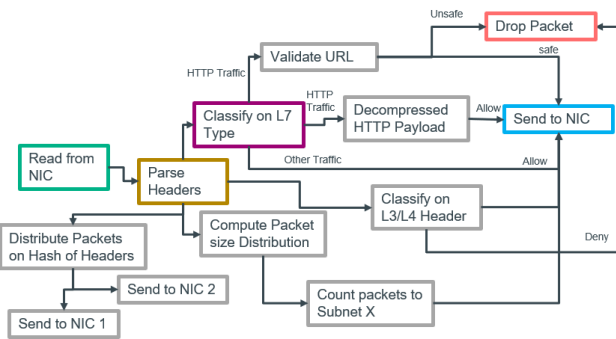


FIGURE 6. Decomposed VNF-FG after re-architecture.

From Figure 7, we can observe that, as the layers (depth) increase, the model’s performance increases, but at the cost of higher computation time. From the learning perspective, the stability in learning occurs at a different stage based on the depth of NNs. For shallower NNs (2,4 hidden layers), the learning begins only after 200 episodes, whereas for deeper NNs (6,8,10 hidden layers), the learning happens considerably earlier. Like shallow layer’s overall performance in terms of the mean of normalized SAR is low compared to the higher layers, indicating that lower layers failed to extract features from the input to provide adequate placement of the SFCs. On the other hand, the number of layers equal to 6, 8, and 10 have a higher performance quality (i.e., the model extracts the features more accurately) but at the cost of a higher runtime. As the depth goes beyond 6, the model provides comparable results. Hence, our study used a depth of 6 to strike a balance between performance and computation.

**VI. SIMULATION RESULTS**

This section discusses the performance of our proposed DRL-based approach across various network topologies. We started our investigation with a simpler topology, Netrail, which has 7 nodes and 10 full-duplex links, and then

progressed to a denser network, BtEurope, which possesses 24 nodes and 37 full-duplex links [34]. For these topologies, we analyzed the performance of Monolithic and Microservice-based SFC placement under various nodal capacities.

At each run, we assumed that each substrate node would commence with the nodal capacity (processing resource) of 12 cores with 4 CPUs (a total of 48 CPU cores) or 8 CPUs (a total of 96 CPU cores). These nodal capacity scenarios can be extended even further for deeper analysis. The initialization of substrate link resources is based on uniform distribution, with link capacity (bandwidth) ranging from 1 to 100 Gbps and link delay (latency) ranging from 0 to 10 ms.

The Erdős-Rényi (ER) model [35] is considered to generate the VNF-FGs, with the  $\epsilon$  of 0.3, where the nodes (VNFs) are connected in a probabilistic manner, introducing diversity to each VNF-FG. This complexity of VLs (or, say, the probability of connectivity  $p$ ) is conditioned on the number of VNFs per VNF-FG ( $\Upsilon$ ) and the  $\epsilon$ , i.e.,  $p = \frac{(1+\epsilon)\log \Upsilon}{\Upsilon}$ . To augment the diversity in the VNF-FGs, the directions of these VLs are selected randomly by the ER model, resulting in a wide variety of VNF-FGs that enhances NN learning. In VNF-FGs, the VL resource is initialized using a uniform distribution with bandwidth ranging from 1 to 10 Gbps. In our case, we selected a constant number of 5 VNFs per SFC to analyze the complex problem efficiently. The processing resource (CPU core) initialization for a VNF is performed based on a normal distribution with a mean of 3 and a standard deviation of 0.4, i.e., each VNF requests a processing resource between 1 to 5 CPU cores.

Each run consists of 3000 episodes with 100 time-steps per episode, and each time-step produces a distinct VNF-FG with a diverse set of resource requirements, as mentioned above. The DDQL 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

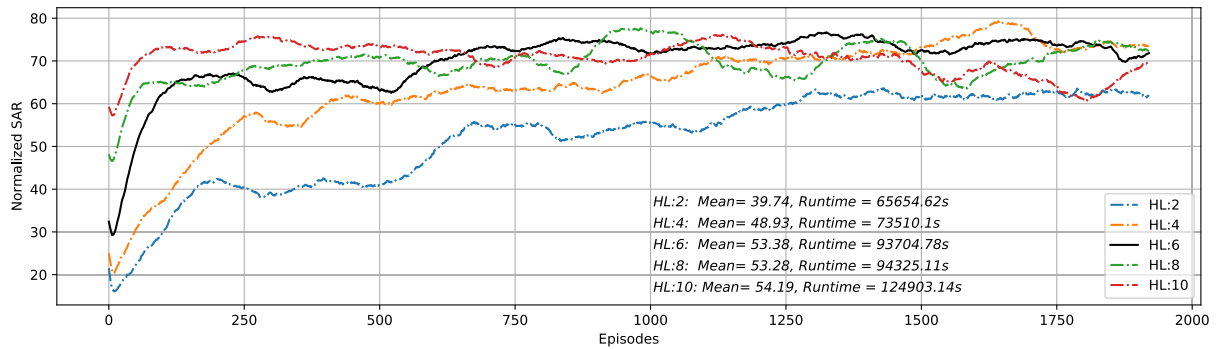


FIGURE 7. Performance of DDQL model with various Hidden Layers (HL).

the parameters applied to develop the DDQL model. Due to the increased complexity of the problem, for a deeper investigation, we have selected a DNN of 6 hidden layers, as explained in Section V.

