Online QoS/QoE-driven SFC Orchestration Leveraging a DRL Approach in SDN/NFV Enabled Networks

Mohamed Escheikh (mohamed.escheikh@enit.utm.tn)
National Engineering School of Tunis: Ecole Nationale d'Ingenieurs de Tunis
wiem Taktak Yakoub
École Nationale d'Ingénieurs de Tunis: Ecole Nationale d'Ingenieurs de Tunis

Research Article

Keywords: SDN/NFV, QoE/QoS, SFC orchestration, RL/DRL, Double DQN, Scale, Hyper-parameters

Posted Date: October 31st, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2842998/v1

Online QoS/QoE-driven SFC Orchestration Leveraging a DRL Approach in SDN/NFV Enabled Networks

Mohamed ESCHEIKH[1*] and Wiem TAKTAK[1†]

[1*]Syscom Laboratory, ENIT,, University of Tunis El Manar, BP 37, LE BELVEDERE, 1002, Tunis, Tunisia.

*Corresponding author(s). E-mail(s): mohamed.escheikh@enit.utm.tn;
Contributing authors: wiem.taktak@enit.utm.tn;
†These authors contributed equally to this work.

Abstract

The proliferation of the ever-increasing number of highly heterogeneous smart devices and the emerging of a wide range of diverse applications in 5G mobile network ecosystems impose to tackle new set of raising challenges related to agile and automated service orchestration and management. Fully leveraging key enablers technologies such as Software Defined Network (SDN), Network Function Virtualization (NFV) and Machine Learning (ML) capabilities in such environment is of paramount importance to address Service Function Chaining (SFC) orchestration issues according to user requirements and network constraints. To meet these challenges, we propose in this paper a Deep Reinforcement Learning (DRL) approach to investigate online Quality of Experience (QoE)/Quality of Service (QoS) aware SFC orchestration problem. The objective of this work is to fulfill intelligent, elastic and automated Virtual Network Functions (VNF)s/Container Network Function (CNF)s deployment optimizing end-to-end user experience while respecting QoS constraints. We implement the DRL approach through using a variant of Deep-Q-Network (DQN) algorithm referred to as Double DQN. We show how DRL agent behaves along the learning process for different PSN scales. We highlight also the impact of a set of hyper-parameters such as batch size and learning rate on solving the sequential decision problem related to SFC orchestration. The evaluation of the learning process is achieved based on the quality of learning with respect to the number of runs. In this regard, we use QoE metric to define a score quantifying the quality of learning.

Keywords: SDN/NFV, QoE/QoS, SFC orchestration, RL/DRL, Double DQN, Scale, Hyper-parameters

1 Introduction

The spectacular proliferation and the outstanding growth in mobile users and mobile devices, applications, and traffic worldwide, have imposed certainly new challenges but creates new opportunities to the service providers to shorten service innovation cycles, reduce the Total Cost of Ownership (TCO) and improve Average Revenue Per User (ARPU). Nowadays the advent of cloud computing, Software as a Service (SaaS) and key enablers technologies such as SDN, NFV and network slicing, are shifting towards the cloudification of the network resources in 5G and beyond where network connections are becoming faster, smarter and more automated. However, the growing density and complexity of traffic flow between service providers and users traversing one or more data centers are becoming more unpredictable and distributed and far more difficult to manage and control. Fully leveraging such technologies combined with emerging cognitive technologies such as ML are envisioned to provide cost-effective solutions achieving useful CAPital EXpense (CAPEX) and OPerational EXpense (OPEX) cost savings and boosting networks innovation. SDN is a promising networking paradigm that decouples control plane and data (forwarding) plane and moves the control logic to the SDN controllers [1] [2]. Thereby, each plane can independently scale to reduce TCO. The objective is to reduce management complexity and cost and to enhance network performance and management efficiency. This is achieved mainly through centralized network management and network programmability. This paradigm promotes agility and scalability in networks that customize the virtualized infrastructure of modern data centers to meet rapidly the evolving business lines requirements. Furthermore, SDN is leveraged for VNF orchestration by providing a centralized logical control thanks to centralized network controllers and by enabling service chains creation. On the other hand NFV represents a new architectural paradigm proposed by European Telecommunications Standards Institute (ETSI) that promotes developing new architectures, systems and applications and creating more agile and cost-effective network infrastructure [3] [4]. By leveraging virtualization technologies, NFV presents new opportunities to provide flexible and cost-efficient Network Functions (NF)s management and orchestration and accelerates successful Network Service (NS) deployment for network operators. Furthermore, NFV aims to virtualize the whole class of network node functions (e.g., FireWall (FW), Load Balancer (LB), Intrusion Detection System (IDS), Network Address Translator (NAT), Content Filters (CF) and others) into building blocks known as VNF that may be judiciously arranged and interconnected based on a given service logic to create composite communication services. NFV virtualizes NS traditionally running on proprietary hardware. These services are bundled into virtual machines or containers on Commercial Off-The-Shelf (COTS) hardware, allowing service operators to run their network on commodity, rather than proprietary servers and offering automated and on-demand service deployment. One of the main motivations of NFV is the agility to deploy new NFs chained together as

SFC on demand. In order to take advantage of this agility, a high level of automation is required to provision, configure and test the performance of VNFs. Instead of using middle-boxes as appliance and typical routing techniques to forward packets from source to destination, SFC proposes packet routing across a chain of NFs before reaching the destination (accounting for the type of service and policy in use). In this context the ETSI Industry Specification Group for NFV (ETSI ISG NFV) had defined an architectural framework referred to as Management and Orchestration (MANO) [3] [5]. MANO is a standard ensuring network resource coordination for cloud-based applications and managing life-cycle of both VNF and NS. It includes the functional blocks NFV Orchestrator (NFVO), VNF Manager (VNFM) and Virtualized Infrastructure Manager (VIM). Its key role is to facilitate the deployment and connection of new services. SFC paradigm in NFV environments describes a network capability that plays a key role in automating NSs deployment. It leverages traffic engineering techniques in forwarding and steering traffics between peers [6] [7]. Based on user demand, SFC request specifies the link order of a sequence of interconnected heterogeneous VNFs. The interconnection between successive VNFs is achieved through virtual links. These VNFs are traversed by user-generated traffic flows (i.e., SFC request) of a specific service/application toward the destination. The SFC may been seen as a processing pipeline composed by a set of VNFs to ensure on-demand end-to-end NSs delivery in virtual networking environment. Whenever traffic is steered across more than one VNF a service orchestrator is instantiated to establish a chain. Incoming flows are forwarded along a predetermined sequence chain of VNFs named SFC. SFC aims to automate the set up of virtual network connections in order to handle different types of traffic flows. The flow request with SFC is called as SFC Request (SR) (in the rest of this paper we will use interchangeably SFC Request and SR). Moreover, VNFs can easily place and run-on different COTS hardware like x86 servers or moved/migrated from one compute resource to another with respect to demand changes. This may be achieved without requiring to install or purchase new specialized hardware, allowing a faster service deployment and providing innovation and a great number of opportunities for the world of networked systems. This flexibility accelerates the delivery of services and applications and allows to respond quickly and to cope with the surge and changing demand in utilization patterns such as Over-The-Top (OTT) multimedia applications. Fundamentally, SFC routes packets through one or more service functions instead of conventional routing that routes packets using destination IP address. The SFC deployment in the SDN/NFV ecosystem, enables composing customized services and handles fine granular policies. The deployment of NS through SFC request embedding requires the mapping of VNF-Forwarding Graph (VNF-FG) on PSN [3]. Mapping between SFC request and PSN determines the allocation of VNFs and virtual links of the SFC request onto the PSN nodes and PSN links (fig.1). This implies to address the challenge of Placement and Chaining of VNFs (PC-VNF) according to the requests of VNF-FGs. Two kinds of sequential decisions making have to be considered while investigating the VNF-FG Embedding (VNF-FGE) problem: (a) where running the VNFs (on which PSN node will be performed the mapping), and (b) how interconnecting them in the PSN, taking into consideration the VNF ordering requirements.

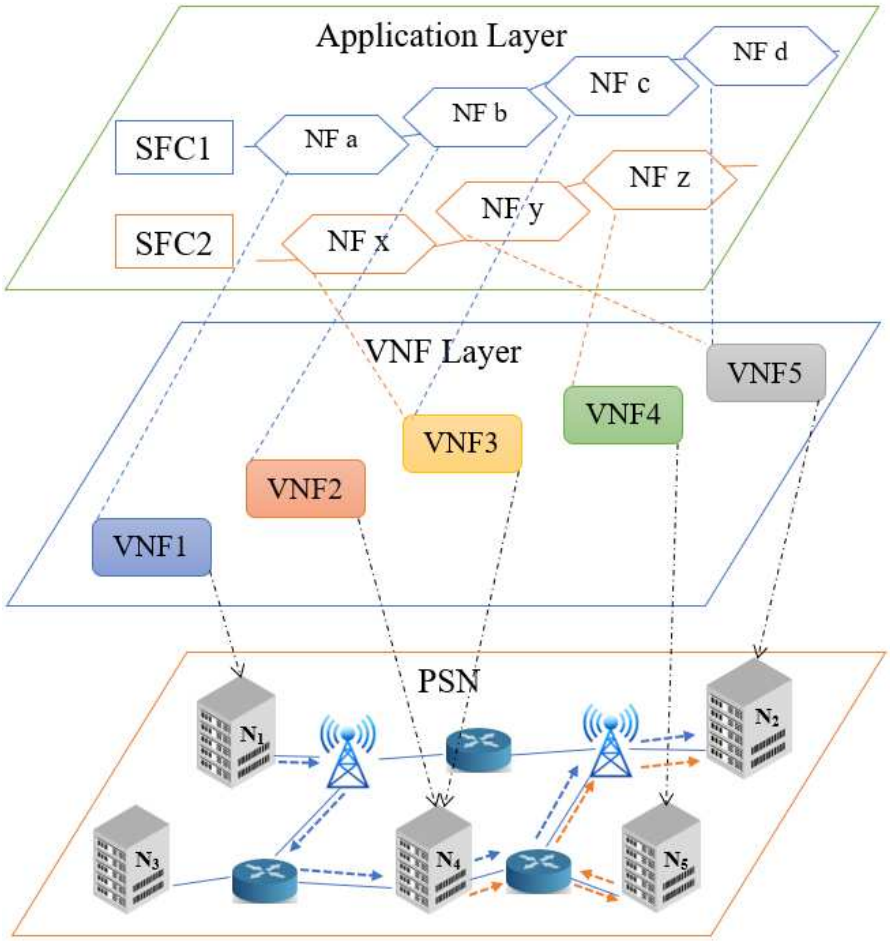


Fig. 1: SFC Orchestration in SDN/NFV Environment

SDN and NFV are often combined to bring a novel dimension for automating network provisioning and fostering future innovative services. They are integrated to create a flexible, resource-efficient, and programmable network architecture enabling to perform instantiating, managing, and orchestrating the life-cycle of service chains [8]. They increase the pace of expediting implementation over virtual networks infrastructure meeting specific needs of the vertical markets on the top on the same shared physical infrastructure. Thanks to SDN and NFV, service delivery automation is enabled. As a consequence, highly programmable networks with automated workflows that supports every single step of the service life-cycle are achievable. Synergy between SFC, SDN and NFV promises to bring agility and flexibility in terms of provisioning and deployment. It provides also centralized management to future mobile networks and accelerates and automates service delivery.

In this paper, we tackle the problem of optimizing resource allocation and management in a distributed NFV enabled networks dealing with dynamic SFC orchestration/deployment and involving VNF placement and virtual link embedding on PSN. We attempt, in this regard, to answer to the question: for each incoming SR, how the limited available resources in PSN should be managed conveniently to maximize QoE profit while meeting the QoS resource constraints. In order to answer this question, we consider an online strategy where each SFC request deployment will be built in a hop by hop manner using fine-grained control decisions and this deployment may be cancelled whenever end to end SR constraints are violated.

Our contributions are threefold:

- We formulate first the QoE/QoS-aware SFC orchestration issue as a mathematical optimization problem based on an objective function maximizing QoE profit while respecting some QoS constraints.
- We implement next the formulated problem using a DRL approach based on Double DQN.
- We investigate then the impact of the PSN scale and hyper-parameters (batch size and learning rate) on the learning process of the DRL agent controlling the SFC orchestration process through extensive simulations.

The remainder of this paper is structured as follows: Section 2 presents the background on NFV resource allocation and SFC orchestration. Section 3 details training, testing and implementation of RL and DRL approaches. We focus in section 4 on applying the DRL approach to solve the QoE/QoS aware SFC orchestration problem. We particularly detail first, in this same section, the proposed system model and problem formulation. The QoE/QoS aware SFC orchestration problem will be described next as a MDP with a specific reward model and solved using a DRL approach. We investigate then in section 5 performance evaluation and simulation results before concluding this paper in section 6.

2 NFV Resource Allocation and SFC Orchestration

2.1 NFV Resource Allocation

The main objective of NFV-Resource Allocation (NFV-RA) is to efficiently leverage the PSN resources to promote the NSs design, delivery and operation by bringing more flexibility and scalability. This may be fulfilled by maximizing resource utilization and/or QoE while minimizing operation overhead. Particularly, NFV-RA approach based on DRL may be proposed to efficiently solve SFC orchestration issue in NFV enabled networks. On the other hand, SFC orchestration issue may be transformed into a problem of SFC request deployment or VNF placement. It consists for each incoming SFC request, composed of ordered VNFs chained through virtual links, in placing all the corresponding VNFs on suitable PSN nodes (NFV-compliant servers) and mapping each of the virtual links (of the SFC request) on appropriate PSN links. The main objective of the SFC orchestration is the VNF placement through finding the optimal or near-optimal path, mapping the incoming users' SFC requests on the available PSN resources.

2.1.1 NFV-RA stages

From NFV-RA perspective, three distinguishable stages are considered [9]: VNFs Chain Composition (VNFs-CC), VNF-FGE and VNFs SCHeduling (VNFs-SCH).

