IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 7, 2019, accepted November 12, 2019, date of publication November 14, 2019, date of current version December 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2953498

Reinforcement Learning for Service Function Chain Reconfiguration in NFV-SDN Metro-Core Optical Networks

SEBASTIAN TROIA, RODOLFO ALVIZU, AND GUIDO MAIER
Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, 20133 Milan, Italy

Corresponding author: Sebastian Troia (sebastian.troia@polimi.it)

The work leading to these results has been supported by the European Community under grant agreement no. 761727 Metro-Haul Project.

ABSTRACT With the advent of 5G technology, we are witnessing the development of increasingly bandwidth-hungry network applications, such as enhanced mobile broadband, massive machine-type communications and ultra-reliable low-latency communications. Software Defined Networking (SDN), Network Function Virtualization (NFV) and Network Slicing (NS) are gaining momentum not only in research but also in IT industry representing the drivers of 5G. NS is an approach to network operations allowing the partition of a physical topology into multiple independent virtual networks, called network slices (or slices). Within a single slice, a set of Service Function Chains (SFCs) is defined and the network resources, e.g. bandwidth, can be provisioned dynamically on demand according to specific Quality of Service (QoS) and Service Level Agreement (SLA) requirements. Traditional schemes for network resources provisioning based on static policies may lead to poor resource utilization and suffer from scalability issues. In this article, we investigate the application of Reinforcement Learning (RL) for performing dynamic SFC resources allocation in NFV-SDN enabled metro-core optical networks. RL allows to build a self-learning system able to solve highly complex problems by employing RL agents to learn policies from an evolving network environment. In particular, we build an RL system able to optimize the resources allocation of SFCs in a multi-layer network (packet over flexi-grid optical layer). The RL agent decides *if* and *when* to reconfigure the SFCs, given state of the network and historical traffic traces. Numerical simulations show significant advantages of our RL-based optimization over rule-based optimization design.

INDEX TERMS Software-defined networking, network function virtualization, network slicing, virtual service chain, reinforcement learning, deep learning, mixed integer linear programming.

I. INTRODUCTION

Traditional network architectures are ill-suited to meet the requirements of today's enterprises, carriers, and end users. In particular, with the development of new technologies such as 5G, the need of more flexible networks and better tools to configure them is urgent. SDN, NFV, and NS are three of these important tools capable of increasing the flexibility of a network and reduce its costs.

After more than ten years of research and development, SDN is finally going mainstream. As Dr. G. Parulkar said: Thanks to SDN, disaggregation and open networking, we have the opportunity to reinvent the most important infrastructure of the society.[1] SDN is a network paradigm that decouples network device hardware from the control decisions. Therefore, the network intelligence is logically centralized in software-based controllers (the control plane), and network devices become simple packet forwarding devices (the data plane) that can be programmed via open interfaces [1].

NFV is the paradigm of moving Network Functions (NFs) from dedicated hardware appliances to software-based applications running on commercial off-the-shelf equipment [2]. These *Virtual* Network Functions (VNFs) provide many benefits to the telecommunications industry such as openness of platforms, scalability and flexibility, shorter development

The associate editor coordinating the review of this manuscript and approving it for publication was Gaurav Somani.

[1] Guru Parulkar, Executive Director, Open Networking Foundation, at the Keynote Panel: SDN@10, Open Networking Summit, March, 2018

cycles and reduced Capital Expenditure (CapEx) and Operating Expenditure (OpEx) [3]. This new technology made the implementation and management of SFC possible and easy. An SFC is a set of multiple VNFs of different types, crossed by a traffic flow in a specific order, so to provide a specific service.

NS is an approach to network operations based on the concept of network abstraction and virtualization, which provides programmability, flexibility, and modularity to traditional networks. It is a well known concept since a long time, but recently revamped for the implementation of 5G standard for mobile communications. It can be applied both to wireless and fixed network architectures [4]. The physical topology is divided into multiple independent virtual networks (VN) called network slices (or slices). Within a single slice, the whole set of network resources can be allocated dynamically on demand according to specific QoS and SLA requirements, but also keeping track of other parameters that customers are not concerned with, but infrastructure providers are, such as power consumption of network devices.

While on the one hand, SDN, NFV and NS bring many benefits, on the other there are some critical aspects to consider: first of all, the allocation of network resources. Provisioning the resources to a network slice, such as the one related to an SFC, means managing a multi-layer network composed by: service, IP/MPLS and optical transport layer. Considering Fig. 1, given a request to provision an SFC, we must consider the availability of resources on the three network layers, such as: 1) computing resources in the service layer, e.g. processing and memory; 2) bandwidth in the IP/MPLS layer; 3) and finally optical resources in the transport layer, e.g. fibers and wavelengths. This task can be performed either statically or dynamically. In static slicing, a fixed set of resources is allocated to an SFC for its entire lifetime. This implies that the allocated resources should support the peak service requirements. In dynamic slicing, the amount of resources allocated to an SFC varies according to the actual service needs [5].

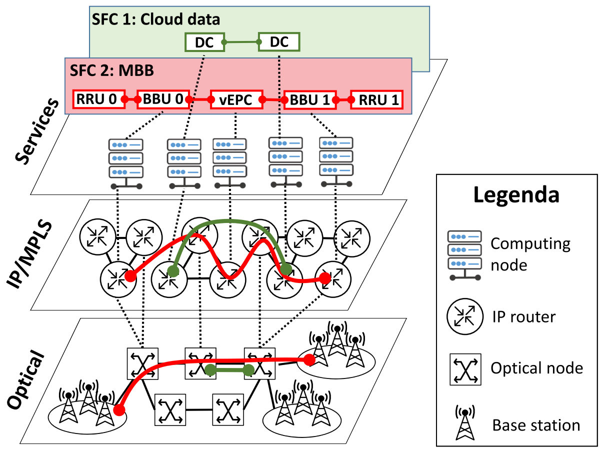

FIGURE 1. Multi-layer network model. We show the general framework of the multi-layer network considered in this work, along with an example of how two SFCs are mapped into the service, IP/MPLS and optical network layer.

In this work, we introduce a novel method based on Reinforcement Learning (RL) for performing dynamic SFC resources allocation in NFV-SDN enabled metro-core optical networks. As shown in Fig. 1, an SFC is composed of an ordered sequence of VNFs. Therefore, given a set of SFC provisioning requests and their QoS requirements, the proposed method finds the optimal positioning of the VNFs in the service layer and allocates the network resources of the IP/MPLS and optical layer, respectively. Furthermore, in order to meet the SFC traffic variation over time, it dynamically predicts *if* and *when* to reconfigure the SFCs that no longer meet the required QoS criteria. Each reconfiguration implies an availability penalty on the service as it requires a temporary service interruption to allow for VNF migration and/or network resource re-allocation.

Thus, the mechanism presented here can be seen as a way of optimizing the spending of reconfiguration tokens over the span of a day. The proposed RL-based algorithm aims to distribute the spending of these tokens over the day in order to minimize the blocking probability of traffic requests finding the best action in the trade-off between availability and performance.

We have chosen as a use case the metro-core network architecture proposed by the Metro-Haul European project [6], which is based on NFV-SDN network elements.

The paper is organized as follow: Section II presents related works. Section III provides a primer on reinforcement learning. Section IV introduces the complete reinforcement learning environment for NFV-SDN networks. Section V describes the Metro-Haul network topology and the dataset used for this work. Numerical results are reported in Section VI. Section VII presents a discussion on a real-world deployment of the proposed methodology. Finally, conclusions are presented in Section VIII.

II. RELATED WORK

The problem of VNF placement for NS and traffic routing for SFC provisioning has been subject to deep investigation in the last years, especially in static settings, such as under a static traffic assumption. Reference [5] proposed an optimization model for the minimization of wavelength-resource usage during a defined period of time. For each experiment they randomly placed the VNFs in the network and inject traffic according to a day/night profile. Every time this traffic profile changes (12 hours), the Mixed Integer Linear Programming (MILP) is rerun and the network is reconfigured. In [7] the authors developed algorithms for VNF placement and traffic routing with protection on a single node and link. They performed end-to-end protection and compared the results with an unprotected case. Their model aimed at the minimization of the number of active nodes where a VNF is placed, which is an indicator of CapEx and OpEx costs. In [8] the authors proposed a context-free formalization of SFC concept and presented a Mixed Integer Quadratically Constrained

Model (MIQCP) for the optimal placement of VNFs and SFCs. They dealt for the first time with a formal modelling of service chaining problem, which finds the placement of VNFs and chains them together considering resource limitation of the network. Authors in [9] designed an algorithm to jointly place VNFs and route traffic between them. They organized the problem in smaller sub-problems (one per VNF in the requested chain) that are solved successively and aggregated to compose the overall chain placement. Authors in [10] dealt with the problem of SFC mapping with multiple SFC instances to minimize network resource consumption. They proposed an ILP, a column-generation-based ILP (CG-ILP), and a two-phase column-generation-based model (2PhMod) to solve this problem. Also some heuristic algorithms for VNF placement have been already proposed, as in [11].