To outline the advantages of the DDQL model for the embedment of microservices, we introduce a bound called Expected Upper Bound (EUB). An EUB determines an upper bound of the expected number of SFCs a network topology can accommodate, which is assessed based on network topology and nodal capacity scenario. The EUB is calculated as

$$EUB = \frac{\zeta_{topology}}{\vartheta \times \Upsilon} \quad (30)$$

where  $\zeta_{topology}$  is the maximum nodal capacity of a topology, as described in Table 3.  $\vartheta$ ,  $\Upsilon$  are the average CPU cores requested by a VNF and the number of VNF per SFC, respectively.

The obtained results are presented in two formats: 1) A moving average of normalized SAR over the last 50 episodes and 2) A moving average of topology's nodal capacity over the last 50 episodes. In the normalized SAR<sup>4</sup> graphs, at the initial stage (episodes), the agent is more prone towards exploring the environment and later, from the acquired data, it starts exploiting the environment. For this reason, there is a steep increment in the acceptance rate for all cases. Furthermore, in comparison to other models, the nodal capacity graphs describe the efficiency of the proposed model in terms of network resource consumption. Overall, to analyze the benefits of the constructed model, we compared the performance with the single estimator DQL model and a heuristic model. Table 4 summarizes the performance of the models in terms of service acceptability for all scenarios and topologies studied. For the monolithic and decomposed models, the 'Mono' and 'Decomp' indicate the achieved mean of the normalized SAR respectively.

#### A. HEURISTIC MODEL

Instead of choosing a substrate node randomly, we have established a heuristic model as mentioned in Algorithm 3,

<sup>4</sup>The SAR is normalized by the EUB values.

where the selection of the substrate nodes is based on auxiliary variables. These auxiliary variables are updated at every step based on the successful placement (steps 8-9), emphasizing the importance of selecting that substrate node. Lower the auxiliary variable value indicates the high importance of choosing that substrate node. The placement of VLs/mVLs and the decomposition are performed the same as in the proposed approach.

#### B. TIME COMPLEXITY

The upsurge of the time complexity in our proposed model depends upon various factors. Starting with the number of episodes assumed per run and their size. As described in section IV, each episode defines several arriving SFCs for the deployment; these services' complexity is measured by the presence of the number of VNFs and VLs chained together. In addition, with the introduction of decomposition (mVNFs/mVLs), based on the fine granularity factor, the time complexity increases. To regulate this, we have proposed a 'Granularity Criteria' and adopted the 'Re-architecture' concept. Apart from the dependency on the service type, the model's complexity is also enhanced based on the topology's density.

Our approach is characterized as a 'Dynamic Optimization', with the primary objective of dynamically embedding the 'short-lived' services. Traditionally, ML method like supervised learning requires the pre-existing datasets for training, testing and validation purposes. However, for a realistic scenario like this, the anticipation of the arriving service type is not feasible. This is one of the reasons for adopting the RL technique, where no pre-existing datasets are required but instead create an own repository from all transitions and learn from it. Depending on the impact factors mentioned above, the learning period of the model varies. During the initial phase, the proposed model provides a sub-optimal solution to all the arriving services. Using these transitions, the model is trained simultaneously until it has reached a stable learning point. Once the model has learned, it starts delivering the sub-optimal solution using the learned model (primary DNN), i.e., prediction occurs. At the same time, the target DNN keeps itself up to date with the new

**Algorithm 1:** VNF-FG Placement Based on DDQL

```

1 Initialize Learning Rate  $\eta$ , Discounting Factor  $\gamma$ ,
  Exploration Rate  $\epsilon$ , Memory Size, mini-Batch size,
  Replace  $\tau$  and Latency
2 Initialize Replay Buffer  $Bu$ 
3 Initialize Primary NN  $Q(s, \mathbf{a})$  with weights  $\theta$ 
4 Initialize Target NN  $Q'(s, \mathbf{a})$  with weights  $\theta' \leftarrow \theta$ 
5 foreach episode  $i = 1 \dots epi$  do
6   Reset the Environment
7   Initialize substrate node resource  $R_H$ , substrate link
    $R_N$ 
8   foreach time-step  $t = 1 \dots T$  do
9     Observer the state  $s_t$ 
10    while all VNFs are given to the agent do
11      Using  $\epsilon$ -Greedy Action Selection Method  $a_t$ 
12      if  $random(0, 1) > \epsilon$  then
13        Exploitation
14         $a_t = \arg \max_a Q(s_t, a_t; \theta_t)$ 
15      else
16        Exploration
17         $a_t = random(A(s))$ 
18      end
19      if Action  $a_t$  successfully embeds VNF  $v_t$  then
20        Reward  $r_t$ 
21      else
22        if check if Decomposition is possible
          then
23          Initialize Decomposition Identifier
24          Initialize Granularity criteria
25          Observe decomposed  $v_t$ , choose
          actions  $A$ 
26          Store micro-segmented  $v_t$  transitions
          in  $Bu$ 
27          Initialize Re-architecture model
28        else
29          Penalty  $r_t$ 
30        end
31      end
32      Take action  $a_t$ , then observe reward/penalty
       $r_t$  and next state  $s_{t+1}$ 
33      Store transitions  $(s_t, a_t, r_t, s_{t+1})$  in  $Bu$ 
34    end
35    foreach update step do
36      Uniformly Sample the mini-batch size
      transitions
37      sample  $b_t = (s_t, a_t, r_t, s_{t+1}) \sim Bu$ 
38      Estimate the Target Value  $Q$  using
      Equation 7
39      Estimate the Loss function Equation 10
40      Update the Primary NN
41      Update the Target NN after every  $\tau$  steps
42       $\theta' \leftarrow \tau * \theta + (1 - \tau) * \theta'$ 
43    end
44    Execute Algorithm 2
45  end
46 end