- VNFs-CC copes with efficient VNFs concatenation and ordering.
- VNF-FGE seeks ensuring optimal VNF placement (virtual node embedding) and virtual link mapping.
- VNFs-SCH fulfills complicated scheduling decisions related to determining how to allocate NFV Infrastructure (NFVI) shared resources to several VNFs.

2.1.2 NFV-RA and SFC orchestration: scenarios, strategies and deployment

Along the SFC orchestration process, the automatic deployment of a composite service involving multiple heterogeneous NFs belonging to different service providers requires SFC orchestrator to enable end-to-end life-cycle management automation of NFs. The objective, in this case, is to build a processing pipeline involving a set of chained and ordered NFs to simplify and streamline complicated operation processes. Automation facilitates operation, manages harder complexity, and brings better accuracy. It also enables for composite and customized services to be achieved without adversely affecting the level of service expected defined in the Service-Level Agreements (SLAs). Furthermore, automation tackles dynamic workload conditions that ultimately improve network agility and scalability. In order to perform SFC orchestration, the operator requires finding for a given incoming SFC request, the right SFC instance complying with various PSN resource limitation and traffic fluctuations. On the other hand, achieving smart QoE/QoS-aware SFC orchestration requires suitable integration of both NFV and SDN with QoE/QoS metrics. It is highly recommended also to fully leverage ML techniques and particularly DRL making use of autonomous agent to meet dynamic NFV orchestration needs.

Actually, a SFC request might involve distributed execution of a set of VNFs interconnected through virtual links. Thus, the software service providers are facing a challenging resource allocation problem regarding VNFs deployment and virtual links mapping in the PSN (NFVI environment). In our case study, an incoming SFC request (characterized by an end-to-end bandwidth and end-to-end delay) looks for a suitable SFC instance (SFC response) that maximizes the user perceived quality (QoE) while accounting for the QoS constraints. Such objective is achieved, in this work, by respecting a set of constraints and limitations related to, the maximum number of VNF instances that may be deployed simultaneously per VNF license, the PSN nodes processing capacity and PSN links bandwidth.

SFC Orchestration Scenarios: In NFV-RA, the SFC orchestration problem may be solved according to two scenarios namely offline and online. In the offline scenario, one-time decisions are taken over aggregation of input traffics (the full SFC requests that are known in advance) to the PSN. Whereas, in the online scenario, the SFC orchestration strategy is adapted according to dynamic network load and the related problem is solved using migration algorithms on real time basis. In such scenario, a PSN under various resource availability constraints is considered with a set

of already deployed SFC requests and the new incoming SFC requests are processed on sequential basis (in a one-by-one modality) [10]. In online scenario, adopted in this paper, the challenge for each SFC request is to find a suitable SFC instance meeting its requirements.

SFC Orchestration Strategies: Along the SFC orchestration process two main strategies may be considered namely monolithic orchestration and incremental orchestration. In monolithic orchestration strategy, we verify for the current step whether there is an overall (a complete) SFC instance in the PSN that verifies or not SFC request requirements (maximizing QoE) and constraints (meeting QoS constraints). If this is not the case, we attempt to find another complete SFC instance in the next step. Conversely, in incremental orchestration strategy, each VNF of SFC request is selected in a hop-by-hop fashion. Therefore, the SFC length gradually increases by incrementally embedding VNFs and mapping virtual links composing the SFC request on the corresponding PSN nodes and PSN links of the NFVI respectively. The advantage of such strategy, adopted in this paper, is that it can be easily mapped to a multi-step RL problem and can perform quite a sophisticated policy maximizing QoE requirements while respecting QoS constraints. This may be achieved by formulating a suitable reward function.

SFC Deployment: From RA perspective and in order to deploy each incoming SR on a PSN, we need to find among all achievable SFC instances on PSN a suitable SFC instance maximizing QoE will meeting QoS requirements. Recall that, SR is a sequence of ordered VNFs chained through virtual links whereas the SFC instance corresponds to a set of available resources in PSN nodes and PSN links to be allocated in order to handle a given SR. The SR deployment process requires embedding VNFs (on PSN nodes) and mapping virtual links (on PSN links). As a result, the successful SR deployment requires the effective fulfillment of successive embedding and mapping actions. This is accomplished through following an incremental strategy aiming to ensure assigning step by step each SR component (VNF/virtual link) to the corresponding PSN component (PSN node/PSN link) of a suitable SFC instance.

2.2 SFC Orchestration

2.2.1 SFC Orchestration Approaches

SFC orchestration consists in automating SFC deployment and determining "a feasible path on the physical network where the vertexes and edges of the path can satisfy both the computing resource requests of VNFs and the bandwidth demands of virtual links" [11]. Mirjalily et al. [12], identifies four stages for SFC orchestration in PSN including description, composition, placement, and scheduling.

The existing approaches for SFC orchestration and adaptive SFC placement focusing on RA can be categorized into three types each of them has its own advantages and or limitations in terms of scalability, agility, flexibility, cost-effectiveness and complexity: According to [13], these types cover: (1) Mathematical optimization-based approach formulated through multi-objective Integer Linear Programming (ILP) models looking for global optimal solution. (2) Heuristic-based models, seeking a suitable trade-off between performance and complexity. (3) Markov decision process

(MDP)-based, RL and DRL models, accounting for the current and the future system performance.

Mathematical Optimization-Based Approaches: Mathematical optimization-based approaches may be achieved by means of several classes of linear programming algorithms, such as ILP [14], Mixed ILP (MILP) [15], and Integer Non-Linear Programming (INLP) [16], [17]. The choice of a given class of such algorithms is tributary of the optimization problem features. Several works in literature formulated the NFV-ORchestration (NFV-OR) problem as ILP model and uses heuristics to find a trade-off between optimality and efficiency [3]. F. Bari et al. [18] formulated OPEX problem as an ILP model, in order to minimize the overall costs and enhancing the network resources utilization. To fulfill some specific optimization objectives, aiming to minimize the number of busy servers and to maximize the acceptance rate of the incoming requests, J. G. Herrera et al. [3] abstracted the NFV-OR problem as an ILP model to solve the global optimal orchestration scheme. G. Lee et al. [19] tackles NFV-OR problem as an ILP model by considering resource limitation constraints. Nevertheless, ILP models for solving NFV-OR problems are proven to be NP-Hard (i.e., there is no known algorithm enabling its resolution in polynomial time) since the ILP solution space is not considered as a convex set, and convex optimization techniques are impractical in such context. Hence ILP models are generally unpractical to deal with large scale networks, since they are hard to scale well especially when the problem size increases and therefore the resolution could only work efficiently for very small instances. D. Li et al. [20] proved that the ILP models take mathematically four times as long as the heuristics with the same optimization toolboxes.

Heuristic and Meta-Heuristic based Approaches In order to mitigate the ILP problem complexity, several researches have resorted to heuristics to find near-optimal solution for the NP-hard problem, seeking for a suitable trade-off between complexity and performance. S. Sahhaf et al. [21] leveraged the heuristic method based on the ILP model, which used backtracking mechanism to obtain the most convenient deployment scheme. However, the corresponding time complexity is shown to be o(n!). D. Li et al. [22] jointly combined ILP model with the dynamic programming-based heuristic method to study the NFV-OR problem. Nonetheless, it prioritized to sharing the same node between VNFs without accounting for resource link limitations. R. Mijumbi et al. [23] formulates the SFC orchestration problem by considering three greedy-based algorithms and a tabu search-based heuristic. However, the greedy-based mechanisms can easily fall into a local optimum. Authors in [24] formulate the virtual Deep Packet Inspection (vDPI) placement problem as a cost minimization problem based on a centrality-based greedy algorithm. However, they only assumed one type of VNFs. In addition, works provided in [25] [26] adopted heuristic methods founded on ILP models to solve multiple objectives problem, in order to minimize the overall costs. Furthermore, they focus on real time system performance instead of considering the steady state system performance. Authors in [27] used a heuristic approach based on Segment Routing IPv6 (SRv6) to propose an effective SFC dynamic orchestration algorithm in a multi-domain NFV enabled networks. The objective is to enhance the SFC deployment in terms of performance metrics such as end-to-end delay, bandwidth consumption, and load balancing. Paper in [28] studies the digital coding scheme of

the heuristic SFC deployment in an NFV environment to enhance time efficiency without decreasing performance. Although the effectiveness of heuristic approaches, they may suffer from convergence problems and therefore slowness. Furthermore, they usually fall into the local optimal dilemma, and they are highly conditioned by the prior knowledge.

MDP-RL-DRL-Based Approaches In order to overcome the limitations of the above approaches, MDP-RL-DRL-based approaches are often proposed. From MDP-RL-DRL perspective SFC request deployment requires two kinds of decisions making: (a) where to run the VNFs (on which PSN node having sufficient available resources the VNF would be hosted and run), and (b) how to interconnect them in the PSN, accounting for VNF ordering requirements.

MDP-based models optimize NFV-OR problem by considering both current and future system performance. In work [29], the SFC problem is abstracted as a MDP and resource allocation is achieved according to a preemptive strategy. In addition, authors propose leveraging Bayesian learning algorithm to forecast the future resource reliability and suggest an MDP-based algorithm by adopting the asynchronous partitioning concept. S. C. Lin et al. [30] proposed RL-based adaptive routing method, for modeling QoS metrics of delay, loss, and throughput as the reward function. Then Softmax-based policy was used to select the next hop forwarding device. However, only packet forwarding strategy was developed and VNF placement was not considered. RL-based methods divided NFV-OR into two sub-problems (i.e., network-level and flow-level) depending on the generality of the learning target, aiming at achieving the automatic network configuration [31] [32] [33]. Notice that, RL maintains a Q-table to store policies, which cannot handle the large infinite state space. J. Pei et al. [34] proposed DDQN-VNFPA to obtain the optimal VNF placement solution from a prohibitively large solution space. But it only considers network-level other than flow-level, and the framework does not take QoS awareness into account in depth. Chen et al. [35] proposed RL algorithm to orchestrate SFCs, considering QoS and QoE synthetically. However, they only consider network-level optimization and no flow-level optimization strategy is devised. In addition, the proposed framework intentionally ignores some constraints, such as node resource in QoS metrics, which is a crucial factor in the real network environment.

2.2.2 QoE/QoS Aware SFC Orchestration

In this paper, we use a DRL approach to tackle QoE/QoS aware SFC Orchestration problem.

QoE Evaluation Based QoS Metrics: QoE is an ecosystem that involves together users, network, and network/service providers to measure the process as well as the outcomes of communication (e.g., user effectiveness, efficiency, satisfaction and enjoyment). It has drawn much attention over the past few years and has become a prominent issue for delivering services and applications. In this direction a huge research effort has been carried out for studying various aspects to apprehending, quantifying, and modelling QoE for a wide range of media services. The objective is to fulfill service assurance through delivering reliable and value-added services satisfying

the requirement of user and meeting user experience requirements while ensuring operational efficiency. High quality of experience of network services is nowadays becoming an essential requirement for NFV and a pivotal differentiator for gauging the effectiveness of telecom operators and service providers. Indeed, QoE degradation may cause critical quality assurance problems and may lead to the most sensitive network impairment. A good interplay is usually needed for real-time services between network and application dynamics so as to deliver an acceptable user experience [36]. The traditional monitoring and optimization of QoS parameters in the network lacks of knowledge of user's QoE and, as a result, of efficiency in improving user experience.

Objective vs Subjective: The QoE is related to both objective and subjective evaluation [35]. Objective QoE evaluation depicts the influence of the network and application performance on the user. Whereas subjective QoE evaluation quantifies the individual user experience obtained, when interacting with technology and business entities in a particular context to provide satisfaction to the end-user. This context may concern emotional state, feeling, preference .... Notice that, it is hard to apply subjective QoE evaluation in large scale networks. As a result, we opt in this paper for objective QoE evaluation which is derived and automated from measurable QoS metrics without involving end-user. This is achieved by characterizing the relationship between network-level QoS parameters and application QoE indicators. Such characterization is leveraged to formulate a DRL approach modeling a QoE/QoS SFC orchestration problem.

SFC Orchestration Maximizing QoE and Meeting QoS In this paper, we particularly focus for SFC orchestration on the online scenario and the incremental strategy and the SFC orchestration issue (including VNF placement and virtual link mapping) is formulated as DRL model maximizing QoE while meeting QoS constraints. This is investigated through a specific implementation based on Double DQN.

3 RL and DRL: Training, Testing and Implementation

3.1 Reinforcement Learning (RL)

In this section, we provide detailed description of the RL technique and how it is used to train an agent to perform a given task. We describe particularly the RL agent learning process and how agent interacts at each learning step with network environments to solve a RL problem and to achieve a given objective. RL is one of the hottest research topics in the field of modern AI expected to produce and boost developing a wide range of innovative solutions based on decision making science. It represents a class of ML algorithms that refers to sequential decision making under uncertainty (uncertain conditions of the real world) with the objective of reward optimization. The learning aims to solve a specific category of problems and is achieved through trial-and-error using rewards and punishments as signals for positive and negative behavior as feedback from agent actions and experiences. During the learning process, the agent strives to maximize some long-term reward (total or expected cumulative

reward over a trajectory including immediate and delayed rewards) in an interactive complex and uncertain environment. In this regard, the agent leverages the rewards from past experience and the environment's feedback to build a decision policy that would be incrementally and continuously enhanced at run-time and that can adapt according to dynamic environment changes [37].

In an active RL the agent executes a sequence of runs to interact with an environment. The objective is to try to learn an optimal policy that maximizes a long-term reward perceived from the environment by associating actions and states through multiple rounds of trial and error, based on the environment's feedback. In other words, the agent learns by trial-and-error how to select actions that maximize its expected discounted future rewards. Interaction between an agent and an unknown environment from the initial to final states is referred as an episode. An episodic task, unlike continuous task, lasts a finite amount of time and have a terminal state.