In [12], the authors proposed an algorithm for dynamic VNF placement for SFC provisioning in a metro network. At each time instant, a certain number of users request a specific SFC, and based on the current condition of the network, VNFs are placed in order to minimize the blocking probability. The algorithm performed VNF placement in such a way that the bandwidth requirements of links, computational requirements of the NFV-nodes and latency requirements of requested SFCs are satisfied, and wavelength continuity at each node is enforced. Authors of the work in [13] provided an on-line algorithm for VNF scaling to dynamically provision network services in a datacenter network. In [14] a Mixed Integer Programming formulation and a heuristic algorithm are provided to dynamically provision SFC, again in a datacenter network, where an appropriate resource management is done based on number of users requesting SFCs. The authors of [15] considered dynamic SFC provisioning for two types of users in the network, new users and existing users that can change location in the network and change their requested SFC. So, they proposed at first an ILP model for service provisioning with the objective to maximize the profit of the service provider; then, to reduce time complexity, they provided a more scalable model based on column generation [16].

With the advances of communication network technologies, the network environment becomes more complicated and dynamic, which makes it hard to be managed and controlled. RL methods provides an efficient way to design software models that can learn and adapt to the dynamic network resource allocation from past observations. The authors in [17] proposed a deep RL algorithm for QoS and Quality of Experience (QoE) aware SFC in NFV-enabled 5G systems. Typical QoS metrics are bandwidth, delay and throughput. The evaluation of QoE normally involves the end-user's participation in rating the service based on direct user perception. The authors quantified QoE by measurable QoS metrics without end-user involvements, according to the Weber-Fechner Law (WFL) [18] and exponential interdependency of QoE and QoS hypothesis [19]. These two principles actually define a non-linear relationship between QoE and QoS. The system state represents the network environment including network topology, QoS/QoE status of the VNF instances, and the QoS requirements of the SFC request. The Deep Q-Learning (DQL) agent selects certain direct successive VNF instance as an action. The reward is a composite function of the QoE gain, the QoS constraint penalty, and the OpEx penalty. A DQL based on Convolutional Neural Networks (CNNs) is implemented to approximate the action-value function. The authors in [20] reviewed the application of a DQL framework in two typical resource management scenarios using NS. For radio resource slicing, the authors simulate a scenario containing one single Base Stations (BS) with different types of services. The reward can be defined as a weighted sum of spectrum efficiency and QoE. For priority-based core NS, the authors simulated a scenario with 3 SFCs demanding different computational resources and waiting time. The reward is the sum of waiting time in different SFCs. Simulation results in both scenarios showed that the DQL framework could exploit more implicit relationships between user activities and resource allocation in resource constrained scenarios, and enhance the effectiveness and agility for NS.

A. PAPER CONTRIBUTION

In our previous work [21], we introduced our proposed machine-learning-based methodology to optimize decisions of mobile-metro-core network orchestration systems. The purpose of this paper is to realize an RL-based system capable of optimizing the dynamic allocation of resources to SFCs in a NFV-SDN enabled metro-core optical network.

Multi-layer network optimization is known to be a difficult task, especially if we are in conditions where network traffic changes over time. The intelligence lies in achieving an optimization task that takes into account the past history of network configurations, instead of implementing thus optimization and resource allocation only based on the current status. To achieve this goal, we have implemented a reinforcement learning system able to learn *if* and *when* to perform reconfigurations of SFCs.

Reinforcement learning is an approach to machine-learning that trains algorithms using a system of positive and negative rewards. An RL algorithm, or *agent*, learns how to perform a task by interacting with its *environment*. The interaction is defined in terms of specific *actions*, *observations* and *rewards*. As we can see from Fig.2, an agent performs an action at time t based on the reward and state (or observations) obtained by the environment. The action performed by the agent will produce another state of the environment and a reward at a time $t + 1$ and so on.

We have addressed the problem as follows:

1) We have built a multi-layer optimization model formulation based on MILP that, given a set of SFC requests, finds the optimal VNF placement and Routing and Wavelength Assignment (RWA). The objective function of the MILP aims at maximizing the number of successfully routed SFCs, while minimizing: reconfiguration penalty, blocking probability and power

consumption of network elements (see Appendix). We encoded the MILP inside an RL system making it as an environment.

2) Together with the MILP environment, we developed the reward function and the agent that decides: 1) which SFC to reconfigure and how to reconfigure it (i.e. migration, scaling-up or down, etc.) and 2) when to perform the reconfiguration. Each SFC has a maximum number of reconfigurations (or tokens) that can be performed, which depends on the QoS requirements. Therefore, the agent learns how best to pace the SFC reconfiguration to minimize the blocking of traffic requests.
3) We tested the proposed system with real traffic traces that represents the mobile traffic of Milan urban area, in Italy. Moreover, we have used as reference the metro-core network architecture proposed by the Metro-Haul European project [6], which is based on NFV-SDN network elements.

In the next section, we will introduce the background knowledge on reinforcement learning.

III. PRIMER ON REINFORCEMENT LEARNING

Reinforcement learning, in the context of artificial intelligence, is an approach to machine-learning that trains algorithms using a system of positive and negative rewards. Contrary to what happens with the supervised and unsupervised machine-learning algorithms, in RL we do not have training, validation and test set of data. A reinforcement learning algorithm, or *agent*, learns how to perform a task by interacting with its *environment*. The interaction is defined in terms of specific *actions*, *observations* and *rewards*. As we can see from Fig.2, an agent performs an action at time t based on the reward and state (or observations) obtained by the environment. The action performed by the agent will produce another state of the environment and a reward at a time $t + 1$ and so on.

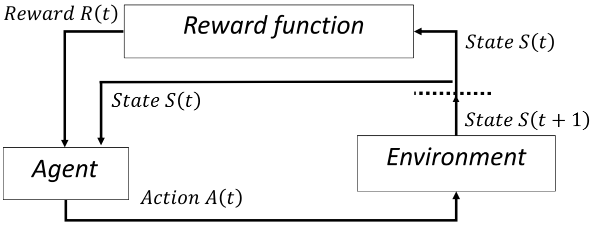

FIGURE 2. Basic reinforcement learning system with an agent, a reward function and the environment.

For example, an agent could be a self-driving car or a program playing chess, and it receives a positive reward depending on how it performs, such as driving to destination safely or winning a game. Conversely, the agent receives a negative reward for performing incorrectly, such as going off the road or being checkmated. The goal of these two examples is to maximize the reward in order to make no accidents and to win the chess game respectively. Considering Fig.2:

1) The *Agent* is the intelligence of the whole system. It implements an algorithm able to learn how to perform *actions* at each time-step t, maximizing the received *reward*.
2) The *Reward function* represents the goodness of the action taken by the agent. It is high if the agent has made a correct action, low otherwise.
3) The *Environment* represents the problem to solve. Consequently to the action performed by the agent, it provides a new *observation* that describes the state of the environment.

The advantages of this approach are manifold. Firstly, it allows to solve complex tasks from unprocessed input data. Secondly, it allows an Artificial Intelligence (AI) program to learn without a programmer spelling out how an agent should perform the task. Both these aspects are fundamental in the development of optimization algorithms for multi-layer networks as they work with a large amount of heterogeneous data that evolves over time. In Section IV, we will see how to apply this approach to optimize the resource allocation of a multi-layer telecommunication network.

IV. REINFORCEMENT LEARNING for Dynamic Multi-layer RESOURCE ALLOCATION

The objective of this work is to develop a learning system based on RL and MILP in order to dynamically optimize SFCs placement and the resources allocation in a multi-layer network. As described in Section III, to build an RL system we need to define several components such as: the environment (including the set of observations), the reward function and the agent.

A. NETWORK ENVIRONMENT

The environment, namely network environment, consists of a multi-layer model. Its formulation is based on MILP that, given a set of SFC requests, finds the optimal VNF placement and RWA. The objective function of the MILP aims to maximize the number of successfully routed SFCs, minimizing: reconfiguration penalty, blocking probability and power consumption of network elements (see Appendix for more details). As shown in Fig. 1, it is composed by three layers:

1) *Optical layer*: represents the physical topology, made by optical devices connected by fiber optics, and handles the RWA of the light-paths requested by the IP/MPLS layer.
2) *IP/MPLS layer*: handles the creation of virtual tunnels (lightpaths) between the various virtual functions of the chains mapped from the service layer.
3) *Service layer*: collects the sets of slices and relative associated chains. The VNFs are the nodes of the SFC in this layer. We assume that each node of the service layer is mapped into a node of the IP/MPLS layer.

At each time step t, i.e. hour, the network environment exposes its state $S(t)$ made by 3 *observations*:

1) $O_1(t)$: total number of reconfigurations of each SFC, at time t.
2) $O_2(t)$: total number of blocked requests of each SFC, at time t.
3) $O_3(t)$: traffic volume request in *Gbps* of each SFC, at time t.