```

**Algorithm 2:** Virtual Link Placement

```

1 if all VNFs/ mVNFs are deployed then
2    $x_0 \leftarrow$  source VNFs/ mVNFs,
3    $x_j \leftarrow$  Destination VNFs/ mVNFs
4   Estimate all path  $P$  between  $x_0$ , and  $x_j$ 
5   foreach substrate path  $p = 1 \dots P$  do
6     while  $b = j$  do
7       Estimate Latency  $(x_0, x_b)$ 
8       if  $Latency(x_0, x_b) \leq Latency$  then
9         Reward is given
10      else
11        Penalty is given
12      end
13    end
14  end
15 else
16   Penalty is given
17 end
18 Store transition  $(s_t, a_t, r_t, s_{t+1})$ 

```

**TABLE 2.** DDQL model parameters.

Parameters	
Hidden Layers	6
Neurons per Layer	300
Neural Network	2
Target NN update ( $\tau$ )	10000
Activation function	ReLU
Optimizer	Adam
Learning Rate	0.0001
Discounting Factor	0.99
Batch Size	64
Memory Size	50000
Epsilon Decay	0.0001
Loss Function	Mean Square Error (MSE)

**TABLE 3.** Topology-based nodal capacity.

Topology	12 – 4scenario	12 – 8scenario
$\zeta_{Netrail}$	336 CPU cores	672 CPU cores
$\zeta_{BtEurope}$	1152 CPU cores	2304 CPU cores

arrivals and periodically updates the primary DNN, which brings that stability to the learning graph. Thus, our algorithm is an online deployment process which learns and delivers the solution at a time.

**C. NETRAIL TOPOLOGY**

Figures 8 and 9 represent the normalized SAR moving average for the Netrail topology for the 12-4 nodal capacity scenario with the EUB as 22 SFCs, and 12-8 nodal capacity scenario with the EUB as 44 SFCs, respectively. These EUB values are normalized and are represented in the figures as normalized EUB. Considering the DDQL model's performances for the mentioned figures, for the 12-4 scenario, the microservice (DDQL+Decomp) paradigm successfully preserved  $\sim 48\%$  more SFCs than the monolithic (DDQL+Mono). Similarly, for the higher nodal capacity

TABLE 4. Mean of normalized SAR.

Topology	Scenarios (CPU cores)	DQL		Heuristic		DDQL	
		Mono	Decomp	Mono	Decomp	Mono	Decomp
Netrail	12-4	27.14	65.98	37.11	44.33	63.57	94.27
	12-8	27.51	60.47	33.06	39.68	67.27	95.40
BtEurope	12-4	8.98	17.16	19.93	25.46	34.99	74.83
	12-8	9.28	17.17	16.63	21.56	34.13	51.13

**Algorithm 3:** VNF-FG Placement Based on Heuristic

```

1 Initialize Latency
2 Initialize auxiliary variables
3 foreach episode  $i = 1 \dots epi$  do
4   Reset the Environment
5   Initialize substrate node resource  $R_H$ , substrate link  $R_N$ 
6   foreach time-step  $t = 1 \dots T$  do
7     Observer the state  $s_t$ 
8     Estimate the weights/auxiliary value for each substrate node
9     Arrange the Action space in ascending order
10    while all VNFs are given to the agent do
11      Using Weighted Action Selection Method  $a_t$ 
12       $a_t$  is the action with the lowest auxiliary value
13      if Action  $a_t$  successfully embeds VNF  $v_t$  then
14        Update the auxiliary variables
15      else
16        Check if Decomposition is possible
17        Initialize Decomposition Identifier
18        Initialize Granularity criteria
19        Observe decomposed  $v_t$ , choose actions  $A$ 
20        Initialize Re-architecture model
21    end
22  end
23  Execute Algorithm 2
24 end
25 end

```

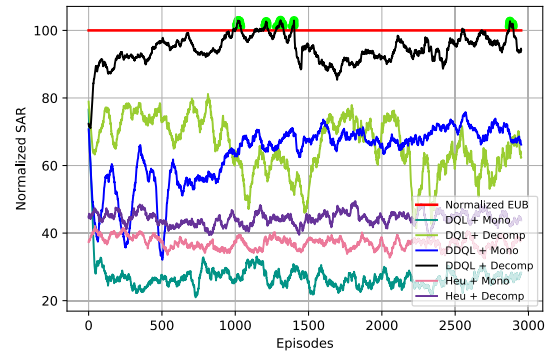


FIGURE 8. Netrail SAR: 12-4 scenario.

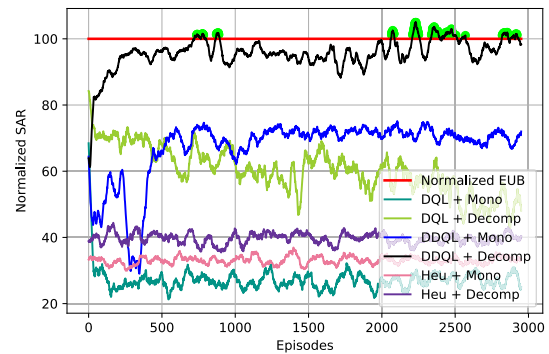


FIGURE 9. Netrail SAR: 12-8 scenario.

(12-8 CPU cores), the DDQL+Decomp model saved ~ 42% more SFCs than DDQL+Mono. This is because the decomposed SFCs can still find a placement when the network availability is low. The proposed approach efficiently identifies the high-resource demanding VNFs, decomposes them into appropriate  $m$ VNFs and successfully deploys them, causing a lower rejection rate. In case of the monolithic VNFs, these high-resource demanding VNFs failed to acquire a substrate node for placement during low network availability, resulting in a higher service rejection rate and delivering inadequate performance. It is evident from the results that the proposed model has significantly embedded more SFCs over monoliths. These figures also demonstrate the advantage of introducing the re-architecture module, which effectively

eliminated duplicate functionalities and saved enough link and nodal resources to deploy the SFCs more than predicted. This is visible from the figures as the normalized SAR surpassed the normalized EUB by ~ 1–2% higher for 12-4 scenarios and almost >2% more for 12-8 cases (which has been highlighted in the figures). Hence, with the combination of microservices and re-architecture, the resources in the Netrail topology have been effectively utilized.