In RL, exploration and exploitation both share the same objective since they endeavor to maximizes rewards. However, what makes the dilemma between them is the limited knowledge or partial observability. Balancing exploration with exploitation and finding the perfect equilibrium between them is a serious challenge. The RL problem is formulated by the following terms:

- Environment: Describes physical or external world in which the agent operates and with which it interacts in order to collect information about the environment.
- State: Presents current situation of the agent. It provides a detailed and complete description of the environment's state.
- Reward: Quantifies feedback from the environment. It represents a signal that the agent leverages to measure its success and to quantify how good or bad the current world state is.
- Policy: Defines method to map agent's state to actions. It tries to maximize a reward and represents a rule leveraged by an agent to decide what actions to select.

We distinguish two kinds of RL methods (namely on-policy and off-policy). On-policy methods (i.e., Policy Iteration, Value Iteration, Sarsa) try to evaluate or enhance the policy used for decision making, whereas, off-policy methods (i.e., Q learning, Expected Sarsa) evaluate or enhance a policy for action selection.

3.2 Training and Testing in Online and Episodic RL Process

In RL, training and testing phases are typically referred to as the learning and evaluation phases, respectively. The RL agent is trained through interacting with its environment via a large number of trial and error learning process balancing between exploitation and exploration. This is achieved for episodic tasks by performing some actions, starting from the initial state and continuing along a series of steps until reaching either its goal (the desired state (final state)) and then it restarts, or the maximum learning duration expires. This duration is defined in a fixed number of episodes. In other words, the learning process may last less than on episode (if the goal state is reached) and at most a given number of episodes corresponding to the maximum learning duration (regardless of whether the goal state is reached or not). Notice also that whenever the agent reaches the final state before the maximum learning duration

expires, the problem is considered solved, otherwise the problem is never solved. It is worth mentioning that only trivial and simple enough RL problems have a solution. However most of the time real world RL problems are complex and may very likely never been solved even with an excessively long learning phase. This may be explained by the fact that the problem complexity is greater than that of the RL algorithm and in such case this latter can never hope full understanding of the problem leading to an optimal policy at the end of the learning process.

Since we are concerned in this paper with online RL testing (evaluation) can be done periodically during training phase (whereas in offline RL testing is achieved in separate phase following the training phase). The objective of the training phase is to reach final state corresponding to a satisfactory level of performance. This performance level can be quantified by a score measuring the learning quality of the agent. Once the current episode is finished a new episode is started by resetting the environment to its initial state. Such process is repeated iteratively to enable the agent to learn in a first step from its experience in every episode and to use in a second step that knowledge to enhance its performance in subsequent episodes.

Before dealing with the experimental results let's recall some preliminary definitions related to step, run, episode and trail used in RL. A step corresponds to a cycle of state-action-reward. Each incoming SR involves 5 steps or actions. The deployment of 100 SRs is fulfilled during one run. An episode is defined as a set of runs and it is stopped whenever a maximum number of runs expired or a training goal (final state) is achieved by the DRL agent. To show how the agent evolves during the learning phase during one episode (a set of runs) we plot the graph scores vs runs by considering the discounted cumulative expected rewards with respect to run. At larger scale, We train the agent during a set of episodes (referred to as a trail). To show how an agent behaves along a trail we plot the number of runs per episode with respect to order of each episode.

3.3 RL implementation via Q-learning: Scalability Issues

In this subsection, we detail Q-learning as example of off-policy RL and its scalability issues.

Q-learning: Q-learning is an off-policy RL algorithm [37] able to learn from data collected by any behavioral policy where the agent estimates the reward for future actions and appends a value to the new state without actually following any greedy policy. In Q-learning, the Q-learning function learns from actions that are outside the current policy. It seeks to find the next best action (decision) to take given the current state without requiring any environment model (model free). More specifically, Q-learning looks for learning a policy that maximizes the total reward. It uses a Q-table that stores Q-values (Q-Table elements) after an episode. This allows the agent to view all possible actions for a given state and to evaluate the performance of any action in a given a state and to choose subsequently the best one. Q-Value quantifies a measure of the expected discounted cumulative reward assuming the agent is in a given state s and performs action a, and then continues playing until reaching the end

of the episode according to some policy.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)) \quad (1)$$

Scalability Issues in Q-learning: In complex environments many network states may be hidden and action space may be huge and as a consequence table-based RL agents will be unable to provide useful solutions. Indeed, tabular methods complexity scales linearly with the number of states and becomes ineffective whenever the state and actions spaces are large. RL algorithms require an exact representation of value functions and policies. Such representation is infeasible in large real-world problems ,and therefore value function approximation methods are used. The aim these methods is to scarify some representation accuracy for the sake of scalability. In order to solve the high-dimensional mapping problem DRL combines RL with DL and uses a Neural Networks (NN)s as a nonlinear type of function approximator to approximate optimal policy and value functions. The goal of the DRL agent is to find the parameterized policy with the maximum expected rewards. Several DRL implementations are proposed in literature including DQN and its variants (Double DQN, Dueling DQN, ...).

3.4 DRL Implementation via DQN

DQN [38] combines Q-learning with Deep NN (DNN) function approximation and experience replay [39]. Q-learning is used to learn the best action to take in the given state whereas DNN brings a compact representation of both high-dimensional observations and the Q-function. It provides better expressiveness to approximate Q-function of Q-learning and allows Q-learning to be applied to more complex and high-dimensional problems intractable with a table-based approach. When compared to standard ML, DNN [40] enables more efficiency as the volume of data increases. It leverages multiple layers to represent the abstractions of data in order to build computational models. Since DNN uses a large number of parameters, it spends a long time to train a model. Conversely, its testing phase takes a short amount of time when compared to standard ML algorithms [41]. Instead of storing an action-value table, DQN leverages a DNN as a function approximator and uses a parameterized Q-function $Q(s, a; \theta) \approx Q(s, a)$ where, θ represents the DNN parameters. By training DNN with gradient descent instead of the Q-Learning iterative update process, DQN aims to minimize a loss function at iteration i:

$$L_i(\theta_i) = \mathbb{E}_{s,a,r,s'}[(y_i^{DQN} - Q(s, a; \theta_i))^2] \quad (2)$$

Although using a non-linear DNN brings significant advantage, it often yields training instability if it is used inappropriately. The big problem is to compute NNs that are both stable and accurate and the challenge is to find the right trade-off between stability (i.e., convergence) and accuracy (i.e., performance). The DQN tackles the fundamental instability problem of using function approximation in RL by leveraging two innovative techniques: experience replay and target networks. The experience

replay aims to break harmful correlations between different training samples however the target Q-network attempts to provide better stability to the training target, the target Q-value. Given the state s', reward r, discount factor γ, DQN computes the target Q-value y_i^{DQN} as follows:

$$y_i^{DQN} = r + \gamma max_{a'} Q(s', a'; \theta^-) \tag{3}$$

where θ^- represents the parameters (weights) of a fixed and separate target network. Standard Q-learning is usually used to learn the parameters of the network $Q(s, a; \theta)$ online. Nevertheless, this estimator provides poor performance in practice. In order to solve this problem and to prevent over-fitting a key breakthrough advancement provided in [38] consists in freezing the target network parameters $Q(s', a'; \theta^-)$ for a fixed number of iterations while updating the online network $Q(s, a; \theta_i)$ by gradient descent. Such freezing technique enables in DRL context to improve the stability of the training and to sample efficiently the learning process. The specific gradient update is given as follows:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a,r,s'}[(y_i^{DQN} - Q(s, a; \theta_i))\nabla_{\theta_i} Q(s, a; \theta_i)] \tag{4}$$

3.5 DRL Implementation via Double DQN

Several DQN variants such as Double DQN and Dueling DQN are proposed in literature. We focus in this paper on Double DQN considered as a model-based RL algorithm. The basic idea behind Double DQN [42] is to improve DQN by addressing the target Q over-estimation problem [43] associated with Q-learning. Double DQN uses two networks to avoid over optimistic Q-values. Such improvement is achieved through decoupling the action selection and action evaluation (Q-value estimation) steps while computing the target Q-value (Eq.(5)).

$$y_i^{DoubleDQN} = r + \gamma Q(s', argmax_{a'} Q(s', a'; \theta_i); \theta^-) \tag{5}$$

Action selection is fulfilled using the current Q-network with weights θ while action evaluation is accomplished using DQN's target Q-network, with weights θ^-.

4 RL Driven QoE/QoS Aware SFC Orchestration

In this section, we introduce first the system model and problem formulation. We present next the MDP modeling the QoE/QoS aware SFC orchestration problem. We propose then the reward model and the main building blocks involved in its conception. We detail last how to solve the QoE/QoS aware SFC orchestration problem based on DRL approach implemented via DQN variants.

4.1 System Model and Problem Formulation

Our research motivation concerns determining how to deploy one by one incoming SFC requests on PSN. In this section, we introduce first the system model representing formal statement of both PSN and SFC request. We provide next a detailed

formulation of the QoE/QoS aware SFC orchestration problem and related constraints explanation.

4.1.1 System Model

In this subsection, we start the system model representing formal statement of both PSN and SR.

PSN Model: The PSN, namely NFV Infrastructure (NFVI) by ETSI [44], is formalized as a model represented by an undirected weighted graph (i.e. a graph where the edges have no orientation and weights are assigned to nodes and edges). This graph is denoted by $G = (V(G), E(G))$ where $E(G)$ is the set of its PSN links and $V(G)$ is the set of the PSN nodes. Furthermore, We assume a PSN topology with fully interconnected PSN nodes (hosting VNFs) through PSN links. We assume also that PSN resources are limited and these limitations concern both PSN nodes and links and are taken into account whenever an incoming SFC request solicits a SFC instance (in the PSN) meeting its requirements in terms of capacity, delay and bandwidth. In this regard the DRL agent, based on an algorithmic approach makes decision about accepting or rejecting this request according to SFC requirements and PSN resource constraints. The SFC request is considered accepted if and only if all the responding VNFs and virtual links are successfully embedded. Also, we characterize each PSN node, $w \in V(G)$, by the corresponding available (idle) computing capacity (in terms of CPU cores) denoted by $C(w)$. Likewise each PSN link, $e \in E(G)$, is commonly described by its available bandwidth $Bw(e)$ and the corresponding transmission delay $D(e)$ for processing traffic flow traversing it. Without loss of generality further metrics such as storage and memory may be taken into consideration to enlarge the PSN node description. From RA perspective, a PSN may be seen as a set of SFC instances. Each of them may be dynamically assigned to a given SR.

SR Model: We consider, in this paper, a SR characterized by both VNFs and virtual links QoS requirements. VNF QoS requirement covers computing capacity C (in CPU cores) whereas virtual link QoS requirements encompasses bandwidth Bw (in $Mbps$) and delay D (in ms) (as mentioned in table 1). The SR is formally modeled by a VNF-FG graph namely $FG = (V(FG), E(FG))$ where $V(FG)$ is the set of VNFs (notice that the ingress and egress nodes are considered of VNF type) and $E(FG)$ is the set of virtual links. It is specified by end-to-end requirements in terms of delay D (the maximum tolerable latency between any pair of endpoints) and bandwidth Bw (the minimum bandwidth between any pair of endpoints)(see table 1). Each VNF, $n \in V(FG)$, is described by the required processing CPU core capacity $C(n)$ whereas each virtual link $l \in E(FG)$ is characterized by its required bandwidth $Bw(l)$ and transmission delay $D(l)$.

4.1.2 Problem formulation

We explicitly formulate, in this subsection, the SFC orchestration problem with specific objective function and constraints as an ILP model. The objective function aims to maximize end-to-end QoE of the incoming SFC request while meeting QoS constraints. These constraints concerns respectively the maximum number of VNF

instance in VNF License, VNF Placement, PSN Node Capacity, PSN Link Capacity, Delay. Detailed explanation of the above problem formulation is provided in the following paragraph. The problem formulation and the main related notations and parameters are summarized in Table 1.

Constraints Explanation: For every incoming SR, we are looking for investigating the online VNF deployment/ embedding and chaining problem on PSN. The question to be answered in this context is how to steer a SFC request to suitable path in PSN. This comes down to find the most appropriate (the best) SFC, in the PSN, meeting the SFC request requirements maximizing QoE and meeting QoS requirements and PSN constraints related to resource limitations. The definition of SFC orchestration problem requires in a first step the detailed specification of the different constraints related to PSN Nodes (resp. links). It requires, in a second step, its mathematical formulation as a constrained optimization problem. Notice that in this paper, we are concerned with maximizing QoE while respecting QoS constraints. The objective of the SFC orchestration is to embed all the VNF (resp. virtual link) of the SFC request on the corresponding PSN node (resp. PSN link) of the PSN. Where Eq.(7) specifies that the maximum number of VNF instances of type i, vnf_i, deployed on the PSN, should not exceed the number L_{vnf_i} of VNF instances per license of a given VNF type owned by the operator. Placement constraints related to deploying the entire SFC request without exceeding the maximum number of individual VNF instances that may be installed and used simultaneously by the license owner are formulated by Eq.(8) and Eq.(9): Eq.(8) guarantees that every VNF type (N) belonging to the same SFC request is effectively deployed at least once and no more than $L_{vnf_i^{r_n}}$ times. Eq.(9) imposes limitation on the amount of requested CPU resources of the VNF that should not exceed the available resources of the PSN node on which it will be embedded. Eq.(10), indicates that the virtual link bandwidth $VLink(i,l)^{r_n}$ should not exceed that of the PSN link $link(j,k)$ of the PSN on which it will be mapped. Eq.(11) (resp. Eq.(12)) explicitly expresses the cumulative delay incurred by all the PSN nodes (resp. PSN links) involved in the SFC request Dn_{r_n} (resp. Dl_{r_n})is expressed as the sum of the processing delay required to map every VNF instance $vnf_i^{r_n}$ (resp. virtual link $VLink(i,l)^{r_n}$) of the SFC request r_n on the corresponding PSN node $nfvi_j$ (resp. PSN link $Link(j,k)$). Eq.(13) indicates that the end-to-end SFC request expected delay must not exceed the given threshold D_{r_n}.