As shown in Fig. 3, the agent, namely Network Agent, is a Deep Neural Network (DNN) that takes as input the state of the environment $S(t)$ and the result of the reward function $R(t)$. As output, the agent decides if one or more SFCs need to be reconfigured. This decision is performed by means of an action $A(t)$ executed on the environment.[2]

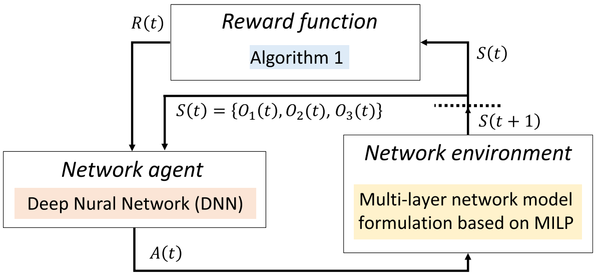

FIGURE 3. Schema of the proposed RL system. The agent is performed by a Deep Neural Network, the reward function by Algorithm 1 and the environment by the multi-layer network model formulation based on MILP.

The reward function represents the effect of the action performed by the agent. It depends on the number of blocked service-chain requests; $R(t)$ is positive if the agent managed to reconfigure the network in order not to have blocked requests, otherwise it is negative.

In the next sections we will show how we have built the reward function and the network agent.

B. REWARD FUNCTION

The role of the reward function is to determine the behavior of the entire learning system. The output of this function is a real number that can be either negative or positive. We shape rewards that get gradual feedback and let the agent know if it is getting better or worst (in Section IV-C we will show how this feedback is interpreted by the agent).

In particular, at each time step (i.e. hour) and for each SFC, the environment provides the total number of reconfigurations and the total number of blocked requests through the observation $O_1(t)$ and $O_2(t)$, respectively. Each service chain has a maximum number of reconfigurations that can be performed in each day, which depends on the policy implemented by the service/network operator. The maximum number of reconfigurations is set by the availability requirement (or class) negotiated in the SLA between operator and customer.

In our work, we have assumed that each slice has a given availability class. Therefore the SFCs of that slice can be reconfigured only a certain number of times a day. We have defined the $Threshold_r$ as the maximum number of reconfigurations per day for the service chain r. The reward function is derived with the aim to have a reward that gradually increases negatively if the agent is allowing the reconfiguration of the service chain r more than the $Threshold_r$, as well as positively if it is under the $Threshold_r$. The Algorithm 1 shows the pseudo-algorithm of the reward function.

Algorithm 1 Reward Function for SFC Provisioning in Multi-Layer Telecommunication Networks

```
The algorithm takes as input: the total number of
blocked requests $B_{count}(t) = \sum_r^{N_{SFC}} B^r(t)$ and the total
number of reconfigurations performed $N^r_{count}(t)$ at each
hour $t \in (1, 24)$ of the day and for each service chain
$r \in N_{SFC}$. It provides a reward $R$ at each time step $t$;
while not at end of the day ($t \leq 24$) and for each service
chain $r \in N_{SFC}$ do
    if $N^r_{count}(t) \geq Threshold_r$ then
        $R(t) = R(t-1) + B_{count}(t) \times (-\eta)$;
    else
        $R(t) = R(t-1) + \eta$;
    end
end
```

The gradual feedback was obtained thanks to the use of an arbitrary parameter η which can be dynamic every day based on the assigned value of $Threshold_r$. In our work, we implemented a single reward function strategy in which η is set to a constant value of 10. The investigation of the dynamic case is left for future works.

C. NETWORK AGENT

The agent, or network agent, consists of a DNN that takes as input the observations obtained from the environment and returns the decision whether or not a service chain should be reconfigured. As we said in Section III, in RL algorithms there are no historical data-sets on which to perform a learning; thus, the learning process is carried out with trials, in which at the beginning the DNN will make random decisions, and later, thanks to the reward function, it will be trained in order to minimize (or maximize) the objective function of the multi-layer network environment.

A DNN is based on a collection of connected units, or nodes, called artificial neurons, which mimic the neurons in a biological brain [22]. Each connection, like the synapses, can transmit a signal from one artificial neuron to another, which are used to estimate or approximate functions that can depend on a large number of generally unknown inputs [22]. Specifically, we model a DNN with the aim to decide in each time interval whether the SFCs already deployed in the network must be reconfigured or not, so to optimize the timing of a limited number of reconfigurations over the day.

In order to describe a DNN we need to specify the *Architecture* and the *Learning rule*.

[2] In other words, the action modifies an input parameter of the MILP. The input parameter is I^r, and it is defined in Table 3 (Appendix).

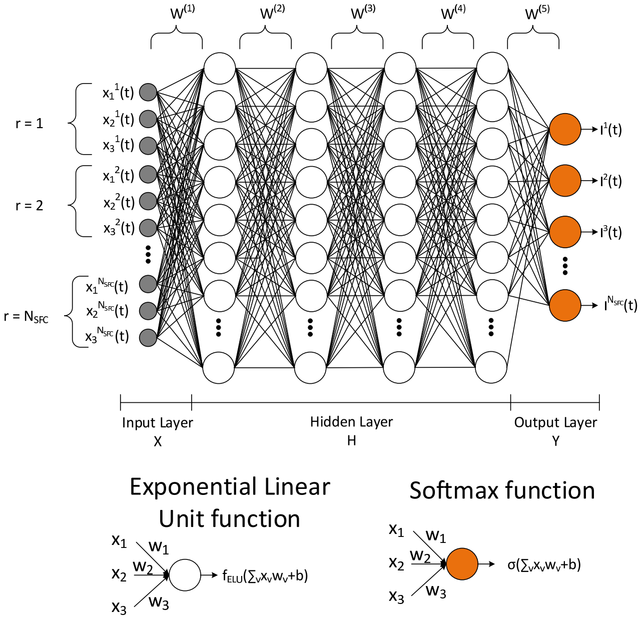

FIGURE 4. Architecture of the deep neural network.

1) ARCHITECTURE

The architecture specifies which variables, or hyper-parameters, are involved in the network and their topological relationships. For example, the hyper-parameters involved in a neural network might be the number of layers, neurons and the weights of the connections between the neurons. In this paper, we have adopted the greedy layer-wise training to determine the hyper-parameters of the DNN architecture [23], i.e., the number of layers and number of neurons in each layer are tested by means of trials. Essentially, we tried different neural networks starting from an architecture composed by one hidden layer, with a minimum of two nodes per layer, until five hidden layers with at most 100 neurons per layer. This procedure showed that, the DNN with the best performance in terms of reward function is the one with four hidden layers with 100 nodes per layer, see Fig.4.

- The input layer is composed by $\bar{X}$ number of input nodes, where $X \in \mathbb{R}^{[1\times\bar{X}]}$ is the input vector and each element represents the observations $[O_1(t), O_2(t), O_3(t)]$. $O_f(t)$ is represented in the form $x^r_f(t)$, where $r \in [1, N_{SFC}]$ is the ID of the service chain, and $f \in [1, 3]$ represents one of the 3 observations:
  - $x^r_1(t)$: equal to 1 if there are reconfigurations of SFC r at time t, 0 otherwise
  - $x^r_2(t)$: equal to 0 if SFC r is blocked at time t, 1 otherwise
  - $x^r_3(t)$: traffic volume request in [Gbps] between network functions of SFC r at time t
- The hidden section of the network is composed by $\bar{H}$ number of hidden layers, with $\bar{H}_s$ number of nodes in the layer s. The *Exponential Linear Unit (ELU)* is the activation function and it is denoted by the Eq.1:

$$f_{ELU}(z) = \begin{cases} \alpha\left(e^{z}-1\right), & z < 0 \\ z, & z \geq 0 \end{cases} \tag{1}$$

  where z is the sum of the input weights multiplied by the output value of the node from the previous layer. The hidden layers are composed by the following set of weights:
  - $W^{(1)} \in \mathbb{R}^{[\bar{X}\times\bar{H}_1]}$, each element represents the weights between the input layer and the first hidden layer
  - $W^{(s)} \in \mathbb{R}^{[\bar{H}_{(s-1)}\times\bar{H}_{(s)}]}$ with $s < \bar{H}$, each element represents the weights between the hidden nodes of the intermediate layers
  - $W^{(\bar{H})} \in \mathbb{R}^{[\bar{H}_{(\bar{H}-1)}\times\bar{Y}]}$, each element represents the weights between the last hidden layer and the output layer
- The output layer is composed by $\bar{Y}$ number of nodes, where $Y \in \mathbb{R}^{[\bar{Y}\times 1]}$ is the output vector and each element is represented in the form of y^r, where r is the ID of the service chain. Since we are building a neural network classifier, the activation function of the last layer is the *softmax*, see Eq.2. It is used in various multi-class classification methods, such as multinomial logistic regression [24]. In this case, the output nodes denoted by $\sigma(z)_r$ represent the action that changes the state of the multi-layer network environment[3]

$$\sigma(z)_r = \frac{e^{z_r}}{\sum_{k=1}^{N_{SFC}} e^{z_k}} \tag{2}$$

  where z is the sum of the input weights multiplied by the output value of the node from the previous layer, and r is the ID of the service chain.