Comparing the DDQL, DQL, and heuristic models' performance in Figures 8 and 9. Even though the heuristic decomposed (Heu+Decomp) model outperforms Heu+Momo and DQL+Mono, in general, both DQL and heuristic models show highly unstable learning. Because the heuristic model suffers from the greedy selection of highly favoured nodes, resulting in the quick depletion of their resources and causing highly underutilized nodes. Whereas the DQL model has only one estimator which attempts to learn both the behaviour and target policies to achieve an optimal solution. This prompts continuous updates to the weights causing a significant divergence in learning, which is observed in all the presented DQL results. The learning becomes more visible in the DDQL

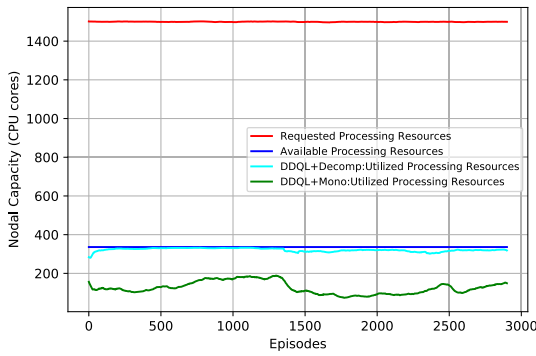


FIGURE 10. Netrail nodal capacity: 12-4 scenario.

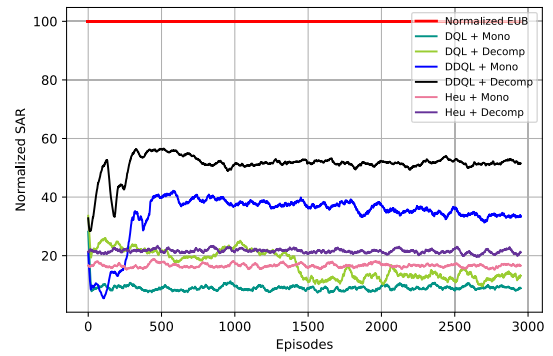


FIGURE 13. BtEurope SAR: 12-8 scenario.

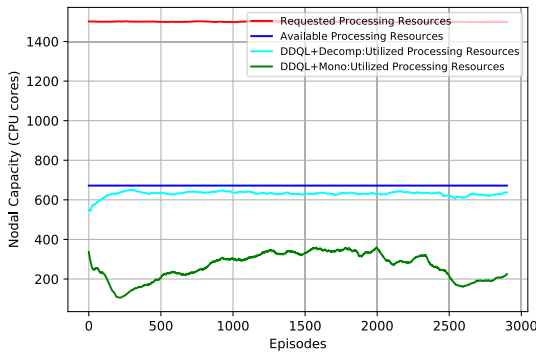


FIGURE 11. Netrail nodal capacity: 12-8 scenario.

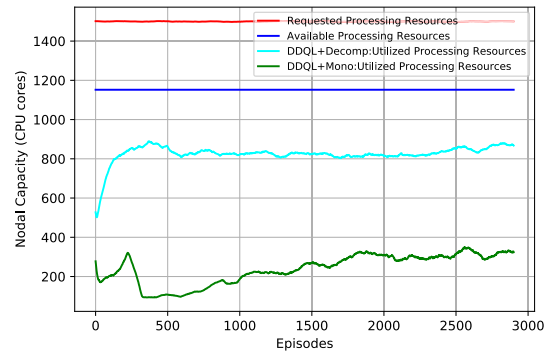


FIGURE 14. BtEurope nodal capacity: 12-4 scenario.

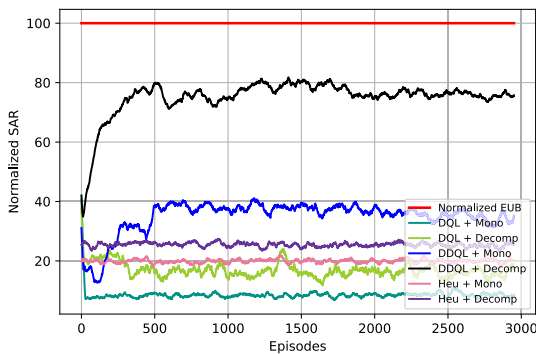


FIGURE 12. BtEurope SAR: 12-4 scenario.

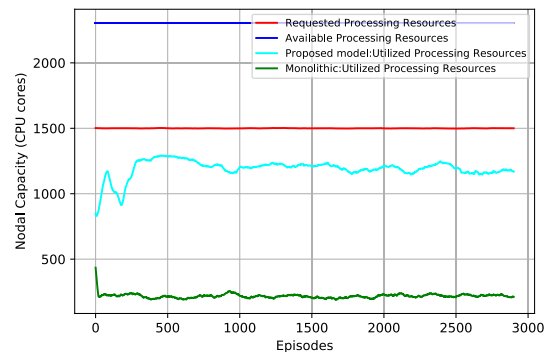


FIGURE 15. BtEurope nodal capacity: 12-8 scenario.

model with the double estimator (Primary + Target NNs), where the target network is updated at every  $\tau$  iteration instead of every other step encouraging much more stable learning.