4.2 MDP Based QoE/QoS Aware SFC Orchestration

We formalize the SFC orchestration issue as a MDP model and we illustrate from reinforcement learning perspective the interaction between the RL agent with the controlled environment interact over a sequence of discrete time steps $t = 0, 1, 2, \ldots, T$ to achieve a goal.

4.2.1 MDP Specification

MDP is formally described by the following tuple: $\{S, A, P, R, \gamma\}$:

- S: denotes the finite set of all possible of states (the state space). Each state $s \in S$ represents the system environment including PSN topology, QoE/QoS status of VNF instance, functional and QoS requirements of the current SR, etc.
- A denotes the finite set of all possible actions (the action space). Each action $a \in A$ indicates selecting the next VNF instance from the current one. To deploy a SR, The first action corresponds to embedding the first VNF on PSN node and the second action corresponds to embedding the second VNF and mapping virtual link $VLink(1, 2)$ interconnecting VNF_1 to VNF_2 on PSN link. The following actions (3, 4 and 5) are executed in a similar manner to action 2. In other words, the i_{th} action $(2 \leq i \leq 5)$ embeds VNF_i on a PSN node and maps $VLink(i-1, i)$ interconnecting VNF_{i-1} to VNF_i on a suitable PSN link. Notice that, for the i^{th} VNF it's possible to choose an action among M_i possible actions.
- P denotes the state transition probability matrix, a function of transition probabilities between states (conditioned to the action taken by the agent). Each transition probability $P^a_{s \to s'}$ represents the case where the QoE/QoS status moves from s to s' under the action a enabling the selection of a VNF instance.
- R denotes reward function (the immediate reward) assigning a specific reward to each state-action pair. Each immediate reward $r \in R$ describes the contribution of the chosen VNF instance vnf_{ij} to the current QoE of the chain.
- $\gamma \in [0, 1]$ is the discount factor, future rewards are discounted at a given rate to bring more importance to immediate (current) reward compared to the future rewards and to ensure convenient trade-off between short term and long term rewards. The convergence of the discounted reward sum is of paramount importance when evaluating the performance of reinforcement learning algorithms.

4.2.2 MDP Process Description

The evolution in time of the Markov decision process is detailed as follows: At every time step t the agent receives state representation of the environment state s_t and performs an action a_t. The objective of the agent is to learn a policy that maps states to appropriate actions in a way that maximizes a reward signal furnished by the environment. As a feedback to the action of the agent the environment moves from the current to the next state s_{t+1} by achieving a Markovian transition. At the next time step $t+1$ the agent receives a reward r_{t+1} assessing its performance and uses this information with the current state s_{t+1} to choose the next action a_{t+1}. The received reward is selected based on the chosen action and the state transition from the current state to the new (next) state of the environment.

The resolution of MDP enables the agent to learn the optimal action to take in every state, for the sake of maximizing the expected cumulative rewards. The quality of a policy $\pi(s, a)$ is assessed by a long term reward over some time horizon instead of an immediate reward r, since a good action is always the best in the long-term. As a result, assigning immediate short-term reward is unprofitable to evaluate the policy quality and to guide the search for an optimal policy. The optimal action is obtained via two value functions:

- The state value function $V^{\pi}(s)$ defined as the expected accumulated discounted rewards starting from s and following policy, π (Eq.(14)). It provides the value of being in a given state.

$$\begin{aligned} V^{\pi}(s) &= E(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \| s_t = s) \\ &= E(r_{t+1} + \gamma V^{\pi}(s_{t+1}) \| s_t = s) \end{aligned} \tag{14}$$

- The state-action value function $Q^{\pi}(s,a)$ (Q function), extends $V^{\pi}(s)$ and takes into consideration the action taken. It specifies the utility for an agent to perform a particular action in a specific state following a fixed policy π and represents the expected accumulated discounted rewards by action a from initial state s (Eq.(15)).

$$\begin{aligned} Q^{\pi}(s,a) &= E(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \| s_t = s, a_t = a)) \\ &= E(r_{t+1} + \gamma V^{\pi}(s_{t+1}, a_{t+1}) \| s_t = s, a_t = a) \end{aligned} \tag{15}$$

where r_t represents the immediate reward of step $t \in \{1, 2, ..., T\}$ and $E(.)$ indicates the mathematical expectation operator. Finding the optimal solution of MDP involves finding the policy that maximizes $V^{\pi}(s)$ (Eq.(16)):

$$\pi^*(s) = argmax V^{\pi}(s) \tag{16}$$

According to Bellman Optimality Equation, the optimal policy π^* is given as follows (Eq.(17)):

$$V^{\pi^*}(s) = max_{a \in A} Q^{\pi^*}(s,a) \tag{17}$$

The resolution of MDP SFC orchestration model seeks the optimal SFC instance $sfci_{c*}$ $c* \in C$ (among a finite set of $sfci$ denoted by C) according to the policy π such that (Eq.(18)):

$$c^* = argmax_{c \in C} E(\sum_{t=0}^{T} \gamma^t r_{t+1}) \tag{18}$$

4.3 The Reward Model: Global Formula and Building Blocks

In order to determine the optimal policy in a RL problem that maximizes the expected cumulative reward over time and over all possible sequences of states and actions, it's of paramount importance to build a the reward model (reward function), enabling to map states and actions to the corresponding rewards. The reward model represents the cornerstone of the decision-making process in reinforcement learning. It enables the quality evaluation of different actions taken by the agent and guides its learning process. In the rest of this subsection, we provide first a global formula of the proposed reward model. We describe next the different building blocks involved in the reward model conception.

4.3.1 The Global Formula

Formulating the SFC orchestration/deployment issue as a RL problem requires defining a rewards model in order to associate convenient rewards/penalties to successful/failed deployment actions. We adopt, in this paper, reward/penalty assignment only for VNF embedding and we consider that whenever two successive VNFs are successfully deployed they are automatically interconnected through a virtual link successfully mapped on a PSN link. The successful embedding of the next VNF depends on checking the achievement of two conditions: (i) finding a suitable PSN node with enough available resource to handle it and (ii) verifying that the mapping of the virtual link interconnecting the first VNF to the next VNF is achievable on appropriate PSN link (with sufficient idle resources). Furthermore, and for convenience, we assume that not only the mapping action is always achieved automatically and successfully but also no reward is assigned to such action. Notice also that every embedding/mapping action along the SR deployment process is preceded by a preliminary capacity constraints check by the DRL agent. This kind of control is repeated iteratively until the entire SR deployment process is completely achieved. In the rest of paragraph, we will detail all the rewards and/or penalties assigned to each action taken by the DRL agent. Whenever SR deployment process is initiated, if the embedding action of the first VNF is correctly achieved according to the chosen (random in this case) policy, a reward 0 is assigned, otherwise the current SR deployment process is considered failed and interrupted and a penalty (P) is assigned to such action. If the first VNF embedding action is successfully fulfilled, the DRL agent should verify any node capacity constraint before beginning the next VNF embedding action. The following action of the DRL agent is to establish the mapping action of the virtual link (onto a suitable PSN link) interconnecting the first VNF to the following one, stated in the SR. The previous embedding/mapping actions are incrementally repeated for every VNF/virtual link until fulfilling the entire SR deployment task. Depending on the success or fail of the VNF embedding action we assign a reward or penalty according to Eq.(19). In this regard, we assume that each one of the first $(N-1)$ VNFs (vnf_i, $i \in [1...N-1]$) of the SFC request, successfully deployed, obtains a reward $R_{QoE-QoS}$ equals to 0 (Eq.(19a)) whereas a successful deployment of the last VNF (the N^{th} VNF, vnf_N) obtains a positive reward value equals to QoE_{sfci_c} (Eq.(19b)) (resp. $QoE_{sfci_c} - P^{r_n}_{sfci_c}$ (Eq.(19c))) whenever condition c_2 (resp. c_3) is satisfied. We assume also that the fail of one action corresponding to VNF embedding (among the set of VNFs composing the SR) yields to cancelling the entire SR deployment and produces a negative reward ($P = -10$) (Eq.(19d)).

$$R_{QoE-QoS} = \begin{cases} 0, & \text{if } (c_1) \quad (19a) \\ QoE_{sfci_c}, & \text{if } (c_2) \quad (19b) \\ QoE_{sfci_c} - P^{r_n}_{sfci_c}, & \text{if } (c_3) \quad (19c) \\ P, & \text{otherwise} \quad (19d) \end{cases}$$

where c_1, c_2, c_3 are logical conditions defined as follows:

- c_1: vnf_i $(i \in [1...N-1]) \in r_n$ is successfully deployed
- c_2: $vnf_N \in r_n$ is successfully deployed and $QoS^t_{sfci_c} = QoS^t_{r_n}, t \in \{1, 2, .., L\}$
- c_3: $vnf_N \in r_n$ is successfully deployed and $QoS^t_{sfci_c} \neq QoS^t_{r_n}, t \in \{1, 2, .., L\}$

For convenience, we may rewritten Eq.(19) in a compact representation and unified form given by (Eq.(20)).

$$R_{QoE-QoS} = \delta_{r_n}.(QoE_{sfci_c} - P^{r_n}_{sfci_c}) - (1 - \delta_{r_n}).P \tag{20}$$

where δ_{r_n}, QoE_{sfci_c} and $P^{r_n}_{sfci_c}$ are given as follows:

- δ_{r_n} (Eq.(21)): represents a decision variable indicating whether the whole SFC request r_n has been successfully deployed or not. Hence, δ_{r_n} equals 1 if SFC request finds a suitable SFC instance on PSN and is effectively deployed and 0 otherwise.

$$\delta_{r_n} = \begin{cases} 1, & \text{if } vnf_i \ (i \in [1...N]) \in r_n \text{ issuccessfully deployed} \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

- QoE_{sfci_c} (Eq.(22)) [45] [46]: is the overall QoE gain of a successfully deployed SR and is given by:

$$QoE_{sfci_c} = \sum_{t=1}^{K} w^t \times QoE^t_{sfci_c} - \sum_{t=K+1}^{L} w^t \times QoE^t_{sfci_c} \tag{22}$$

QoE_{sfci_c} quantifies the reward obtained in response to the effective SFC deployment where:

$$QoE^t_{sfci_c} = \begin{cases} \gamma_p \times log(\alpha_p \times qos^t_c + \beta_p) + \theta_p, \ t \in \{1, 2, .., k\} \\ \gamma_n \times e^{(\alpha_n \times qos^t_c + \beta_n)} + \theta_n, \ t \in \{k+1, k+2, .., L\} \end{cases} \tag{23}$$

It is worth noting that the constant parameters α_p, β_p, γ_p, θ_p, α_n, β_n, γ_n and θ_n may be used to achieve fine adjustment of the quantitative inter-dependency between QoE and QoS. Notice also that $QoE^t_{sfci_c}$ denotes the QoE gain related to the t^{th} QoS metric of $sfci_c$.

- $P^{r_n}_{sfci_c}$ (Eq.(24)) [35]: is the QoS constraints penalty given by:

$$P^{r_n}_{sfci_c} = P \cdot e^{-\sqrt{\sum_{t=1}^{L} \| QoS^t_{sfci_c} - QoS^t_{r_n} \|^2}} \tag{24}$$

4.3.2 The Building Blocks

A well-designed reward model will certainly contribute to efficient and effective learning. From this perspective, We adopt a modular approach for building the reward model based on separated building blocks where each block is designed to perform a specific task. The advantage of such approach is to enable reward modeling flexibility fostering easier reward extensions. We can easily extend the proposed reward model by accounting for an additional building block quantifying the OPEX cost related to SR deployment. This is out of the scope of this paper. In the rest of this paragraph, we consider and we detail two main building blocks potentially in the global formula of the proposed reward model (see paragraph 4.3.1). These building blocks concern QoE Gain and QoS constraints penalty.

QoE Gain (QoE_{sfci_c}): The QoE gain is conceived according to QoE control mechanism based on measurable QoS parameters bringing positive experience. The objective of such mechanism is to:

- Identify key performance QoS metrics that affect the user's QoE (such as delay, bandwidth)
- Implement continuous monitoring tools enabling effective measure of QoS metrics
- Set a mapping between QoS and QoE. Such mapping will be useful to establish a correlation between high latency in a video streaming for example and poor video quality and/or a negative user experience [47] [48].
- Set QoS thresholds to ensure that the QoE remains within an acceptable range.
- Establish a control mechanism enabling dynamic adjustment of network and application parameters in order to preserve QoS into an acceptable range accounting for the defined thresholds and to enhance QoE.

It is worth mentioning that QoE assessment may be based on subjective or objective QoS measures. In our context and for cost and complexity reasons, we focus only on objective measures. The main QoS objectives measures are established according to two major mathematical models referred to as WFL and IQX [45] [46]. They provide mapping between QoS and QoE in telecommunications and networking through non-linear relationships. Weber-Fechner Law (WFL) [46] [49] establishes a logarithmic dependency between QoS metrics and the resulting QoE whereas the IQX hypothesis expresses the generic Exponential Inter-dependency between QoE and measurable QoS (Eq.(23)). WFL is used for mapping positive QoS metrics (i.e. bandwidth) where the larger the value, the better the QoS, while IQX is used for mapping negative QoS metrics (i.e. delay) [45] where the smaller the value, the better the QoS. The major difference between mapping in WFL and IQX is that WFL mapping is based on differential function, whereas IQX mapping is based on an exponential function. Each model is used according to the specific requirements of the QoS and QoE metrics being assessed.