2) LEARNING RULE

The learning rule specifies the way in which the neural network's weights $W^{(i)}$ change over time. Typically a learning rule is a cost function that measures how well the network with weights solves the task. The softmax function used in the final layer of a neural network-based classifier is trained exploiting the cross-entropy C as cost function. There are two properties that make the cross-entropy function suitable for training a classification system. First, it is a positive function; second, if the neuron's output is close to the desired output for all training inputs X, then the cross-entropy will be close to zero [25].

As we said in Section III, in RL algorithms we do not have training and test data-sets. In this case, the algorithm learns from the actions performed on the environment building step by step its own historical data-set. Differently from the training procedure applied in the case of classical neural networks, we used the policy gradient descend [26] in which the cost

[3] The action will modify the input parameter I^r defined in Table 3 (Appendix), and it will be $I^r = \sigma(z)_r$.

function C is multiplied by a *reward*. The multiplication with the reward eventually move the cost function to be very high if the action performed was wrong and be very low if it was correct. Differently from classic gradient descend algorithms, we define a policy to train the DNN. In other words, how do we differentiate the good and wrong action? If the action performed by the agent at time t has minimized the cost function, it does not mean that giving a high reward is the right choice. In this case we apply the *discounted reward*, that is, the reward at time t is discounted by a parameter $D \in [0.95, 0.99]$ that multiplies the past rewards [27], see Eq.3.

$$R(t) = \sum_{j=0}^{t-1} R(j) \times D^j \tag{3}$$

Thanks to this reward procedure, the DNN will learn how to reconfigure the service chains in future time steps in order to maximize the long term reward and therefore the objective function of the MILP.

V. METRO-HAUL USE CASE

In this paper, we took as reference the metro-core network architecture proposed by the Metro-Haul European project [6]. The network comprises metro nodes interconnected by a high capacity dynamic and flexible optical network. The nodes host both IT and TLC equipment, following the Multi-access Edge Computing (MEC) model defined by ETSI, so that they can support the instantiation of VNFs. They are named AMEN (Access Metro Edge Network Nodes) and MCEN (Metro Core Edge Nodes), according to their location in the topology. Transmission distances range between 50km and 200km. To meet the Metro-Haul objectives, both storage and computing resources will be necessary both in the AMEN and the MCEN in order to host VNFs and enable NS. The Metro-Haul network is designed to support several slices according to the 5G-MEC model.

A. NETWORK TOPOLOGY

In this paper, we assumed a metro network having the feature defined by the Metro-Haul project. Its topology has two rings and a mesh core network, see Fig. 5.

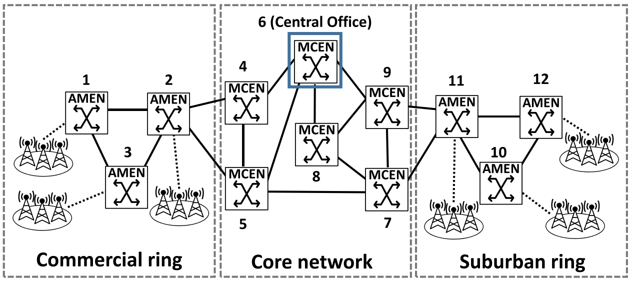

FIGURE 5. Network topology based on Metro-Haul project.

It is composed by six MCENs. One of the MCEN contains a central office (CO) with an internet exchange point and a data center (DC). Each link of the core mesh is 40 km long. The other two rings represent the metropolitan network segment. The AMENs are the nodes from which the traffic is injected in the network. The links in the two rings have a length of 10 km. Each physical link contains two fibers, each one with 20 wavelengths with a capacity of $10 Gbps/wavelength$.

B. NETWORK SLICES

Following the Metro-Haul guidelines, we introduced two kinds of network slices. For each slice, we defined the type of traffic it uses, its maximum latency and the possible chains associated to it. In Table 1 we listed the slices to be provisioned in the proposed topology with their relative characteristics.

TABLE 1. List of network slices along with their SFCs.

Mobile broadband (MBB)	
Latency	100 ms
SFCs	1) RRU - BBU - vEPC 2) vEPC - BBU - RRU
Traffic volume	1) 10x between RRU and BBU 2) x between BBU and vEPC
Availability	1) RRU - BBU (*and viceversa*) = 99.95% 2) BBU - vEPC (*and viceversa*) = 99.9%
Threshold	1) RRU - BBU (*and viceversa*) = 5 2) BBU - vEPC (*and viceversa*) = 13
Cloud data	
Latency	50 ms
SFC	DC - DC
Traffic volume	50 to 500 Gbps
Availability	99.95%
Threshold	5

The Mobile Broad Band (MBB) slice handles all the mobile cellular internet traffic in the network, except for the video traffic. In the radio access network (RAN), there are two main deployment options. The traditional packet-based backhaul deployment where the baseband processing is performed at the antenna site, and the centralized RAN (C-RAN) deployment where baseband processing units (BBUs) are centralized in a BBU pool, and digitized radio signals are transmitted to remote radio units (RRUs) over a fronthaul network [28]. The virtual Evolved Packet Core (vEPC) functions can be specific to a slice, and placed at different locations depending on the requirements defined by 3GPP specifications. In this work, we followed the C-RAN deployment where the MBB slice is composed by: RRU (Remote Radio Unit), BBU (Base Band Unit) and vEPC (virtual Evolved Packet Core) [29]. The *Cloud data* slice tries to emulate the traffic flows among distributed DCs. These data centers host applications for social networking, e-commerce, and video hosting [30].

In order to assign a *Threshold* for each SFC, or the maximum number of reconfigurations that a slice can undergo, we assigned an availability class to each of them. Availability is usually expressed as a percentage of up-time in a given year. The availability is expressed in terms of *Nines*: for example, network services that are delivered without interruptions 99.95% of the time means that they can remain 43.2 seconds in downtime per day.

Authors in [31] made an evaluation study on an NFV-SDN orchestration system devised to select and allocate virtual resources in distributed DCs connected through a multi-layer (packet over flexi-grid optical) network. The experimental performance evaluation is carried out using a real testbed encompassing a cloud/network infrastructure. Considering existing flexi-grid optical flows, they demonstrated that the average setup time of an SFC with at most 5 VNFs, is between 1 and 1.5 seconds. Furthermore, the authors in [32] showed that the total downtime during the real-time migration of a heavy loaded VNF is 0.9 seconds. Considering the worst case scenario, to reconfigure an entire SFC, such as the MBB slice, would take at most 8 seconds. Given that, we considered this value to derive the maximum number of reconfigurations for each SFC. In particular, for the availability class of 99.95% with 43.2 seconds of downtime per day, we considered $Threshold = \lfloor \frac{43.2}{8} \rfloor$ as maximum reconfiguration number. In Table 1 we computed the total reconfiguration thresholds for each considered SFC.

C. NETWORK TRAFFIC

The data used in this work is taken form our previous work [33]. It refers to the mobile traffic data of Milan urban area, measured during November and December 2013 [34]. In [33] we have implemented a clustering method in order to highlight the tidal traffic effect. In Fig. 6 we show the typical patterns of three types of traffic: cloud, commercial and suburban.

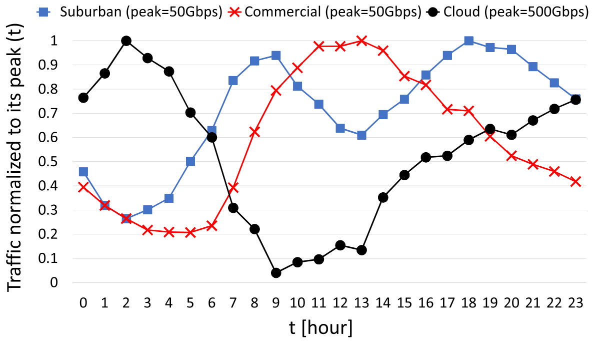

FIGURE 6. Tidal traffic patterns.

In this use case, we assumed to find the best resource allocation in order to deploy 9 SFCs: 6 *MBB* and 3 *Cloud data*. Cloud traffic is characterized by a high throughput during night hours and a lower one during day hours. This curve is normalized to its peak, which is 500*Gbps*. The mobile traffic, instead, handles the mobile traffic generated by citizens in the city of Milan. In particular, there are two kind of MBB traffic. The *commercial* one, generated by the nodes on the right metro ring, presents peaks during the afternoon hours. Then we have also the *suburban* traffic that is injected in the network by the nodes in the left metro ring. This traffic presents peaks during the morning and night hours. For both *commercial* and *suburban* traffic their peak is 50*Gbps*.

VI. SIMULATIONS

The simulation is composed of discrete events. At each event, the following steps are performed:

1) The agent receives the inputs described in Section IV-C.1. Inputs are processed by the DNN that will give as output a scheme of reconfigurations for the SFCs.
2) The environment is run, i.e. the MILP derives the best network resources assignment. The output of this operation represents the state of the environment and it will be taken as input by the reward function.
3) The reward function evaluates the effect of the action. Based on Algorithm 1, it returns a reward to the agent, i.e. it modifies the weights of the DNN, positive if there are no blocked requests, negative otherwise.