The behaviour of nodal capacity on the topology is depicted in Figures 10 and 11. The red and blue trails represent the total processing resources (CPU cores) requested by the 100 SFCs over the episodes and the initial nodal resource on topology, respectively. For both 12-4 and 12-8 scenarios, our suggested model (DDQL+Decomp) more efficiently utilized the available nodal resources than the monolithic (DDQL+Mono) approach, resulting in increased in-service acceptance which is evident in Figures 8 and 9. Thus, this depicts the efficiency of the proposed approach.

#### D. BtEurope TOPOLOGY

Figures 12 and 13 illustrate the moving average of normalized SAR over the last 50 episodes for 12-4 and 12-8 nodal capacity scenarios for BtEurope topology, where the EUB is 76 SFCs and 153 SFCs, respectively.

The proposed model DDQL+Decomp successfully embedded about twice as many services as DDQL+Mono in the 12-4 scenario for the denser topology, as shown in Figure 12. Figure 13 indicates an almost 50% increment in the service acceptance by the proposed model for larger nodal capacity (12-8 CPU cores) than monolithic. Even with an excess of nodal resources, the designed model could only max up to 50% of normalized SAR, and this is due to the

TABLE 5. Notations.

Notations	Descriptions
$S$	State-space
$s_t$	State at time-step $t$
$A$	Action-space
$a_t$	Action at time-step $t$
$R_t$	Discounted Rewards at time-step $t$
$\pi_t$	Policy at time-step $t$
$Q_{s_t, a_t}^\pi$	Action value function
$\theta_t$	Parameters for $Q$ estimator
$\theta'_t$	Parameters for $Q'$ estimator
$G$	Directed graph for topology
$H$	Set of substrate nodes
$N$	Set of substrate links
$h_y$	Set of all nodal resource types in a substrate node $y$
$n_u$	Set of all link resource types in a substrate link $u$
$h_{y,w}$	Amount of resource type $w$ in a substrate node $y$
$n_{u,w}$	Amount of resource type $w$ in a substrate link $u$
$J_{node}$	Total nodal resource types
$J_{link}$	Total link resource types
$G'$	Directed graph for VNF-FG
$V$	Set of VNFs
$B$	Set of VLs
$\rho$	Potential/Candidate VNF for decomposition
$G'_\rho$	Directed graph for $m$ VNF-FG
$K_\rho$	Set of $m$ VNFs
$D_\rho$	Set of $m$ VLs
$V_\Psi$	Set of VNFs in SFC $\Psi$
$K_\Psi$	Set of $m$ VNFs in SFC $\Psi$
$R_\Psi$	Set of demanded resources by all VNFs in SFC $\Psi$
$L_\Psi$	Set of demanded resources by all $m$ VNFs in SFC $\Psi$
$r_{\Psi,x}$	Set of requested resources by VNF $x$ in SFC $\Psi$
$k_{\Psi,\rho}$	Set of finely-decomposed $m$ VNFs
$l_{\Psi,\rho}$	Set of requested resources by candidate VNF $\rho$ in SFC $\Psi$
$v_{\Psi,x}$	VNF $x$ in SFC $\Psi$
$r_{\Psi,x,w}$	Amount of requested resource type $w$ in VNF $x$ , SFC $\Psi$
$l_{\Psi,\rho,z,w}$	Amount of requested resource type $w$ in $m$ VNF $z$ , SFC $\Psi$
$A_\Psi^{vnf}$	Set of action-space for all monolithic VNFs in SFC $\Psi$
$A_\Psi^{mvnf}$	Set of action-space for all decomposed VNFs in SFC $\Psi$
$a_{\Psi,x}^{vnf}$	Set of action-space for monolithic VNF $x$ , SFC $\Psi$
$a_{\Psi,x}^{mvnf}$	Set of action-space for decomposed $m$ VNF, SFC $\Psi$
$R(\Psi)$	Reward function for SFC $\Psi$
$L_{reward}(\Psi_{vnf/mvnf})$	Local reward for VNF or $m$ VNF
$G_{reward}(\Psi_{vnf/mvnf})$	Global reward for VNF-FG or $m$ VNF-FG
$GR_{G'}^{pt}$	Global reward points for VNF-FG
$GR_{G''}^{pt}$	Global reward points for $m$ VNF-FG
$R_{vnf/mvnf}^{pt}$	Reward points for VNFs or $m$ VNFs
$P_{vnf/mvnf}^{pt}$	Penalty points for VNFs or $m$ VNFs
$R_{vl/mvl}^{pt}$	Reward points for VLs or $m$ VLs
$P_{vl/mvl}^{pt}$	Penalty points for VLs or $m$ VLs
$\Xi(\cdot)$	Agent's function
$\mathcal{R}(\cdot)$	Currently available resource for selected action
$E_\Psi^{vnf}$	Set of agent selected actions for all VNFs, SFC $\Psi$
$E_\Psi^{mvnf}$	Set of agent selected actions for all $m$ VNFs, SFC $\Psi$
$e_{\Psi,x}^{vnf}$	Obtained action for VNF $x$ , SFC $\Psi$
$e_{\Psi,\rho}^{mvnf}$	Decomposition status for VNF $\rho$ , SFC $\Psi$
$e_{\Psi,\rho,f}^{mvnf}$	Obtained action for $m$ VNF $f$ , VNF $\rho$ , SFC $\Psi$
$\zeta_{topology}$	Maximum nodal capacity of a node in a topology
$\varphi_{topology,t}$	Currently available nodal resource in a topology at time-step $t$

TABLE 6. Notations.