The QoS Constraints Penalty ($P^{r_n}_{sfci_c}$): The reward formulation is conceived in order to enable the DRL agent to seek through trials and errors, exploration and exploitation the 'best' SFC instance, among all the SFC instances of the PSN, having the highest QoE while meeting QoS requirements. These requirements lead indeed to two paradoxical needs. On one hand, we tend to consume more resources to further

improve QoE gain and on the other hand we need to take care to rationalize as much as possible resources consumption to avoid unnecessary wastes. Consuming larger resources while avoiding QoS constraint violations fosters providing closer distance between SFC instance QoS metrics and SR QoS requirements. Therefore, whenever QoS constraints are met, the closer the distance, the smaller the penalty should be. Such penalty ($P^{r_n}_{sfci_c}$) is formulated as an exponential function of the square of the Euclidean distance between QoS metrics of $sfci_c$ ($QoS^t_{sfci_c} \in \{Bw_{sfci_c}, D_{sfci_c}\}$) (see Table 1) and the SR QoS constraints ($QoS^t_{r_n} \in \{Bw_{r_n}, D_{r_n}\}$) (see Table 1) (Eq. (24)). To summarize, whenever QoS requirements are met two possibilities are to be considered. The first one corresponds to the case where the difference between QoS metrics and QoS constraints is zero the reward in such case is equals to the QoE gain. The second one corresponds to the case where this difference is not null, and in order to discourage such behavior a proportionate penalty is assigned and the reward value is given by Eq.(24). Also, in order to severely penalize any QoS violations our choice has been made towards a sufficiently large constant value of the penalty P. These violations may occur whenever an imbalance could arise between the two sides of the equation Eq.(10) and/or Eq.(13). Notice also that any QoS constraint transgression leads without fail to the strictest penalty (P). It should be pointed out that QoS metrics of $sfci_c$ may concern bandwidth and/or delay and each QoS metric violation may be penalized in an appropriate way in order to attain the predefined goal.

4.4 DRL Driven QoE/QoS Aware SFC Orchestration

The NP hard stochastic optimization problem related to QoE/QoS aware SFC orchestration is then established as a Markov decision process (MDP) and will be solved according to a DRL approach. Indeed, taking into consideration the complex and large-scale nature of the decision-making problems related to the QoE.QoS aware SFC orchestration and given the scalability issues of Q-learning we adopt in the rest of this paper the DRL approach for solving the MDPs. This choice is justified by the DRL ability to handle high-dimensional state spaces, complex patterns of the reward structures, and long-term dependencies in sequential decision-making problems. By adopting the DRL approach and through using the action value function the expected reward (to maximize) of each action (i.e., selecting a specific VNF instance) to take in a given state may be evaluated. In this regard the VNF selection is dynamically adapted to changing network conditions and resource constraints. This enables optimizing accordingly the network service deployment in real-time, leading to better network performance and streamlining resource utilization. In the rest of this subsection we provide first a detailed description of the VNF instances selection process, we describe next the action value function pf our reinforcement learning problem before presenting the proposed algorithm.

4.4.1 Action Value Function

RL algorithms require an exact representation of the two major components referred to as value function and policy that are used by the agent in decision making. Whenever the state-action space is small enough to be represented in a table, tabular methods

such as Q-learning are suitable to keep track of the states, actions, and their expected rewards. In such case the agent updates the Q-values in the table to represent its updated estimate of the expected reward for taking each action in each state. However in complex environments with large-scale or continuous state space, tackling the RL problem exactly or incrementally it is often prohibitive. The action value function $Q(s, a)$ is used to define the long term as the sum of all the discounted immediate rewards r. As soon as a VNF instance vnf_{ij} is selected, immediate reward r_{ij} is assigned, and accordingly used to update the current value of action value function $Q(s, vnf_{ij})$. Given that the QoE/QoS state space is continue, tabular based RL are inconvenient for storage and updates and DRL implementations via DQN and its variants are used to fit each long-term reward $Q(s, vnf_{ij})$ from a given state s and the corresponding immediate reward r_{ij}. Instead of storing all state-action pairs in a Q-table like in Q-learning, DQN, a variant of Q-learning, relies on Neural Network (NN) (Convolutional NN (CNN) or Deep NN (DNN)) to approximate complex and nonlinear function, referred to as action value function $Q(s, a)$.

DQN relies on two separate and structurally identical networks, namely evaluation network (eval-net) (or online network) and target network (target-net). DQN aims through such separation to enhance the learning process stability and to attenuate and to prevent oscillations and divergence in the estimated Q-values. The target network delivers a stable target for the eval-net to learn from, since the Q-values estimated by the eval-net may rapidly change while updating its parameters. The eval-net is used to select actions and estimate the Q-values for each state-action pair, while the target network is used to generate the target values for updating the evaluation network's weights.At the end of each iteration, eval-net updates its parameter θ whereas the target-net, which is a fixed copy of the evaluation network, temporarily frozen its parameter θ^- and updates it periodically after a set iterations C.

DQN algorithm [50] may be seen as a black-box function taking as input the state of the environment and outputs a Q-value (the expected reward) for each possible action. It is worth mentioning also that the neural network is used to map for each action the state of the environment to a Q-value. The goal of the DQN algorithm is to learn the optimal Q-value function enabling to know the best action in a certain state and following a given policy thereafter and such information may be used by the agent for real time decision making. The update of the Q-value function is achieved through trial-and-error, and based on the experiences stored in a replay memory to progressively enhance the Q-value predictions accuracy. During the training process DQN agent adopts the Temporal Difference (TD) Error as a loss function is defined as the Mean Square Error (MSE) to quantify the difference between the expected reward for a given state-action pair (as predicted by the network) and the actual reward obtained once the action in that state is taken. This difference is afterwards leveraged for updating the network's weights and performing better predictions. The TD Error relies on the Bellman Equation to define the expected return for each state-action pair in a Markov Decision Process. Optimizing the TD Error (MSE loss function) with respect to the network's weights enables the DQN agent to learn the optimal Q-value function, which in turn enables to find the best action to perform in each state. The loss function minimization is being possible through updating the weights. Optimizing

loss function by using a gradient (Eq.(26)) is then leveraged to minimize the training of neural network.

$$L(\theta) = E[(r_{ij} + \gamma max_{vnf'_{ij}} Q(s', vnf'_{ij}; \theta^-) - Q(s, vnf_{ij}; \theta))^2] \tag{25}$$

$$\frac{\partial L(\theta)}{\partial \theta} = E[(r_{ij} + \gamma max_{vnf'_{ij}} Q(s', vnf'_{ij}; \theta^-) - Q(s, vnf_{ij}; \theta)) \frac{\partial Q(s, vnf_{ij}; \theta)}{\partial \theta}] \tag{26}$$

4.4.2 VNF Instances Selection

The DRL agent strives to learn how to map PSN states to actions so as to maximize a numerical reward signal that quantifies the numerical feedback for the actions taken in uncertain environment. The objective of the learning process is to choose the convenient VNF instance by alternating between exploration and exploitation. This enables to seek for good balance between trying new strategies (exploration) and sticking with successful strategies already found in the past (exploitation). To handle the exploration-exploitation dilemma we opt for a simple and effective exploration strategy based on the ϵ-greedy algorithm. This algorithm is used in the Multi-Armed Bandit (MAB) problem to balance the trade-off between exploring different (new) arms (actions) to acquire new knowledge and exploiting the best arm based on existing knowledge. It's also used in simple tabular reinforcement learning, to find a compromise between exploring new actions selected at random to collect information about the environment, and exploiting the best actions providing the maximum future reward based on the available information gathered so far. In ϵ-greedy algorithm the agent randomly alternates between exploration and exploitation. The choice of randomly choosing to explore a new action with probability ϵ whereas exploitation consists in choosing the best action based on current knowledge with probability $1 - \epsilon$ (Eq.(27)). In our case study, exploration seeks to try out new VNF instances that have not been yet executed, whereas exploitation attempts satisfy the need to stick to the known successful VNF instances by choosing the best VNF instance among those known enabling to obtain the maximum known return with a relatively conservative approach.

$$\pi = \begin{cases} 1 - \epsilon + \frac{\epsilon}{M_i}, & \text{if } vnf_{ij} = argmax_{j=1}^{M_i} Q(s, vnf_{ij}) \\ \frac{\epsilon}{M_i}, & \text{if } vnf_{ij} \neq argmax_{j=1}^{M_i} Q(s, vnf_{ij}) \end{cases} \tag{27}$$

4.4.3 Implementations of the DRL Driven QoE/QoS Aware SFC Orchestration Algorithm

Based on the modeling approach described above, we propose in what follows a DRL implementation of the tackled QoS/QoE-aware SFC orchestration problem. This implementation (Algorithm 1) concerns Double DQN algorithm (detailed in subsection 3.5) and aims to evaluate the DRL agent behavior along the training process. In the algorithm 1, we use the following parameters:

- BS (mini-Batch Size) is the mini-batch size.
- RS (Replay buffer Size) is the buffer size of the replay memory D.
- Q is a vector of dimension M_i representing $Q(s, vnf_{ij})$s for i^{th} VNF instance vnf_i.

- E specifies the number of training episodes ($E = 5000$).
- Req defines the number of SRs used during the training process ($Req = 100$).
- QoE_{Sc-Th} is a threshold score. It quantifies a QoE learning quality to reach (on average) by the DRL agent along the last 100 runs.

These algorithms will be extensively explored by simulation in the next section in order to highlight the DRL agent behavior along the training process for different hyper-parameters and performance metrics.

Algorithm 1 Double DQN_QoS/QoE_SFC

initialize replay memory D to capacity RS, $\tau << 1$
initialize action value function Q_θ, with random weights θ
initialize target action value function $Q_{\theta'}$ with weights $\theta^{'}$
initialize QoE_{Sc-Th}, $Sum = 0$
for $episode = 1..E$ **do**
  `reset environment`
  $Sum_r = 0$
  for $sfc\_req = 1..Req$ **do**
    `/*Sampling phase`
    `initialize chain` c `and observe initial observation` s
    for $i = 1..N$ **do**
      `select a connected instance` vnf_{ij} `by eq.(27)`
      `observe` s `by QoS and observe` r_{ij} `by eq.(19)`
      `store transition (`s`,`vnf_{ij}`,`r_{ij}`,`s'`) in` D
      s=s'
    end for
    $Sum_r += r_{ij}$
    `/*Learning phase`
    if enough experiences in D **then**
      for `each update step` **do**
        `sample mini-batch of` BS `transitions` $(s, vnf_{ij}, r_{ij}, s')$ `from` D
        `compute target` Q `value:` $Q^*(s, vnf_{ij}) \approx r_{ij} + \gamma Q_\theta(s', argmax_{vnf'_{ij}} Q_{\theta'}(s', vnf'_{ij}))$
        `perform a gradient descent step on` $(Q^*(s, vnf_{ij}) - Q_\theta(s, vnf_{ij}))^2$
        `update target network parameters:` $\theta' \leftarrow \tau * \theta + (1 - \tau) * \theta'$
      end for
    end if
  end for
  `calculate` Sum `the sum of last 100 mean rewards (`(Sum_r/Req)`)`
  if $(Sum/100) \geq QoE_{Sc-Th}$ **then**
    break
  end if
end for

5 Performance Evaluation

In this section, we provide a simulation-based performance analysis to evaluate the RL effectiveness of the proposed algorithm dealing with online incremental QoE/QoS aware SFC orchestration problem. This algorithm uses Double DQN and aims to achieve the optimal deployment scheme of each incoming SR. We conduct several simulation experiments to evaluate and test the learning capacity of the DRL agent to reach a predefined QoE score used as a performance training metric. We particularly focus on finding a suitable compromise between performance in terms of learning quality and convergence of the training process. In the rest of this section we give first the mains assumptions considered for the DRL agent environment. We detail next the different experimental results related to the effect of both PSN scaling and hyper-parameters (referred to as batch size and learning rate) on performance-convergence trade-off.

5.1 Main Assumptions

The definition of the simulation environment and the input parameters involves specifying the main assumptions related to hyper-parameters (Table 3), the PSN topology, the workload (the incoming SFC requests (SR)), and the considered performance metrics to investigate.

- PSN assumptions:
  - PSN nodes: Each PSN node is assumed of unlimited capacity and can host one or more VNF(s).
  - PSN links: Each PSN link is assumed with limited bandwidth capacity (Bw_{link}) and non zero transmission delay (D_{link}). It may handle one or more Vlink(s). Bw_{link} is randomly chosen in the range of [768−1280]Mbps, and D_{link} is randomly selected in $[10-20]$ms.
- PSN attributes:
  - M: The number of nodes in PSN (M=5).
  - M_{link}: The number of links in PSN ($M_{link} = M*(M-1)/2 = 10$), (PSN nodes are assumed fully interconnected via PSN links).
- SR attributes:
  - N_{VNF}: The number of VNF instances (N_{VNF}=5) involved in the SR. We assume that the embedding of each VNF instance on PSN node requires no CPU capacity and no processing delay. Obviously this assumption may be easily relaxed to assign a non null processing capacity and non null time delay in the PSN node. This is out of the scope of this paper.
  - N_{Vlink}: The number of Vlinks (N_{Vlink}=4).
  - Bw_r: End-to-End Bandwidth varies randomly in the range of $[16-256]$Mbps.
  - D_r: End-to-End Delay varies randomly in the range of $[50-90]$ms.
- Evaluation metric for online DRL: Is used for evaluation during training (learning) phase. It measures the quality of the learned policy (or RL algorithm) and is

quantified by the average learning level (i.e., QoE Threshold Score (QoE_{Sc_Th})) or the cumulative reward to be reached by the DRL agent in the last 100 runs of the training phase.

- Scenario: The attributes of each incoming SR and the PSN change rapidly over time and the agent needs to adapt quickly and dynamically to these new conditions.