The total amount of events corresponds to the total hours (2184) of the data under consideration. In order to show the performance of our system, we have divided the results into three different categories:

1) First, we analyzed the performance of the RL algorithm in terms of rewards and objective function of the MILP. In particular, we want to answer the following questions:
   - How many iterations are needed from our algorithm in order to learn how to maximize the reward and minimize the objective function?
   - Furthermore, in presence of tidal traffic effect, do we get acceptable results?
2) Secondly, we made a comparison with baseline schedulers to show how (and when) our system decides to reconfigure the network.
3) Finally we discussed the complexity of the whole system.

We run the simulations in an Ubuntu 18.04 x64 machine with 64*GB* of RAM, Intel(R) Core(TM) $i7 - 7700$ CPU @ 3.6*GHz* using openAI Gym Python library. [4]

A. PERFORMANCE OF THE RL ALGORITHM

We analyzed the performance of the RL algorithm in terms of rewards and objective function of the MILP. As shown in Table 1, each SFC has an availability class. This means that there is a maximum number of reconfigurations that can be done by the algorithm.

In Fig. 7, we show how the reward and the objective function of the MILP evolve over time. We compared the objective function in two cases: with and without RL. For the last case, we have implemented a mechanism in which the MILP model is run every hour with the freedom to reconfigure the SFCs when it is necessary. Once the maximum number of possible reconfigurations has been reached, the possibility of reconfiguring the SFCs is revoked.

[4] Website: https://gym.openai.com/ (last access 08/07/2019)

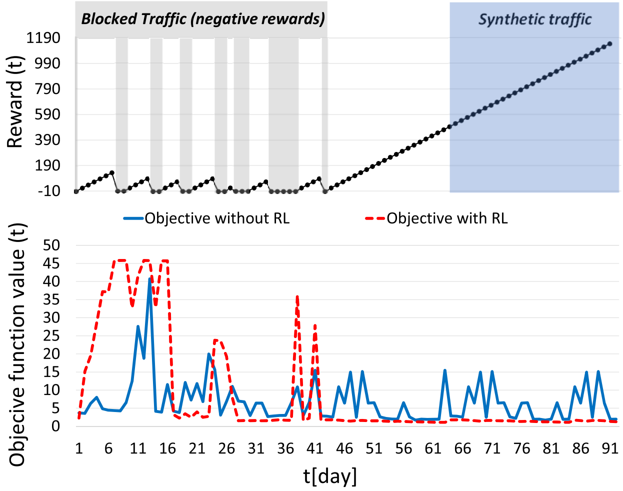

FIGURE 7. Reward and objective function for each day of the simulation.

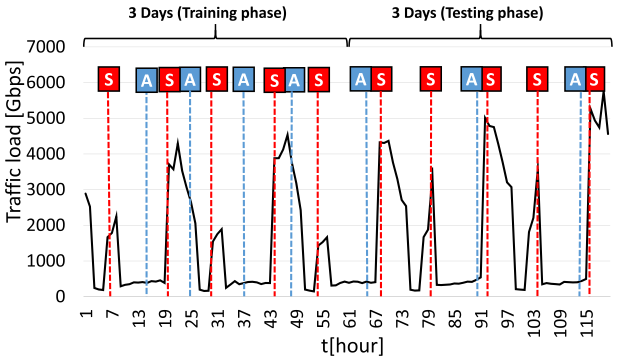

FIGURE 8. Cloud data slice 1. S represents the reconfiguration events suggested by the simulated annealing scheduler. A represents the reconfiguration events suggested by the reinforcement learning agent scheduler. The figure shows 6 days of observation: 3 days taken by the training data and 3 days taken by the test data. In this case, the agent A proposes to reconfigure the network one hour before the peak.

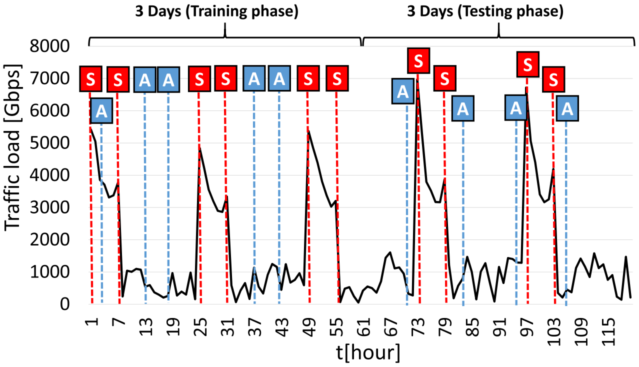

FIGURE 9. Cloud data slice 2. S represents the reconfiguration events suggested by the simulated annealing scheduler. A represents the reconfiguration events suggested by the reinforcement learning agent scheduler. The figure shows 6 days of observation: 3 days taken by the training data and 3 days taken by the test data. In this case, in addition to reconfiguring the network one hour before the peak, the agent A proposes to spend the last reconfiguration token one/two hours after the last peak.

Initially the system assigns negative rewards that decreases hour by hour when the algorithm does not block the requests. The objective function is high respect to the case without RL, mainly because the agent is deciding randomly the SFCs to reconfigure. For a total of 91 days, it took 42 days to learn how to maximize the reward function, i.e. how to reconfigure the SFCs in order to route the traffic requests. As we stated in Section I, the mechanism presented here can be seen as a way of spending reconfiguration tokens over the span of a day. The algorithm wins if it succeeds in spending these tokens in order to accept all the traffic and in the meantime to minimize the power consumption of the underline network.

The right side of the figure represents the simulations done in presence of synthetic traffic. We artificially added a month of traffic consisting of a random mixture of days belonging to the two months of real traffic. The reason for this operation is to show that our model is able to learn the tidal effect of the traffic and it is robust to its variations. As can be seen from the Fig. 7, the reward continues to grow linearly and the objective function is stable to a minimum value respect to the case without RL.

B. NETWORK PERFORMANCE

1) COMPARISON WITH BASELINE SCHEDULERS

The method proposed in this work can be used as a scheduler to perform network reconfigurations. In our previous works [35], [36], we proposed hourly reconfigurations of the optical network [35], and a scheduling heuristic to calculate a limited set of reconfiguration time points [36]. The proposed scheduling provides a good trade-off between reduction of routing changes (to avoid disruption) and resource allocation efficiency (to increase energy savings).

The scheduling algorithm is a simulated-annealing-based heuristic method, which obtains a reconfiguration scheduling (T) by finding the minimum number of reconfiguration time points $(|T|)$ with an expected allocation efficiency (total demanded bandwidth/total allocated bandwidth) [36].

In this work, we have implemented the simulated-annealing scheduler to find the best reconfiguration time points of two most sensitive SFCs, i.e. the Cloud data (see Table 1). Considering that they can be reconfigured at most 5 times a day, we plot 6 days of traffic of these SFCs, highlighting the moments in which both approaches: simulated annealing and reinforcement learning based reconfiguration algorithm, reconfigure the network. The first 3 days are taken from the training phase, the other 3 from the test phase.[5] In Figures 8 and 9 (S = *Simulated Annealing*, A = *reinforcement learning Agent*), we show two interesting behaviours of the two schedulers:

[5] Training phase refers to the initial days when the network agent made random decisions, while testing phase are those days (after day 42^{th}) in which the agent gets positive rewards and therefore has learned to make good decisions.

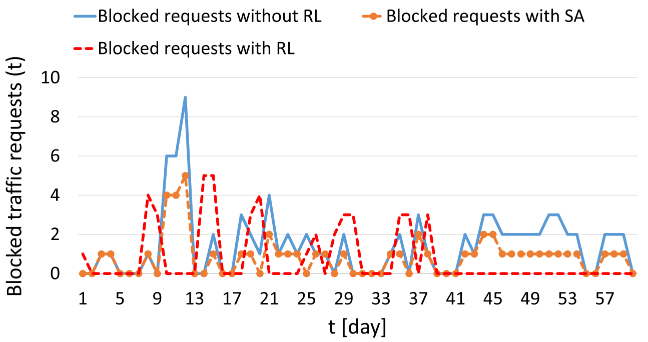

FIGURE 10. Blocked traffic requests of the three reconfiguration methods: RL-based, without RL and SA-based over 60 days of real traffic traces.

1) First, the simulated-annealing **S** proposes to reconfigure the network resources always when there is a peak in the traffic. During the 3 days of training, the agent **A** performs random reconfigurations obtaining negative rewards, while during the 3 test days, once the model has been trained, the agent **A** proposes to reconfigure the network one hour before the peak. Moreover, Fig. 8 shows that when there are two peaks close to each other, the agent does not 'spend' the last available reconfiguration token, while the simulated annealing does it. This interesting behaviour shows that the agent learnt how to maximize the reward from the past trials, predicting the next hour reconfiguration.
2) Secondly, we have noticed from Fig. 9, that in addition to reconfiguring the network one hour before the peak, the agent proposes to spend the last reconfiguration token one/two hours after the last peak. Once again, this behavior shows how the agent has learned to plan reconfiguration events in order to maximize the reward; consequently, since maximizing the reward function means finding a daily network reconfiguration schedule that reduces the number of blocked requests B^r, this contributes to the minimization of the objective function.