Notations	Descriptions
$S$	State-space
$s_t$	State at time-step $t$
$A$	Action-space
$a_t$	Action at time-step $t$
$R_t$	Discounted Rewards at time-step $t$
$\pi_t$	Policy at time-step $t$
$Q_{s_t, a_t}^\pi$	Action value function
$\theta_t$	Parameters for $Q$ estimator
$\theta'_t$	Parameters for $Q'$ estimator
$G$	Directed graph for topology
$H$	Set of substrate nodes
$N$	Set of substrate links
$h_y$	Set of all nodal resource types in a substrate node $y$
$n_u$	Set of all link resource types in a substrate link $u$
$h_{y,w}$	Amount of resource type $w$ in a substrate node $y$
$n_{u,w}$	Amount of resource type $w$ in a substrate link $u$
$J_{node}$	Total nodal resource types
$J_{link}$	Total link resource types
$G'$	Directed graph for VNF-FG
$V$	Set of VNFs
$B$	Set of VLs
$\rho$	Potential/Candidate VNF for decomposition
$G'_\rho$	Directed graph for $m$ VNF-FG
$K_\rho$	Set of $m$ VNFs
$D_\rho$	Set of $m$ VLs
$V_\Psi$	Set of VNFs in SFC $\Psi$
$K_\Psi$	Set of $m$ VNFs in SFC $\Psi$
$R_\Psi$	Set of demanded resources by all VNFs in SFC $\Psi$
$L_\Psi$	Set of demanded resources by all $m$ VNFs in SFC $\Psi$
$r_{\Psi,x}$	Set of requested resources by VNF $x$ in SFC $\Psi$
$k_{\Psi,\rho}$	Set of finely-decomposed $m$ VNFs
$l_{\Psi,\rho}$	Set of requested resources by candidate VNF $\rho$ in SFC $\Psi$
$v_{\Psi,x}$	VNF $x$ in SFC $\Psi$
$r_{\Psi,x,w}$	Amount of requested resource type $w$ in VNF $x$ , SFC $\Psi$
$l_{\Psi,\rho,z,w}$	Amount of requested resource type $w$ in $m$ VNF $z$ , SFC $\Psi$
$A_\Psi^{vnf}$	Set of action-space for all monolithic VNFs in SFC $\Psi$
$A_\Psi^{mvnf}$	Set of action-space for all decomposed VNFs in SFC $\Psi$
$a_{\Psi,x}^{vnf}$	Set of action-space for monolithic VNF $x$ , SFC $\Psi$
$a_{\Psi,x}^{mvnf}$	Set of action-space for decomposed $m$ VNF, SFC $\Psi$
$R(\Psi)$	Reward function for SFC $\Psi$
$L_{reward}(\Psi_{vnf/mvnf})$	Local reward for VNF or $m$ VNF
$G_{reward}(\Psi_{vnf/mvnf})$	Global reward for VNF-FG or $m$ VNF-FG
$GR_{G'}^{pt}$	Global reward points for VNF-FG
$GR_{G''}^{pt}$	Global reward points for $m$ VNF-FG
$R_{vnf/mvnf}^{pt}$	Reward points for VNFs or $m$ VNFs
$P_{vnf/mvnf}^{pt}$	Penalty points for VNFs or $m$ VNFs
$R_{vl/mvl}^{pt}$	Reward points for VLs or $m$ VLs
$P_{vl/mvl}^{pt}$	Penalty points for VLs or $m$ VLs
$\Xi(\cdot)$	Agent's function
$\mathcal{R}(\cdot)$	Currently available resource for selected action
$E_\Psi^{vnf}$	Set of agent selected actions for all VNFs, SFC $\Psi$
$E_\Psi^{mvnf}$	Set of agent selected actions for all $m$ VNFs, SFC $\Psi$
$e_{\Psi,x}^{vnf}$	Obtained action for VNF $x$ , SFC $\Psi$
$e_{\Psi,\rho}^{mvnf}$	Decomposition status for VNF $\rho$ , SFC $\Psi$
$e_{\Psi,\rho,f}^{mvnf}$	Obtained action for $m$ VNF $f$ , VNF $\rho$ , SFC $\Psi$
$\zeta_{topology}$	Maximum nodal capacity of a node in a topology
$\varphi_{topology,t}$	Currently available nodal resource in a topology at time-step $t$



exhaustion of link resources. Again, observing the DQL and heuristic models, the performances remain unstable, with substantial learning divergence. Even with the increase in topology complexity, the model learned to determine the decomposing VNFs and their granularity, as well as restructuring the SFCs, leading to a more profitable normalized SAR.

### E. NODAL CAPACITY

Figures 10 and 11, Figures 14 and 15 demonstrate the understanding of how nodal capacity is utilized by the various models depending on topology. The utilized processing resources by DDQL+Decomp nearly overlap with the total available processing resources in the topology, as shown in Figures 10 and 11, indicating that the DDQL+Decomp model almost employed the available resources to deploy arriving SFCs. However, in Figures 14 and 15, the proposed model DDQL+Decomp depicts some residual resources in the topology; with adequate learning, this would have improved the SARs. This can be improvised by using much denser NNs.

In summary, we used DDQL to address the microservices-based RA problem and evaluated its performance against DQL and a heuristic approach under different network density and nodal capacity situations. In comparison to monolithic SFC, the suggested approach demonstrated a significant increase in service acceptability for microservices-based SFC. The benefit of re-architecting is also fairly clear, especially in the Netrail topology.

### VII. CONCLUSION

In order to efficiently deploy and decompose the VNFs onto substrate networks, this work examines a novel RA technique based on microservices. Our analysis discovered that it could be a resource- and time-intensive to decompose every arriving VNF. The proposed algorithm finds the right balance of when and how much decomposition is allowed. This study proposes a criterion for determining the potential VNFs for decomposition and fine granularity technique and adopts the re-architecture concept for decomposed VNFs.