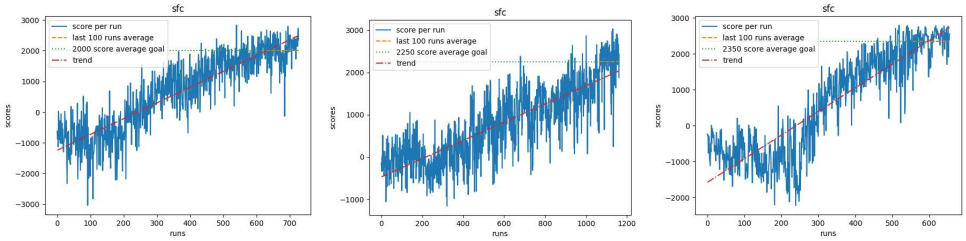
5.2 Experimental Results

5.2.1 Effect of PSN Scaling on Performance-Convergence Trade-off

We investigate in the rest of this subsection performance-convergence trade-off along the training process of the DRL agent and how the PSN scaling impacts this trade-off. In this respect, we examine two sets of experiments (PSN with 5 (set1) and 10 (set2) PSN nodes respectively). The agent training process is examined along one episode (one episode = 5000 runs). For the two sets, the DRL agent aims to reach as soon as possible a predefined learning quality quantified by the evaluation metric (QoE_{Sc_Th}). For each experiment, the DRL agent may attain or not the desired learning quality. In order to distinguish between experiences where the agent succeeded in reaching QoE_{Sc_Th} from experiences where the agent fails in reaching QoE_{Sc_Th} during the same episode, we intentionally split each set into two separated subsets (set1 into subset11 and subset12 and set2 into subset21 and subset22). Subset11 and subset21 cover experiments where the DRL agent succeeded in reaching QoE_{Sc_Th} before an episode ends. Whereas the subset12 and subset22 cover experiments where the DRL agent fails in attaining the desired quality within one episode. Reaching the learning quality objective may require one episode, more than one episode or may be never achieved at all regardless of the number of runs (in this case the convergence is not ensured in a finite number of episodes). Toward the different experiences, we investigate performance and convergence issues of the DRL agent for different QoE_{Sc_Th}. In order to attempt to learn the best (optimal or near-optimal) policy, we progressively and incrementally increase the QoE_{Sc_Th} and we intend to see for each experience how this impacts simultaneously the learning quality and the learning convergence speed.

In set1 we distinguish between two subsets: subset11 (fig.2: fig.2a, fig.2b, fig.2c, fig.2d and fig.2e) and subset12 (fig.3: fig.3a and fig.3b). Notice that in subset11 rising QoE_{Sc_Th} does not automatically lead to a longer learning phase. Indeed, for example, increasing QoE_{Sc_Th} from 2000 (fig.2a) to 2250 (fig.2b) yields a growth of the required number of runs from 700 to 1150 to attain the desired QoE_{Sc_Th} . Conversely, increasing QoE_{Sc_Th} from 2250 (fig.2b) to 2350 (fig.2c) yields a decrease of the required number of runs from 1200 to 670 to reach the predefined QoE_{Sc_Th} . This behavior may be explained by the following justifications. Along the training phase, the agent learns through trial and error, and incrementally builds online decision making by choosing randomly between exploration and exploitation in a probabilistic manner according to ϵ-greedy algorithm. In a nutshell, the training process is stochastic and the learning phase may last for experiences with the same setting more or less long period. The above explanation fully justifies the agent behavior in subset11 for different QoE_{Sc_Th}.

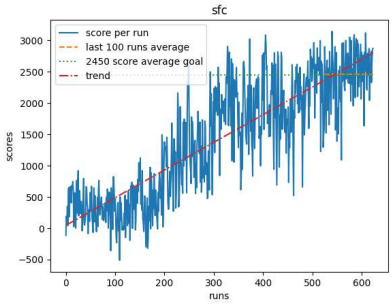
In subset12, for QoE_{Sc_Th} relatively high (2700 in fig.3a) and 3000 in fig.3b), obviously the learning episode may finish without allowing the agent to attain the



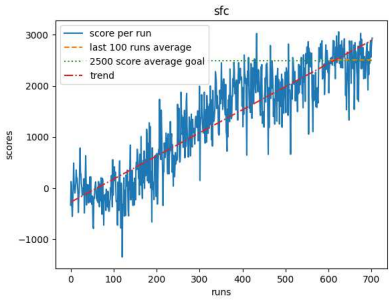
(a) $QoE_{Sc-Th} = 2000$

(b) $QoE_{Sc-Th} = 2250$

(c) $QoE_{Sc-Th} = 2350$



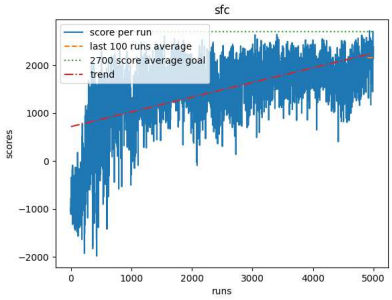
(d) $QoE_{Sc-Th} = 2450$



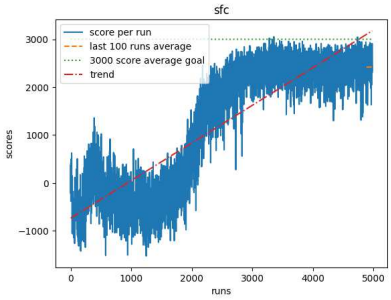
(e) $QoE_{Sc-Th} = 2500$

Fig. 2: QoE score versus number of runs ($M = 5$)

desired learning quality. Therefore the agent is very likely unable and tries in vain to perform actions, during the last 100 runs of the training phase, that make it more profitable to improve the training performance. Such behavior is due probably to the performance limitation of the neural network model and/or the DRL algorithm (Double DQN).



(a) $QoE_{Sc-Th} = 2700$

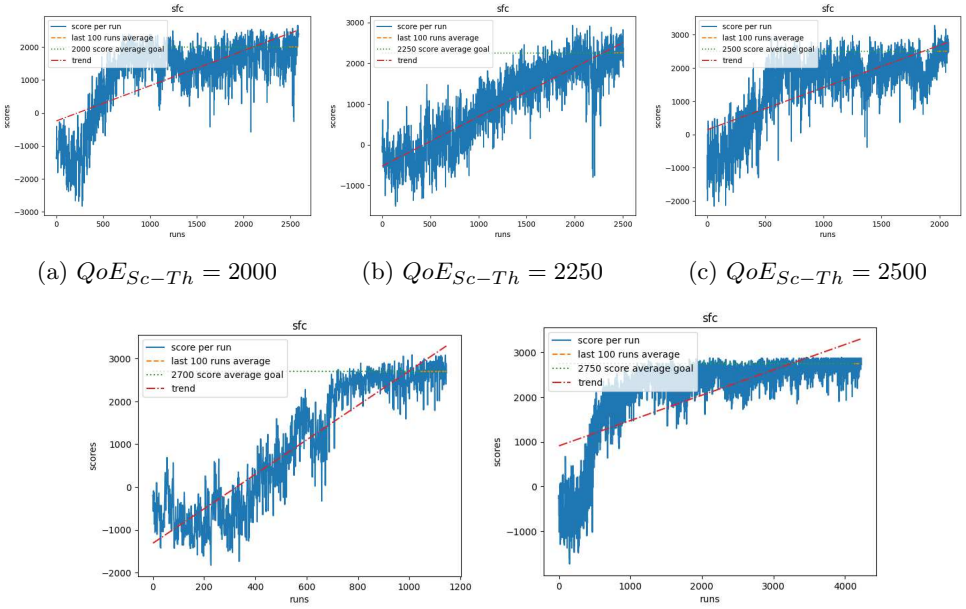


(b) $QoE_{Sc-Th} = 3000$

Fig. 3: QoE score versus number of runs ($M = 5$)

In a second set of experiments (set2: fig.4 and fig.5), we adopt similar setting as in set1 (fig.2 and fig.3) but with a larger PSN scale (a number of PSN nodes=10)). Through numerical investigations (fig.2–fig.5), we attempt, for a given QoE_{Sc_Th} preliminary defined, bringing convincing answers to the following question: How increasing PSN scale impacts performance and convergence of the DRL agent?

For relatively low QoE_{Sc_Th} (QoE_{Sc_Th} = 2000, 2250, ..., 2500), the convergence is guaranteed for the two sets (set1 and set2) in a number of runs less than 5000 (One episode = 5000). On the other hand, set2 provides slower convergence than set1. For relatively high QoE_{Sc_Th} the agent converges (for QoE_{Sc_Th}=2700 (fig.4d), 2750 (fig.4e) in set2) in a number of runs less than one episode, whereas for QoE_{Sc_Th}=2700 (fig.3a in set1) the agent fails in converging in a laps of time within the same episode. For higher QoE_{Sc_Th}, the agent convergence is unreachable within an episode (QoE_{Sc_Th} = 3000 (fig.3b) of set1 and QoE_{Sc_Th} = 2800 (fig.5a), 3000 (fig.5b) in set2).



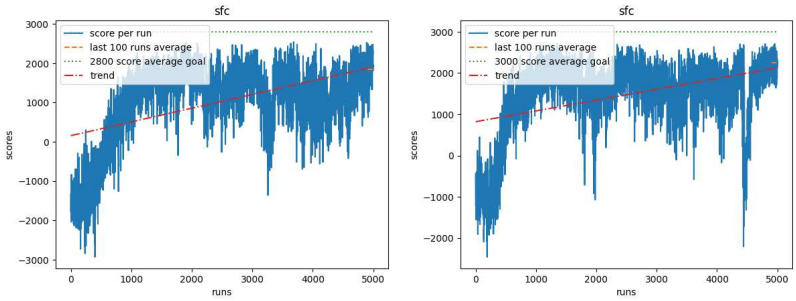
(a) QoE_{Sc-Th} = 2000 (b) QoE_{Sc-Th} = 2250 (c) QoE_{Sc-Th} = 2500

(d) QoE_{Sc-Th} = 2700 (e) QoE_{Sc-Th} = 2750

Fig. 4: QoE score versus number of runs ($M = 10$)

Two major lessons can be learned from increasing the network scale (the PSN state space becomes larger) and when comparing set2 to set1. First, in set2, the DRL agent needs more exploration and hence more time to converge to an optimal policy. This is illustrated by comparing the two curves (fig.2a, fig.4a), indeed for the same algorithm (Double DQN) a larger network scale involves slower convergence (for example the convergence speed to reach the learning quality (QoE_{Sc_Th} = 2000) decreases significantly from 700 to 2500). Second, the DRL agent has greater opportunities to find a

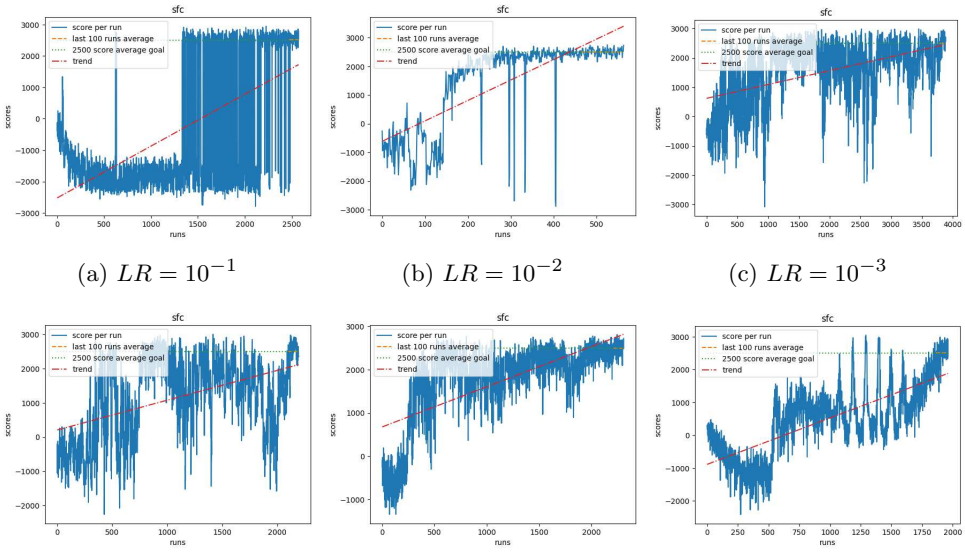
suitable SFC instance among the set of SFC instances meeting SFC request requirements and hence has more chance to reach higher QoE_{Sc_Th}. This fact is significantly highlighted by comparing ((fig.3a, fig.4d).



(a) $QoE_{Sc-Th} = 2800$ (b) $QoE_{Sc-Th} = 3000$

Fig. 5: QoE score versus number of runs ($M = 10$)

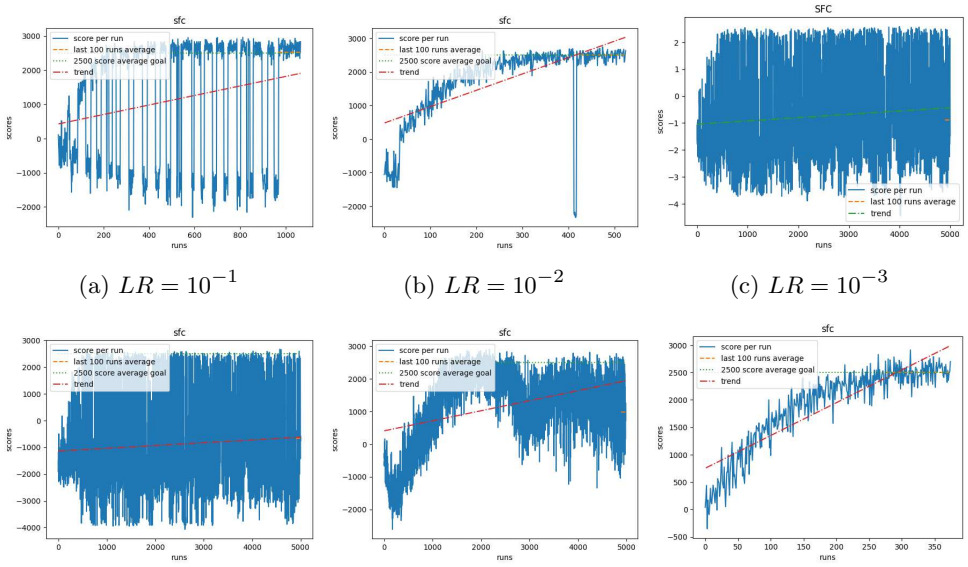
5.2.2 Effect of Hyper-parameters on Performance-Convergence Trade-off