Finally, in Fig. 10 we show a comparison based on the number of blocked traffic requests with the three approaches considered so far: RL-based, without RL and SA-based. In this comparison we are considering only the days with real traffic traces (e.g. without the synthetic traffic as in Fig. 7). The method based on SA is slightly better than the case in which the MILP performs the reconfigurations without RL, indeed the total number of blocked traffic requests at the end of the simulations is equal to 50 and 88, respectively. Considering the reconfiguration method based on RL, the blocked requests occur during the learning phase, while after the day 42, all the traffic requests are accommodated in the network. The total number of blocked traffic requests is 40, more than half less than the case without RL.

C. COMPLEXITY ANALYSIS

The complete model is composed by two main blocks: the agent and the environment. The complexity analysis of the environment corresponds to a flow formulation in the IP/MPLS layer and a path formulation for the routing of traffic onto the optical layer. The complexity of the environment is in the RWA. It is known to be a NP-complete problem [37].

The complexity of the agent depends on the architecture of the DNN. Authors in [38] proved that training a 3 node neural network is NP-complete as well. A more recent work [39] performed an exhaustive research on computational complexity of neural network classifiers in which they derived different kind of complexity functions depending on the type of neuron functions implemented. In our case, the complexity in terms of Big $\mathcal{O}$ notation depends on the number of epochs T, input $\bar{X}$, hidden $\bar{H}$ and output nodes $\bar{Y}$ of the DNN; specifically: $\mathcal{O}[T\bar{H}(\bar{X}+\bar{Y})]$.

VII. DISCUSSION ON A REAL-WORLD DEPLOYMENT

In this section, we discuss two open issues that should be considered in a possible real-world deployment of our methodology: resolution time of optimization algorithms and NFV-SDN cross-layer resource allocation. As stated in section VI-C, the RWA problem considered in this work is known to be NP-complete. In this case, the MILP-based optimization algorithm may have a very large resolution time, even of hours or days, since it depends on the number of nodes in the network. In this case, considering the network topology and the optimization model, the proposed MILP has a resolution time of about 11 minutes and therefore the one-hour optimization interval turns to be feasible. In case of a larger network, the MILP could be optimized or even replaced by a heuristic algorithm that provides the solutions more quickly. In particular, many heuristic algorithms have been proposed in the literature to speed up the computation of RWA optimization problems. Ref. [40] provides an extensive overview on RWA algorithms. In this paper, we did not considered heuristics and we left the scalability issues for future works.

The second issue regards the NFV-SDN cross-layer resource allocation. Our proposed RL-based method does not specify the set of actions to enforce a fast and congestion-free reconfiguration of the network. Considering a real world deployment, once the optimization algorithm computes the solution to the resource allocation problem for the next time interval, it is implemented in the NFV-SDN network through the virtual infrastructure manager for the VNFs instantiation/migration and the SDN controller for the IP-over-WDM routing setup. In section V-B we introduced some recent works [31], [32] on NFV-SDN orchestration systems devised to select and allocate virtual resources in a multi-layer (packet over flexi-grid optical) network. We can infer that the cross-layer reconfiguration of an entire SFC, such as the MBB slice, would take at most 8 seconds.

The modern network/service operator needs to find a trade-off between the resolution time of the optimization algorithms and the time required to implement the NFV-SDN cross-layer resource allocation in its own network. The NFV-SDN technology allows the implementation of different ML-based network optimization algorithms, each of which

TABLE 2. Variables of the multi-layer network environment.

Variables					
Service Layer		IP/MPLS Layer		Optical Layer	
$\bar{\mathcal{G}}(\bar{\mathcal{V}},\bar{\mathcal{E}})$	Bi-directed graph of Service layer	$\hat{\mathcal{G}}(\hat{\mathcal{V}},\hat{\mathcal{E}})$	Bi-directed full-mesh graph of IP/MPLS layer	$\mathcal{G}(\mathcal{V},\mathcal{E})$	Bi-directed graph representing the optical topology
$\bar{\mathcal{V}}$	set of VNFs and PNFs $\bar{v}$	$\hat{\mathcal{V}}$	Set of nodes with virtualization capabilities $\hat{v}$	$\mathcal{V}$	Set of Optical layer nodes (ROADMs).
$\bar{\mathcal{E}}$	set of virtual links $(\bar{u},\bar{v})$ between two different NFs	$\hat{\mathcal{E}}$	Set of lightpath that can be requested to the optical layer	$\mathcal{E}$	Set of directed physical fiber links $\mathcal{V}=\hat{\mathcal{V}}$
$n_{\bar{v}}$	maximum number of replicas the NF $\bar{v}$ can be placed in the network. For a PNF $n_{\bar{v}}=1$, while for a VNF $n_{\bar{v}}\geq 1$	$T(\hat{v})$	Available transmitters in node $\hat{v}$	F_{uv}	Available number of fibers in link (u,v)
$\mathcal{R}$	set of chains that compose the requested slices	$R(\hat{v})$	Available receivers in node $\hat{v}$	L_{uv}	Capacity of the wavelengths in link (u,v) [Gbps]
$m^r_{\bar{v}}$	1 if $\bar{v}$ is part of chain r, 0 otherwise	$L_{\hat{u}\hat{v}}$	Maximum rate of the wavelengths for lightpath $(\hat{u},\hat{v})$	$\delta^{\hat{u}\hat{v},p}_{uv}$	1 if link (u,v) of lightpath $(\hat{u},\hat{v})$ is in path p, 0 otherwise
$\eta^r_{\bar{u}\bar{v}}$	1 if virtual link $(\bar{u},\bar{v})\in\bar{\mathcal{E}}$ is part of chain r, 0 otherwise	$l_{\bar{v}\hat{v}}$	1 if NF $\bar{v}$ can be placed in node $\hat{v}$, 0 otherwise	$\mathcal{P}_{\hat{u}\hat{v}}$	Set of optical paths for lightpath $(\hat{u},\hat{v})$
b^r	latency requirement of chain r $[\mu s]$	$\tilde{y}^r_{\bar{v}\hat{v}}$	current VNF placement of chain r. 1 if NF $\bar{v}$ is placed in node $\hat{v}$, 0 otherwise	$\sigma^{\hat{u}\hat{v}}_p$	Delay of path $p\in\mathcal{P}_{\hat{u}\hat{v}}$ of tunnel $(\hat{u},\hat{v})$
$h^r_{\bar{u}\bar{v}}$	**traffic volume request between NFs** $(\bar{u},\bar{v})$ of chain r [Gbps]	$C_{\hat{v}}$	Number of CPU cores available in node $\hat{v}$	ρ_p	Minimum number of fibers in the path $p\in\mathcal{P}_{\hat{u}\hat{v}}$
θ^r	service disruption penalty due to reconfiguration of chain r			W	Maximum number of wavelength channels per fiber
Ω^r	Blocking penalty of chain r				
$\Delta_{\bar{v}}$	Processing latency per Gbps. $\Delta_{\bar{v}}=132\mu s$ [42]				
$c_{\bar{v}}$	Number of CPU cores per Gbps required by NF $\bar{v}$				
$o_{\bar{v}}$	1 if the $\bar{v}$ is a VNF, 0 if it is a PNF				

aims at a different cost function. The operator can simultaneously train different ML algorithms in off-line mode, i.e. without implementing the output in the real network, and only after testing each of them, he can deploy the one that gives the best performance. To conclude, NFV, SDN and ML enables intelligent control and configuration actions in a very short time and represents a fundamental feature of future telecommunication networks.

VIII. CONCLUSION

This article proposes an effective self-learning system based on RL to optimize the resources allocation of SFCs in NFV-SDN enabled metro-core optical networks. Exploiting the programmability and full network visibility leveraged by SDN and NFV, this system can be deployed as an SDN application to perform reconfiguration of the network based on its current state and historical traffic load. The proposed RL system was tested with a reference metro-core network architecture proposed by the Metro-Haul European project [6] and a real traffic data-set from Milan urban area, Italy. Initially the system assigns negative rewards that decreases hour by hour when the algorithm does not block the requests. After 42 days of training, the RL agent figured out how to maximize the reward function and keep the objective function of the MILP stable to a minimum value respect to the case in which we do not use RL. In other words, it learnt how and when to reconfigure the SFCs in order to route the traffic requests decreasing the blocking probability. As additional figure of merit, we have implemented a state-of-the-art simulated-annealing scheduler to find the best reconfiguration time points of two types of network slices. Comparisons with the proposed RL-based reconfiguration algorithm showed that it is able to predict sudden changes in traffic shape and trigger the reconfiguration of the SFCs in advance.

APPENDIX
MULTI-LAYER NETWORK MODEL FORMULATION

This section is dedicated to the mathematical explanation of the multi-layer network model. As mentioned in Section IV, the network is composed of 3 layers: Service, IP/MPLS and Optical. Each of these layer is defined in terms of variables and input parameters described in Table 2 and Table 3, respectively. Table 4 is the complete mathematical formulation of the multi-layer network model.