We investigated DDQL to solve the NFV-RA problem and compared the performance with the DQL and a heuristic model. The performance of DDQL for microservices and monoliths are thoroughly investigated under different network density and nodal capacity situations. By eliminating the over-estimation problem, the DDQL agent discovered the characteristics of the topologies under defining conditions and provided efficient solutions. The proposed model showed a dramatic improvement in the service acceptance for microservices-based SFCs over monolithic SFCs, where even 50% of more recovery by microservices is noticeable. The advantage of re-architecture is also quite evident, especially in the Netrail topology, where the model understands microservices' importance and prefers to opt for it. The model exceeds the EUB for a smaller topology; it tries to reach the upper bound for the denser topology but is limited due to the link resource exhaustion. We advocate using

significantly denser NNs for improved learning for denser topology. Consequently, our proposed model performs better under sparse resource availability.

With the increased topology complexity, the model requires a higher learning period to understand the environment, inducing higher computation time to solve the NFV-RA problem. The Deep Deterministic Policy Gradient (DDPG), a model-free, actor-critic technique that deals with larger action spaces, can be used to improve the results produced and will be a part of our future work. Moreover, implementing the microservices approach in real-time may raise many challenging questions, like security and increased operational complexity. In our future works, we will concentrate on understanding and providing solutions to such scenarios.

### APPENDIX

See Tables 5 and 6.

### REFERENCES

- [1] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, May 2020.
- [2] H. Ahmadi, A. Nag, Z. Khar, K. Sayrafian, and S. Rahardja, "Networked twins and twins of networks: An overview on the relationship between digital twins and 6G," *IEEE Commun. Standards Mag.*, vol. 5, no. 4, pp. 154–160, Dec. 2021.
- [3] *Network Functions Virtualisation Introductory White Paper*. Accessed: Jan. 29, 2020. [Online]. Available: [https://portal.etsi.org/NFV/NFV\\_White\\_Paper.pdf](https://portal.etsi.org/NFV/NFV_White_Paper.pdf)
- [4] M. Nekovee, S. Sharma, N. Uniyal, A. Nag, R. Nejabati, and D. Simeonidou, "Towards AI-enabled microservice architecture for network function virtualization," in *Proc. IEEE 8th Int. Conf. Commun. Netw. (ComNet)*, Oct. 2020, pp. 1–8.
- [5] S. Sharma, N. Uniyal, B. Tola, and Y. Jiang, "On monolithic and microservice deployment of network functions," in *Proc. IEEE Conf. Netw. Softwarization (NetSoft)*, Jun. 2019, pp. 387–395.
- [6] D. Moro, G. Verticale, and A. Capone, "A framework for network function decomposition and deployment," in *Proc. 16th Int. Conf. Design Reliable Commun. Netw. (DRCN)*, Mar. 2020, pp. 1–6.
- [7] 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, May 2019.
- [8] S. B. Chetty, H. Ahmadi, and A. Nag, "Virtual network function embedding under nodal outage using reinforcement learning," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS)*, Dec. 2020, pp. 1–6.
- [9] S. B. Chetty, H. Ahmadi, S. Sharma, and A. Nag, "Virtual network function embedding under nodal outage using deep Q-learning," *Future Internet*, vol. 13, no. 3, p. 82, Mar. 2021.
- [10] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.
- [11] M. Fowler. *Microservices Guide*. Accessed: Aug. 12, 2018. [Online]. Available: <https://martinfowler.com/microservices/>
- [12] K. Probst and J. Becker. *Engineering Trade-Offs and the Netflix API Re-Architecture*. Accessed: Nov. 11, 2018. [Online]. Available: <https://netflixtechblog.com/engineering-trade-offs-and-the-netflix-api-re-architecture-64f122b277dd>
- [13] *Implementing Microservices on AWS*, Amazon Web Services, Seattle, WA, USA, Nov. 2021.
- [14] M. Rost and S. Schmid, "On the hardness and inapproximability of virtual network embeddings," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 791–803, Apr. 2020.
- [15] N. D. Cicco, E. F. Mercan, O. Karandin, O. Ayoub, S. Troia, F. Musumeci, and M. Tornatore, "On deep reinforcement learning for static routing and wavelength assignment," *IEEE J. Sel. Topics Quantum Electron.*, vol. 28, no. 4, pp. 1–12, Jul. 2022.