(a) $LR = 10^{-1}$ (b) $LR = 10^{-2}$ (c) $LR = 10^{-3}$

(d) $LR = 2.5 * 10^{-4}$ (e) $LR = 10^{-5}$ (f) $LR = 10^{-7}$

Fig. 9: QoE score versus number of runs by Double DQN Algorithm ($QoE_{Sc\_Th} = 2500$, $M = 5$, $BS = 256$)

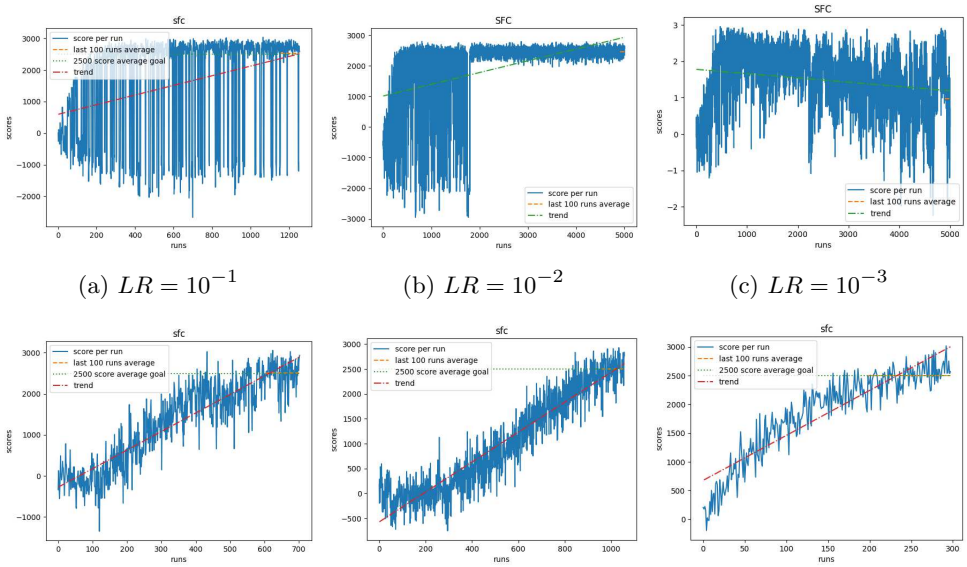


(a) $LR = 10^{-1}$ (b) $LR = 10^{-2}$ (c) $LR = 10^{-3}$

(d) $LR = 2.5 * 10^{-4}$ (e) $LR = 10^{-5}$ (f) $LR = 10^{-7}$

Fig. 6: QoE score versus number of runs by Double DQN Algorithm ($QoE_{Sc\_Th} = 2500$, $M = 5$, $BS = 32$).

In RL, hyper-parameters such as the learning rate (LR) and batch size (BS) can impact significantly performance and convergence of the algorithm. LR may be adjusted to tune the updating rate of the model parameters after each iteration. A higher LR (for example fig.6a ($LR = 10^{-1}$, $BS = 32$) and fig.7a ($LR = 10^{-1}$, $BS = 64$)) yields faster parameters updates, however it may also bring undesirable overshooting of the optimal values leading to oscillations, divergence and instability in the learning process. On the other hand, a lower LR (for example fig.8f ($LR = 10^{-7}$, $BS = 128$) and fig.9f ($LR = 10^{-7}$, $BS = 256$)) can lead to slow convergence and a longer training phase of the agent. Therefore, in order to find convenient trade-off between convergence speed and stability it is important to establish the right choice of the LR with respect to the context. BS is another important hyper-parameter that enables to control how many samples are used in each update of the model parameters. A larger BS (for example fig.9c ($LR = 10^{-3}$, $BS = 256$) and fig.9d ($LR = 2.5 * 10^{-4}$, $BS = 256$)) can bring further accuracy in estimating the gradient and more stability to the learning process, but it may also require additional memory and computational resources. On the other hand, a smaller BS (for example fig.6c ($LR = 10^{-3}$, $BS = 32$) and fig.6d ($LR = 2.5 * 10^{-4}$, $BS = 32$)) can lead to faster updates and more exploration of the state space, but it may also cause more noise and slower convergence. Unsuitable hyper-parameter choices can result in slow convergence, instability, over-fitting, and memory issues and as a result can cause significant performance degradation of RL algorithms and the overall learning process. In this respect and in order to address this challenge and to mitigate the negative impact on the final training outcome, it is



(a) $LR = 10^{-1}$ (b) $LR = 10^{-2}$ (c) $LR = 10^{-3}$

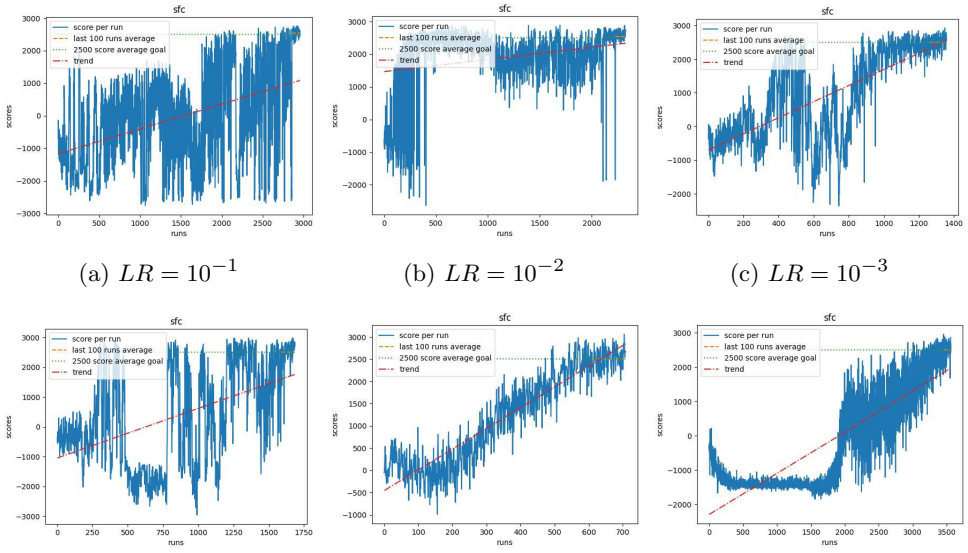
(d) $LR = 2.5 * 10^{-4}$ (e) $LR = 10^{-5}$ (f) $LR = 10^{-7}$

Fig. 7: QoE score versus number of runs by Double DQN Algorithm ($QoE_{Sc\_Th} = 2500$, $M = 5$, $BS = 64$)

important to carefully select these hyper-parameters based on problem specificity and the investigated context.

In what follows, we fix a learning quality goal ($QoE_{Sc\_Th} = 2500$) to reach by the DRL agent and we investigate through extensive experiences how changing hyper-parameters (LR and BS) may impact the model performance in terms of performance-convergence trade-off. In this direction, we keep constant the $BS = 32$ and we assess the learning curve of the reward sum for different LRs (10^{-1} ... 10^{-7}) (fig.6, fig.10a). This process is repeated for different BSs (64 (fig.7, fig.10b), 128 (fig.8, fig.10c) and 256 (fig.9, fig.10d)).

As a synthesis of hyper-parameters effect on performance-convergence trade-off, we can draw the best pairs of LR and BS (fig.6b ($LR = 10^{-2}$, $BS = 32$), fig.6f ($LR = 10^{-7}$, $BS = 32$), fig.7d ($LR = 2.5 * 10^{-4}$, $BS = 64$), fig.7f ($LR = 10^{-7}$, $BS = 64$), fig.8e ($LR = 10^{-5}$, $BS = 128$), fig.9b ($LR = 10^{-2}$, $BS = 256$) that lead to the best training results. These results are better illustrated through representing the learning process evolution of a fixed BS and different LR in the same figure (fig 10). They confirm the default choice ($LR = 2.5 * 10^{-4}$, $BS = 64$ in table **??**) that we had established along the investigated experiments.



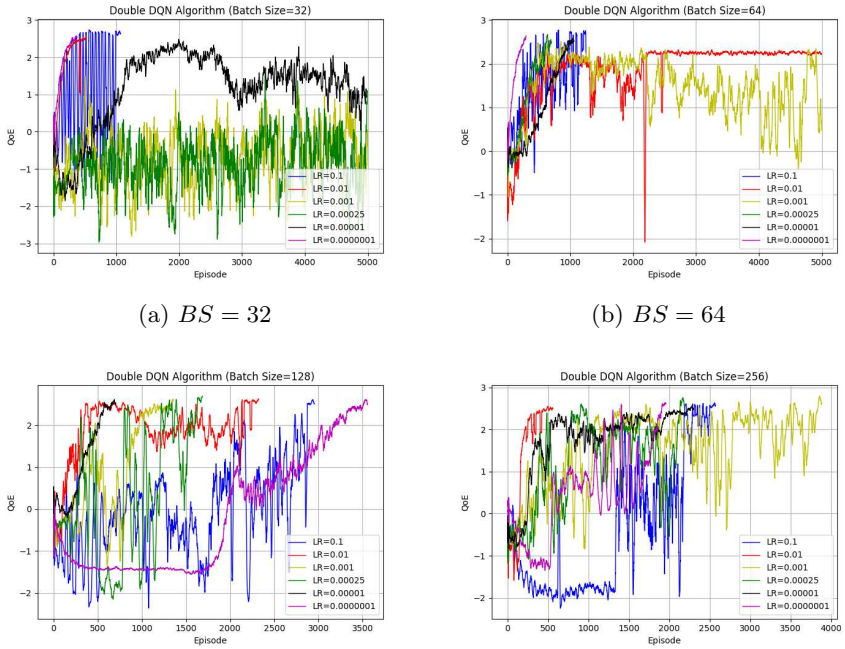
(a) $LR = 10^{-1}$ (b) $LR = 10^{-2}$ (c) $LR = 10^{-3}$

(d) $LR = 2.5 * 10^{-4}$ (e) $LR = 10^{-5}$ (f) $LR = 10^{-7}$

Fig. 8: QoE score versus number of runs by Double DQN Algorithm ($QoE_{Sc\_Th} = 2500$, $M = 5$, $BS = 128$)

6 Conclusion and Future Work

Achieving SFC orchestration and deployment that maximizes QoE while meeting QoS requirements regarding end-to-end bandwidth and end-to-end delays represents a crucial step toward a broader adoption of the NFV concept. We investigate in this paper an online QoE/QoS aware SFC orchestration problem in NFV/SDN-enabled networks based on DRL. The DRL combines DL and RL and aims to yield more effectiveness and stability to function approximations, particularly for high-dimensional and large scale problems. The DRL approach adopted in this paper is implemented through Double DQN by considering extended assumptions about server sharing and limited resources and capacities. These limitations concerns PSN nodes and PSN links, the maximum number of instances per VNF licence.... We illustrate through extensive numerical results the effectiveness of the proposed modeling and analysis method to solve multi-objective sequential making decision problem related to SFC orchestration and deployment by assessing the learning quality of the DRL agent and by highlighting the impact of PSN scaling and hyper-parameters such as batch size and learning rate on performance and convergence. We considered that the learning quality of the agent is satisfactory enough whenever the average score reaches a predefined threshold level on the last 100 runs of each learning experience. In future works we intend to take into consideration affinity and anti-affinity rules in recommending or preventing certain VNFs from sharing the same PSN resource. We also plan to investigate multi-domain context and dependability issues such as robustness in solving the SFC orchestration problem.



(a) $BS = 32$ (b) $BS = 64$

(c) $BS = 128$ (d) $BS = 256$

Fig. 10: QoE score versus number of runs by Double DQN Algorithm ($QoE_{Sc\_Th} = 2500$, $M = 5$)

Declarations

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

- Funding 'Not applicable'
- Conflict of interest/Competing interests (check journal-specific guidelines for which heading to use) 'Not applicable'
- Ethics approval 'Not applicable'
- Consent to participate 'Not applicable'
- Consent for publication 'Not applicable'
- Availability of data and materials 'Not applicable'
- Code availability 'Not applicable'
- Authors' contributions 'Not applicable'

References

[1] Benzekki, K., El Fergougui, A., Elbelrhiti Elalaoui, A.: Software-defined networking (sdn): a survey. Security and communication networks **9**(18), 5803–5833

(2016)

[2] Rowshanrad, S., Namvarasl, S., Abdi, V., Hajizadeh, M., Keshtgary, M.: A survey on sdn, the future of networking. Journal of Advanced Computer Science & Technology **3**(2), 232–248 (2014)

[3] Herrera, J.G., Botero, J.F.: Resource allocation in nfv: A comprehensive survey. IEEE Transactions on Network and Service Management **13**(3), 518–532 (2016)

[4] Bonfim, M.S., Dias, K.L., Fernandes, S.F.: Integrated nfv/sdn architectures: A systematic literature review. ACM Computing Surveys (CSUR) **51**(6), 1–39 (2019)

[5] Barakabitze, A.A., Ahmad, A., Mijumbi, R., Hines, A.: 5g network slicing using sdn and nfv: A survey of taxonomy, architectures and future challenges. Computer Networks **167**, 106984 (2020)

[6] Bhamare, D., Jain, R., Samaka, M., Erbad, A.: A survey on service function chaining. Journal of Network and Computer Applications **75**, 138–155 (2016)

[7] Hantouti, H., Benamar, N., Bagaa, M., Taleb, T.: Symmetry-aware sfc framework for 5g networks. IEEE Network **35**(5), 234–241 (2021)

[8] Özdem, M., Alkan, M.: Subscriber aware dynamic service function chaining. Computer Networks **194**, 108138 (2021)

[9] Carpio, F., Dhahri, S., Jukan, A.: Vnf placement with replication for loac balancing in nfv networks. In: 2017 IEEE International Conference on Communications (ICC), pp. 1–6 (2017). IEEE

[10] Cevallos Moreno, J.F., Sattler, R., Caulier Cisterna, R.P., Ricciardi Celsi, L., Sánchez Rodríguez, A., Mecella, M.: Online service function chain deployment for live-streaming in virtualized content delivery networks: A deep reinforcement learning approach. Future Internet **13**(11), 278 (2021)

[11] Sun, G., Li, Y., Liao, D., Chang, V.: Service function chain orchestration across multiple domains: A full mesh aggregation approach. IEEE Transactions on Network and Service Management **15**(3), 1175–1191 (2018)

[12] Mirjalily, G., Luo, Z.: Optimal network function virtualization and service function chaining: A survey. Chinese Journal of Electronics **27**(4), 704–717 (2018)

[13] Chen, J., Chen, J., Zhang, H.: Drl-qor: Deep reinforcement learning-based qos/qoe-aware adaptive online orchestration in nfv-enabled networks. IEEE Transactions on Network and Service Management **18**(2), 1758–1774 (2021)

[14] Mijumbi, R., Serrat, J., Gorricho, J.-L., Latre, S., Charalambides, M., Lopez, D.: Management and orchestration challenges in network functions virtualization.