A. FORMULATION IN TABLE 4

The goal of the objective function in (1) is to maximize the number of successfully routed chains and minimize the reconfiguration penalty. The first term in (1): $\sum_{r\in\mathcal{R}}\{\Omega^r(1-B^r)+\theta^r\}$, minimizes the number blocked SFC requests and the service disruption due to the reconfiguration of chain $r\in\mathcal{R}$. The second term minimizes the power consumption of the IP/MPLS and Optical layer where α refers to the overall cost of optical network devices such as: OXC, OLA and WDM terminals with 80 wavelengths channels [41]. This model is a multi-objective path formulation MILP focused on the mapping of slices on a network, and the routing and wavelength assignment of the input traffic through the chains composing the slices on the physical network. For each traffic request it is able to create

TABLE 3. Input parameters of the multi-layer network environment.

Input Parameters			
Service Function Chaining and VNF Placement		Traffic Grooming and RWA	
B^r	1 **if chain** r **is successfully allocated,** 0 **otherwise**	$V_{\hat{u}\hat{v}}$	Number of active lightpaths between $(\hat{u}, \hat{v})$
i^r	**Number of reconfigurations in the VNF placement of chain** r	$V^{w,r}_{\hat{u}\hat{v}}$	Number of active lightpaths between $(\hat{u}, \hat{v})$ on wavelength w of chain r
$i^r_{\bar{v}\hat{v}}$	1 if NF $\bar{v}$ will be move in or out of node $\hat{v}$, 0 if it remains in the same node	$T^{w,r}_{\hat{u}\hat{v}}$	Fraction of the capacity used by chain r in lightpaths between $(\hat{u}, \hat{v})$ on wavelength w. If $T^{w,r}_{\hat{u}\hat{v}} \geq 1$ r uses multiple lightpaths (over other fibers or link-disjoint path)
$y^r_{\bar{v},\hat{v}}$	1 if NF $\bar{v}$ of chain r is placed in node $\hat{v}$, 0 otherwise		
$\hat{y}_{\bar{v},\hat{v}}$	1 if NF $\bar{v}$ is placed in node $\hat{v} \in \hat{\mathcal{V}}$, 0 otherwise	$P^{\hat{u}\hat{v},w,r}_p$	Number of lightpaths on wavelength w between $(\hat{u}, \hat{v})$ used by chain r and routed through precomputed-path $p \in \mathcal{P}_{\hat{u}\hat{v}}$
$\lambda^{\bar{u}\bar{v},r}_{\hat{u}\hat{v}}$	1 if virtual link $(\bar{u}, \bar{v})$ of chain r uses a lightpath between $(\hat{u}, \hat{v})$	$\hat{P}^{\hat{u}\hat{v},r}_p$	Binary, 1 if a lightpath between $(\hat{u}, \hat{v})$ is used by chain r and routed through path $p \in \mathcal{P}_{\hat{u}\hat{v}}$
I^r	**It is the output of the Network Agent. It is** 0 **when the agent decided to not reconfigure the chain** r**,** 1 **otherwise**	$S^r_{\hat{u}\hat{v}}$	Maximum delay among the lightpaths of chain r between $(\hat{u}, \hat{v})$

TABLE 4. Multi-layer network environment formulation.

<table>
<tr><th colspan="2">Multi-layer network environment formulation</th></tr>
<tr><td colspan="2"><i>Objective</i>: Minimize blocking and consumed bandwidth
$min \quad \sum_{r\in\mathcal{R}} \{\Omega^r(1-B^r)+\theta^r\} + \alpha \sum_{r\in\mathcal{R}} \sum_{(\bar{u},\bar{v})\in\bar{\mathcal{E}}} \sum_{(\hat{u},\hat{v})\in\hat{\mathcal{E}}} \sum_{p\in\hat{\mathcal{P}}_{\hat{u},\hat{v}}} \sum_{(u,v)\in\mathcal{E}} h^r_{\bar{u}\bar{v}}\eta^r_{\bar{u}\bar{v}}\lambda^{\bar{u}\bar{v},r}_{\hat{u}\hat{v}}\hat{P}^{\hat{u}\hat{v},r}_p\delta^{\hat{u}\hat{v},p}_{uv}$ (1)</td></tr>
<tr><th>SFC and VNF Placement</th><th>Reconfigurations and Blocking</th></tr>
<tr><td>$\hat{y}_{\bar{v},\hat{v}} \leq l_{\bar{v},\hat{v}}; \quad \forall\bar{v}\in\mathcal{D}, \hat{v}\in\hat{\mathcal{V}},\ r\in\mathcal{R}$ (2)
$\sum_{\hat{v}\in\hat{\mathcal{V}}} \hat{y}_{\bar{v},\hat{v}} \leq n_{\bar{v}} \quad \forall\bar{v}\in\mathcal{D}$ (3)
$y^r_{\bar{v},\hat{v}} \leq \hat{y}_{\bar{v},\hat{v}}$
$\forall\bar{v}\in\mathcal{D}, \bar{r}\in\mathcal{R}, \hat{v}\in\hat{\mathcal{V}}$ (4)
$\lambda^{\bar{u}\bar{v},r}_{\hat{u}\hat{v}} \leq \eta^r_{\bar{u}\bar{v}} * y^r_{\bar{u},\hat{u}} * y^r_{\bar{v},\hat{v}}$
$\forall\ (\bar{u},\bar{v})\in\bar{\mathcal{E}}, (\hat{u},\hat{v})\in\hat{\mathcal{E}}, r\in\mathcal{R}$ (5)
$\sum_{r\in\mathcal{R}} \sum_{(\bar{u}\bar{v})\in\bar{\mathcal{V}}} h^r_{\bar{u}\bar{v}}\eta^r_{\bar{u}\bar{v}}c_{\bar{v}}y^r_{\bar{v}\hat{v}} \leq C_{\hat{v}}$ (6)</td><td>$\left(y^r_{\bar{v}\hat{v}} + \tilde{y}^r_{\bar{v}\hat{v}} - 2y^r_{\bar{v}\hat{v}}\tilde{y}^r_{\bar{v}\hat{v}}\right) m^r_{\bar{v}} = i^r_{\bar{u}\bar{v}}; \quad \forall\ r\in\mathcal{R}, \bar{v}\in\bar{\mathcal{V}}, \hat{v}\in\hat{\mathcal{V}}$ (7)
$\sum_{\bar{v}\in\bar{\mathcal{V}}} \sum_{\hat{v}\in\hat{\mathcal{V}}} i^r_{\bar{u}\bar{v}}m^r_{\bar{v}} = i^r; \quad \forall\ r\in\mathcal{R}$ (8)
$i^r \leq [I^r + (1-B^r)]\,\mathcal{M}; \quad \forall\ r\in\mathcal{R}$ (9)
$B^r\eta^r_{\bar{u}\bar{v}} = \sum_{(\hat{u}\hat{v})\in\hat{\mathcal{E}}} \lambda^{\bar{u}\bar{v},r}_{\hat{u}\hat{v}}; \quad \forall(\bar{u},\bar{v})\in\bar{\mathcal{E}},\ r\in\mathcal{R}$ (10)
$B^r m^r_{\bar{v}} = \sum_{\hat{v}\in\hat{\mathcal{V}}} y^r_{\bar{v},\hat{v}}; \quad \forall\bar{v}\in\mathcal{D}, r\in\mathcal{R}$ (11)</td></tr>
<tr><th>Traffic Grooming</th><th>Routing and Wavelength Assignment</th></tr>
<tr><td>$\sum_{\hat{v}\in\hat{\mathcal{V}}:(\hat{u},\hat{v})\in\hat{\mathcal{E}},\hat{u}\neq\hat{v}} V_{\hat{u}\hat{v}} \leq T(\hat{u}); \quad \forall\hat{u}\in\hat{\mathcal{V}}$ (12)
$\sum_{\hat{u}\in\hat{\mathcal{V}}:(\hat{u},\hat{v})\in\hat{\mathcal{E}},\hat{u}\neq\hat{v}} V_{\hat{u}\hat{v}} \leq R(\hat{v}); \quad \forall\hat{v}\in\hat{\mathcal{V}}$ (13)
$\sum_{w\in\mathcal{W},r\in\mathcal{R}} V^{w,r}_{\hat{u}\hat{v}} = V_{\hat{u}\hat{v}}; \quad \forall\ (\hat{u},\hat{v})\in\hat{\mathcal{E}}, \hat{u}\neq\hat{v}$ (14)
$\sum_{(\bar{u},\bar{v})\in\bar{\mathcal{E}}} h^r_{\bar{u}\bar{v}}\lambda^{\bar{u}\bar{v},r}_{\hat{u}\hat{v}} = \sum_{w\in\mathcal{W}} T^{w,r}_{\hat{u}\hat{v}}L_{\hat{u}\hat{v}}$
$\forall\ (\hat{u},\hat{v})\in\hat{\mathcal{E}}\vert\hat{u}\neq\hat{v}, r\in\mathcal{R}$ (15)
$T^{w,r}_{\hat{u}\hat{v}} \leq V^{w,r}_{\hat{u}\hat{v}}; \quad \forall(\hat{u},\hat{v})\in\hat{\mathcal{E}},\ w\in\mathcal{W}, r\in\mathcal{R}$ (16)</td><td>$\sum_{p\in\mathcal{P}_{\hat{u}\hat{v}}\vert(u,v)\in p} \sum_{r\in\mathcal{R}} \sum_{(\hat{u}\hat{v})\in\hat{\mathcal{E}}} \delta^{\hat{u}\hat{v},p}_{uv}P^{\hat{u}\hat{v},w,r}_p \leq F_{uv}; \quad \forall\ (u,v)\in\mathcal{E}, w\in\mathcal{W}$ (17)
$\sum_{p\in\mathcal{P}_{\hat{u}\hat{v}}} P^{\hat{u}\hat{v},w,r}_p = V^{w,r}_{\hat{u}\hat{v}}; \quad \forall(\hat{u},\hat{v})\in\hat{\mathcal{E}}\vert\hat{u}\neq\hat{v},\ w\in\mathcal{W},\ r\in\mathcal{R}$ (18)
$\sum_{w\in\mathcal{W}} \sum_{p\in\mathcal{P}_{\hat{u}}} P^{\hat{u}\hat{v},w,r}_{uv} \geq \lambda^{\bar{u}\bar{v},r}_{\hat{u}\hat{v}}; \quad \forall\ (\bar{u},\bar{v})\in\bar{\mathcal{E}},\ (\hat{u},\hat{v})\in\hat{\mathcal{E}}\ \vert\ \hat{u}\neq\hat{v},\ r\in\mathcal{R}$ (19)
$P^{\hat{u}\hat{v},w,r}_p \leq \rho_p \sum_{(\bar{u}\bar{v})\in\bar{\mathcal{E}}} \lambda^{\bar{u}\bar{v},r}_{\hat{u}\hat{v}} \quad \forall\ (\hat{u},\hat{v})\in\hat{\mathcal{E}}\vert\hat{u}\neq\hat{v},\ p\in\mathcal{P}_{\hat{u}\hat{v}},\ r\in\mathcal{R},\ w\in\mathcal{W}$ (20)
$\hat{P}^{\hat{u}\hat{v},r}_p \geq P^{\hat{u}\hat{v},w,r}_p \quad \forall\ (\hat{u},\hat{v})\in\hat{\mathcal{E}}\vert\hat{u}\neq\hat{v},\ p\in\mathcal{P}_{\hat{u}\hat{v}},\ r\in\mathcal{R},\ w\in\mathcal{W}$ (21)</td></tr>
<tr><th colspan="2">Latency constraints</th></tr>
<tr><td colspan="2">$S^r_{\hat{u}\hat{v}} \geq \sigma^{\hat{u}\hat{v}}_p\hat{P}^{\hat{u}\hat{v},r}_p; \quad \forall r\in\mathcal{R},\ (\hat{u},\hat{v})\in\hat{\mathcal{E}}\ \vert\ \hat{u}\neq\hat{v},\ p\in\mathcal{P}_{\hat{u}\hat{v}}$ (22)
$\sum_{(\hat{u}\hat{v})\in\hat{\mathcal{E}}\vert\hat{u}\neq\hat{v}} S^r_{\hat{u}\hat{v}} + \sum_{\hat{v}\in\hat{\mathcal{V}}} \Delta_{\bar{v}}h^r_{\bar{u}\bar{v}} \leq b^r \quad \forall\ r\in\mathcal{R}$ (23)</td></tr>
</table>