- [16] P. T. A. Quang, Y. Hadjadj-Aoul, and A. Outtagarts, "A deep reinforcement learning approach for VNF forwarding graph embedding," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 4, pp. 1318–1331, Dec. 2019.
- [17] E. Datsika, A. Antonopoulos, N. Zorba, and C. Verikoukis, "Software defined network service chaining for OTT service providers in 5G networks," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 124–131, Nov. 2017.
- [18] K. D. Chinmaya and K. S. Prasan, "DYVINE: Fitness-based dynamic virtual network embedding in cloud computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1029–1045, May 2019.
- [19] S. Agarwal, F. Malandrino, C. F. Chiasserini, and S. De, "VNF placement and resource allocation for the support of vertical services in 5G networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 433–446, Jan. 2019.
- [20] M. A. T. Nejad, S. Parsaeefard, M. A. Maddah-Ali, T. Mahmoodi, and B. H. Khalaj, "vSPACE: VNF simultaneous placement, admission control and embedding," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 542–557, Mar. 2018.
- [21] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. D. Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *Proc. 1st IEEE Conf. Netw. Softwarization (NetSoft)*, Apr. 2015, pp. 1–9.
- [22] L. Ruiz, R. J. D. Barroso, I. D. Miguel, N. Merayo, J. C. Aguado, R. D. L. Rosa, P. Fernandez, R. M. Lorenzo, and E. J. Abril, "Genetic algorithm for holistic VNF-mapping and virtual topology design," *IEEE Access*, vol. 8, pp. 55893–55904, 2020.
- [23] M. A. Habibi, F. Z. Yousaf, and H. D. Schotten, "Mapping the VNFs and VLs of a RAN slice onto intelligent PoPs in beyond 5G mobile networks," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 670–704, 2022.
- [24] 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, Apr. 2020.
- [25] V. Sciancalepore, F. Z. Yousaf, and X. Costa-Perez, "Z-TORCH: An automated NFV orchestration and monitoring solution," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 4, pp. 1292–1306, Dec. 2018.
- [26] S. R. Chowdhury, H. Bian, T. Bai, and R. Boutaba, "A disaggregated packet processing architecture for network function virtualization," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1075–1088, Jun. 2020.
- [27] X. Fu, F. R. Yu, J. Wang, Q. Qi, and J. Liao, "Service function chain embedding for NFV-enabled IoT based on deep reinforcement learning," *IEEE Commun. Mag.*, vol. 57, no. 11, pp. 102–108, Nov. 2019.
- [28] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspar, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 98–106.
- [29] S. R. Chowdhury, R. Ahmed, N. Shahriar, A. Khan, R. Boutaba, J. Mitra, and L. Liu, "Revine: Reallocation of virtual network embedding to eliminate substrate bottlenecks," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Management*, May 2017, pp. 116–124.
- [30] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.
- [31] Y. Xie, Z. Liu, S. Wang, and Y. Wang, "Service function chaining resource allocation: A survey," 2016, *arXiv:1608.00095*.
- [32] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI*, vol. 30, no. 1, 2016, pp. 2094–2100.
- [33] 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, Mar. 2018.
- [34] *The Internet Topology Zoo*. Accessed: Apr. 15, 2020. [Online]. Available: <http://www.topology-zoo.org/dataset.html>
- [35] P. Erdos and A. Rényi, "On random graphs I," *Publ. Math. Debrecen*, vol. 6, nos. 290–297, p. 18, 1959.



**SWARNA B. CHETTY** received the B.E. degree in electronics and communication engineering from Sathyabama University, Chennai, India, in 2014, and the M.S. degree in mobile communication systems from the University of Surrey, U.K., in 2016. She is currently pursuing the Ph.D. degree with University College Dublin, Ireland. Prior to her Ph.D., she gained professional expertise as a software developer. Her research interests include network virtualization, resource allocations, microservices, machine learning (especially reinforcement and deep learning), 5G, and beyond communications.



**HAMED AHMADI** (Senior Member, IEEE) received the Ph.D. degree from the National University of Singapore in 2012. He was a Ph.D. Scholar at the Institute for Infocomm Research, A\*STAR. Since then, he worked at different academic and industrial positions in Ireland and U.K. He is currently a Senior Lecturer (Associate Professor) with the School of Physics, Engineering and Technology, University of York, U.K. He is also an Adjunct Academician at the School of Electrical and Electronic Engineering, University College Dublin, Ireland. He has published more than 70 peer-reviewed book chapters, journals, and conference papers. His current research interests include design, analysis and optimization of wireless communications networks, the application of machine learning in wireless networks, green networks, airborne networks, digital twins of networks, and the Internet of Things. He is a member of Editorial Board of IEEE SYSTEMS JOURNAL, *IEEE Communication Standards Magazine*, and *Wireless Networks* (Springer). He is a fellow of the U.K. Higher Education Academy, and Networks Working Group Chair of COST Action CA20120 (INTERACT).



**MASSIMO TORNATORE** (Senior Member, IEEE) is currently an Associate Professor with the Department of Electronics, Information, and Bioengineering, Politecnico di Milano. He has also held appointments as an Adjunct Professor at the University of California at Davis, Davis, CA, USA, and as a Visiting Professor at the University of Waterloo, Canada. He participated in several EU research and development projects (among others, FP7 COMBO, H2020 MetroHaul, and Cost Action RECODIS) as well as in several projects in the USA, Canada, and Italy. His research interests include performance evaluation, optimization and design of communication networks (with a textitasis on the application of optical networking technologies), network virtualization, network reliability, and machine learning application for network management. In these areas, he coauthored more than 400 peer-reviewed conferences and journal papers (with 19 best paper awards), two books, and one patent. He is a member of the Editorial Board of IEEE COMMUNICATION SURVEYS AND TUTORIALS, IEEE COMMUNICATION LETTERS, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, and *Optical Switching and Networking* (Elsevier). He is an Active Member of the Technical Program Committee of various networking conferences, such as INFOCOM, OFC, ICC, and GLOBECOM. He acted as a Technical Program Chair of ONDM 2016, DRCN 2017, and DRCN 2019 Conferences.



**AVISHEK NAG** (Senior Member, IEEE) received the B.E. (Hons.) degree from Jadavpur University, Kolkata, India, in 2005, the M.Tech. degree from the Indian Institute of Technology, Kharagpur, India, in 2007, and the Ph.D. degree from the University of California at Davis, Davis, CA, USA, in 2012. He worked as a Research Associate at the CONNECT Centre for Future Networks and Communication, Trinity College Dublin, before joining University College Dublin, Ireland. He is currently an Assistant Professor with the School of Electrical and Electronic Engineering, University College Dublin. His research interests include 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. He is the Outreach Lead for Ireland for the IEEE U.K. and Ireland Blockchain Group.