IEEE Communications Magazine **54**(1), 98–105 (2016)

[15] Liu, Y., Lu, Y., Qiao, W., Chen, X.: Reliability-aware service chaining mapping in nfv-enabled networks. Etri Journal **41**(2), 207–223 (2019)

[16] Yang, S., Li, F., Trajanovski, S., Chen, X., Wang, Y., Fu, X.: Delay-aware virtual network function placement and routing in edge clouds. IEEE Transactions on Mobile Computing **20**(2), 445–459 (2019)

[17] Pei, J., Hong, P., Xue, K., Li, D.: Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system. IEEE Transactions on Parallel and Distributed Systems **30**(10), 2179–2192 (2018)

[18] Bari, F., Chowdhury, S.R., Ahmed, R., Boutaba, R., Duarte, O.C.M.B.: Orchestrating virtualized network functions. IEEE Transactions on Network and Service Management **13**(4), 725–739 (2016)

[19] Lee, G., Kim, M., Choo, S., Pack, S., Kim, Y.: Optimal flow distribution in service function chaining. In: The 10th International Conference on Future Internet, pp. 17–20 (2015)

[20] Li, D., Hong, P., Xue, K., Pei, J.: Virtual network function placement and resource optimization in nfv and edge computing enabled networks. Computer Networks **152**, 12–24 (2019)

[21] Sahhaf, S., Tavernier, W., Rost, M., Schmid, S., Colle, D., Pickavet, M., Demeester, P.: Network service chaining with optimized network function embedding supporting service decompositions. Computer Networks **93**, 492–505 (2015)

[22] Li, D., Hong, P., Xue, K., *et al.*: Virtual network function placement considering resource optimization and sfc requests in cloud datacenter. IEEE Transactions on Parallel and Distributed Systems **29**(7), 1664–1677 (2018)

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

[24] Bouet, M., Leguay, J., Combe, T., Conan, V.: Cost-based placement of vdpi functions in nfv infrastructures. International Journal of Network Management **25**(6), 490–506 (2015)

[25] Sallam, G., Gupta, G.R., Li, B., Ji, B.: Shortest path and maximum flow problems under service function chaining constraints. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, pp. 2132–2140 (2018). IEEE

[26] Ahvar, S., Phyu, H.P., Buddhacharya, S.M., Ahvar, E., Crespi, N., Glitho, R.: Ccvp: Cost-efficient centrality-based vnf placement and chaining algorithm for network service provisioning. In: 2017 IEEE Conference on Network Softwarization (NetSoft), pp. 1–9 (2017). IEEE

[27] Wu, Y., Zhou, J.: Dynamic service function chaining orchestration in a multi-domain: A heuristic approach based on srv6. Sensors **21**(19), 6563 (2021)

[28] Xu, L., Hu, H., Liu, Y.: Heuristic strategy of service function chain deployment based on n-base continuous digital coding in network function virtualization environment. Electronics **11**(3), 331 (2022)

[29] Shi, R., Zhang, J., Chu, W., Bao, Q., Jin, X., Gong, C., Zhu, Q., Yu, C., Rosenberg, S.: Mdp and machine learning-based cost-optimization of dynamic resource allocation for network function virtualization. In: 2015 IEEE International Conference on Services Computing, pp. 65–73 (2015). IEEE

[30] Lin, S.-C., Akyildiz, I.F., Wang, P., Luo, M.: Qos-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach. In: 2016 IEEE International Conference on Services Computing (SCC), pp. 25–33 (2016). IEEE

[31] Nakanoya, M., Sato, Y., Shimonishi, H.: Environment-adaptive sizing and placement of nfv service chains with accelerated reinforcement learning. In: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 36–44 (2019). IEEE

[32] Chen, J., Chen, J., Hu, R., Zhang, H.: Qmora: A q-learning based multi-objective resource allocation scheme for nfv orchestration. In: 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), pp. 1–6 (2020). IEEE

[33] Sun, J., Huang, G., Sun, G., Yu, H., Sangaiah, A.K., Chang, V.: A q-learning-based approach for deploying dynamic service function chains. Symmetry **10**(11), 646 (2018)

[34] Pei, J., Hong, P., Pan, M., Liu, J., Zhou, J.: Optimal vnf placement via deep reinforcement learning in sdn/nfv-enabled networks. IEEE Journal on Selected Areas in Communications **38**(2), 263–278 (2019)

[35] Chen, X., Li, Z., Zhang, Y., Long, R., Yu, H., Du, X., Guizani, M.: Reinforcement learning–based qos/qoe-aware service function chaining in software-driven 5g slices. Transactions on Emerging Telecommunications Technologies **29**(11), 3477 (2018)

[36] Carofiglio, G., Grassi, G., Loparco, E., Muscariello, L., Papalini, M., Samain, J.: Characterizing the relationship between application qoe and network qos for real-time services. In: Proceedings of the ACM SIGCOMM 2021 Workshop on

Network-Application Integration, pp. 20–25 (2021)

[37] Sutton, R.S., Barto, A.G., et al.: Introduction to reinforcement learning (1998)

[38] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., *et al.*: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)

[39] Lin, L.-J.: Reinforcement Learning for Robots Using Neural Networks. Carnegie Mellon University, ??? (1992)

[40] Sarker, I.H.: Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. SN Computer Science **2**(6), 1–20 (2021)

[41] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)

[42] Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30 (2016)

[43] Hasselt, H.: Double q-learning. Advances in neural information processing systems **23** (2010)

[44] ETSI, G.: Network functions virtualisation (NFV) release 4; management and orchestration; VNF descriptor and packaging specification. ETSI

[45] Fiedler, M., Hossfeld, T., Tran-Gia, P.: A generic quantitative relationship between quality of experience and quality of service. IEEE Network **24**(2), 36–41 (2010)

[46] Reichl, P., Egger, S., Schatz, R., D'Alconzo, A.: The logarithmic nature of qoe and the role of the weber-fechner law in qoe assessment. In: 2010 IEEE International Conference on Communications, pp. 1–5 (2010). IEEE

[47] Zinner, T., Hohlfeld, O., Abboud, O., Hoßfeld, T.: Impact of frame rate and resolution on objective qoe metrics. In: 2010 Second International Workshop on Quality of Multimedia Experience (QoMEX), pp. 29–34 (2010). IEEE

[48] Isuwa, S., Dey, S., Ortega, A.P., Singh, A.K., Al-Hashimi, B.M., Merrett, G.V.: Quarem: Maximising qoe through adaptive resource management in mobile mpsoc platforms. ACM Transactions on Embedded Computing Systems (TECS) (2022)

[49] Nuka, S.: Investigation of Multi-dimensional QoE Models with the Impact of Resolution Vs Stalls for Video Streaming (2018)

[50] Li, J., Chen, Y., Zhao, X., Huang, J.: An improved dqn path planning algorithm.

The Journal of Supercomputing **78**(1), 616–639 (2022)

Table 1: List of key notations and parameters

Variables	Description
	Virtual network attributes
VN	Virtual network (a set of VNFs of different types interconnected via virtual links)
VNF	The set of VNF instances in **VN**
vnf_i	The VNF instance of VNF type i, $vnf_i \in$ **VNF** ($i \in \{1...N\}$)
N	The number of VNF types in **VN**
VL	The set of virtual links in **VN**
$VLink(i,l)$	The virtual link interconnecting vnf_i to vnf_l, $VLink(i,l) \in$ **VL** ($i,l \in \{1...N\}$ *where* $i \neq l$)
	PSN attributes
NFVI	NFV infrastructure (a set of PSN nodes and PSN links)
$nfvi_j$	The j^{th} PSN node, $nfvi_j \in$ **NFVI** ($j \in \{1...M\}$)
M	The number of PSN nodes in **NFVI**
$D_{nfvi_j}^{vnf_i^{r_n}}$	The average processing delay for mapping $vnf_i^{r_n}$ in $nfvi_j$
C_{nfvi_j}	The residual available capacity of $nfvi_j$ (in CPU cores)
$Link(j,k)$	The PSN link connecting $nfvi_j$ to $nfvi_k$ ($j,k \in \{1...M\}$ *with* $j \neq k$)
$D_{j,k}$	The communication latency on PSN link $Link(j,k)$
$Bw_{j,k}$	The average PSN link bandwidth of $Link(j,k)$
	SFC request attributes
R	The set of SFC incoming requests
r_n	The n^{th} SFC request $r_n \in$ **R** ($n \in \{1...SFC\}$), r_n=$\{In_{r_n}, En_{r_n}, VNF_{r_n}, VLINK_{r_n}, Bw_{r_n}, D_{r_n}\}$
SFC	The number of SFC requests in **R**
In_{r_n}	The ingress node of r_n
En_{r_n}	The egress node of r_n
VNF_{r_n}	The set of ordered VNF instances specified in r_n
$vnf_i^{r_n}$	The VNF instance of VNF type i instantiated by r_n, $vnf_i^{r_n} \in VNF_{r_n} (i \in \{1...N^{r_n}\})$
$VLINK_{r_n}$	The set of virtual links specified in r_n
$VLink(i,l)^{r_n}$	The virtual link joining $vnf_i^{r_n}$ to $vnf_l^{r_n}$, $VLink(i,l)^{r_n} \in VLINK_{r_n} (i,l \in \{1...N^{r_n}\})$
N^{r_n}	The number of VNF types of r_n
$L_{vnf_i^{r_n}}$	The maximum number of VNF instances actually deployed for each type of $vnf_i^{r_n}$ simultaneously
$vnf_{ij}^{r_n}$	The j^{th} VNF instance of VNF type i ($j \in \{1...L_{vnf_i^{r_n}}\}$)
$C_{vnf_i^{r_n}}$	The requested capacity by $vnf_i^{r_n}$ (in CPU cores)
Bw_{r_n}	The r_n end-to-end bandwidth
D_{r_n}	The r_n end-to-end network delay
Dl_{r_n}	The sum of delays incurred by each link involved in r_n
Dn_{r_n}	The sum of delays incurred by each VNF instantiated by r_n
	SFC instance attributes
SFCI	The set of all SFC instances
$sfci_c$	The c^{th}SFC instance, $sfci_c \in$ **SFCI** ($c \in \{1...C\}$)
C	The number of SFC instances in **SFCI**
QoE_{sfci_c}	The QoE of $sfci_c$
QoS_{sfci_c}	The QoS of $sfci_c$
	Decision boolean variables
$x_{sfci_c,nfvi_j}^{r_n,vnf_i^{r_n}}$	$\begin{cases} 1, & \text{if } vnf_i \text{ specified in } r_n \text{ is located in } nfvi_j \\ 0, & \text{otherwise} \end{cases}$
$y_{sfci_c,j,k}^{r_n,i,l}$	$\begin{cases} 1, & \text{if } VLink(i,l)^{r_n} \text{ is embedded on } Link(j,k) \\ 0, & \text{otherwise} \end{cases}$

Table 2: Problem formulation of Online QoS/QoE-driven SFC Orchestration

Objective function:

$$max_{c \in C} R_{QoE-QoS}, \forall r_n \in \mathbf{R} \tag{6}$$

Constraints:

VNF License:

$$1 \leq \sum_{j=1}^{M_i} x_{sfci_c,nfvi_j}^{r_n,vnf_i^{r_n}} \leq L_{vnf_i^{r_n}}, vnf_i^{r_n} \in VNF_{r_n} \tag{7}$$

VNF Placement:

$$\sum_{i=1}^{N} \sum_{j=1}^{M_i} x_{sfci_c,nfvi_j}^{r_n,vnf_i^{r_n}} \geq N^{r_n} \tag{8}$$

PSN Node Capacity:

$$x_{sfci_c,nfvi_j}^{r_n,vnf_i^{r_n}} . C_{vnf_i^{r_n}} \leq C_{nfvi_j}, vnf_i^{r_n} \in VNF_{r_n},\ nfvi_j \in NFVI \tag{9}$$

PSN Link Capacity:

$$y_{sfci_c,j,k}^{r_n,i,l} . Bw_{j,k} \geq Bw_{r_n}, VLink(i,l)^{r_n} \in VLINK_{r_n},\ Link(j,k) \in NFVI \tag{10}$$

Delay:

$$Dn_{r_n} = \sum_{i=1}^{N} \sum_{j=1}^{M_i} x_{sfci_c,nfvi_j}^{r_n,vnf_i^{r_n}} . D_{nfvi_j}^{vnf_i^{r_n}} \tag{11}$$

$$Dl_{r_n} = \sum_{i=1}^{N-1} \sum_{l=i+1}^{N} \sum_{j=1}^{M_i-1} \sum_{k=j+1}^{M_i} y_{sfci_c,j,k}^{r_n,i,l} . D_{j,k} \tag{12}$$

$$Dn_{r_n} + Dl_{r_n} \leq D_{r_n} \tag{13}$$

Table 3: Default Double DQN Hyper-parameters

Hyper-parameter	**Value**
Number of Frames	State Number
Discount Factor γ	$0,95$
Batch Size (BS)	64
Loss Function	L_2
Optimizer	Adam
Learning Rate (LR)	$2,5 * 10^{-4}$
Target Q Update Frequency	100 SFC requests
Replay Buffer	Prioritized
Replay Buffer Size (RS)	$20K$
Initial ϵ	$0,8$
ϵ decay	$0,99$