multiple paths on the optical network, from the source to the destination node and it can also route the traffic on multiple fibers on a single link.

B. CONSTRAINTS ON SFC AND VNF PLACEMENT

Constraint (2) enforces that every VNF is mapped on a physical node among the available ones. The expressions (3) and (4) prevent the use of a VNF more than a certain number of times specified as input of the MILP. (5) creates a tunnel between two nodes only if there exist traffic between the VNFs associated to the MPLS nodes. (6) ensures that each VNF in the chain r has enough capacity in terms of number of CPU available for the incoming traffic.

C. CONSTRAINTS ON RECONFIGURATIONS AND BLOCKING

(7) derives the number of reconfigurations of the VNFs. (8) calculates i^r which is the number of reconfigurations in the VNF placement of chain r. The difference with (7) is that the information is computed by the network function and by the chain, while (8) calculates the total number of reconfigurations in the whole chain. The constraint in **(9)** is modified by the agent. It suggests to reconfigure, or not, the chain r by choosing the value of I^r. If it is 0, then the ILP will not make reconfigurations of the chain r, 1 otherwise. The constraints (10) and (11) calculate if a chain is blocked either due to lack of optical capacity (10) or lack of virtual capacity (11).

D. CONSTRAINTS ON TRAFFIC GROOMING

In each node in the MPLS layer the number of tunnels coming in or going out from the nodes must be less than the number of transmitters or receivers in the node. Thus, it is enforced by the constraints (12) and (13). (14) ensures that the number of virtual links from two nodes in the MPLS layer is the sum of the tunnels on each wavelength between the two nodes. The constraints (15) (16) find the percentage of capacity used and the usage of each MPLS tunnel and wavelength for each demand.

E. CONSTRAINTS ON ROUTING AND WAVELENGTH ASSIGNMENT

The constraint (17) ensures that all the paths passing on a node for each wavelength do not exceed the number of active fibers present in the optical link. (18) ensures that we have at least one physical path for each optical tunnel of the IP/MPLS layer, while (19) and (20) map the path on the physical layer for each IP/MPLS tunnels with different source and destination nodes. (21) tells us if there exist at least one path between a source and a destination nodes for a certain slice.

F. CONSTRAINTS ON LATENCY

For each tunnel, equation (22) searches the longest optical path in terms of propagation delay. (23) is the latency constraint for each request and slice and ensures that all the traffic demands for a certain slice do not exceed the latency required by the slice. The delay is calculated as the sum of the longest paths for each part of a chain. This can be done because our hypothesis is that each VNF in the network aggregates the traffic of each chain, so the maximum delay between two VNFs is given by the longest path between these two VNFs.

Among the many variables and input parameters, the MILP (or the network environment) exposes 3 of these as observations. In particular, at each time step, i.e. hour, it exposes the observations to the agent and the reward function, which are:

- $O_1(t) \rightarrow i^r(t)$: equal to 1 if there are reconfigurations of VNFs of chain r at time t, 0 otherwise
- $O_2(t) \rightarrow B^r(t)$: equal to 0 if chain r is blocked at time t, 1 otherwise
- $O_3(t) \rightarrow h^r_{\bar{u}\bar{v}}(t)$: traffic volume request between VNFs $(\bar{u}, \bar{v})$ of chain r [Gbps] at time t

At each time step, the environment receives an input, or *action*, from the network agent. The action modifies an input parameter of the environment, that is I^r (see Table 3). If it is equal to 1, the MILP is free to reconfigure the chain r, otherwise if it is 0, the MILP cannot make any reconfiguration.

ACKNOWLEDGMENT

The work leading to these results has been supported by the European Community under grant agreement no. 761727 Metro-Haul project.

REFERENCES

[1] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[2] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "NFV: State of the art, challenges and implementation in next generation mobile networks (vEPC)," 2014, *arXiv:1409.4149*. [Online]. Available: https://arxiv.org/abs/1409.4149

[3] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Deng, "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action," in *Proc. SDN OpenFlow World Congr.*, vol. 48, 2012, pp. 1–16.

[4] T. Soenen, R. Banerjee, W. Tavernier, D. Colle, and M. Pickavet, "Demystifying network slicing: From theory to practice," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, May 2017, pp. 1115–1120.

[5] M. R. Raza, M. Fiorani, A. Rostami, P. Öhlen, L. Wosinska, and P. Monti, "Benefits of programmability in 5G transport networks," in *Proc. Opt. Fiber Commun. Conf.*, 2017, pp. 1–3, Paper M2G.3.

[6] *The Metro-Haul Project Deliverables*. [Online]. Available: https://metrohaul.eu/deliverables/

[7] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, and A. Pattavina, "Protection strategies for virtual network functions placement and service chains provisioning," *Networks*, vol. 70, no. 4, pp. 373–387, 2017.

[8] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2014, pp. 7–13.

[9] C. Ghribi, M. Mechtri, and D. Zeghlache, "A dynamic programming algorithm for joint VNF placement and chaining," in *Proc. ACM Workshop Cloud-Assist. Netw.*, 2016, pp. 19–24.

[10] A. Gupta, B. Jaumard, M. Tornatore, and B. Mukherjee, "A scalable approach for service chain mapping with multiple SC instances in a wide-area network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 529–541, Mar. 2018.

[11] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE Conf. Comput. Commun.*, Apr./May 2015, pp. 1346–1354

[12] L. Askari, A. Hmaity, F. Musumeci, and M. Tornatore, "Virtual-network-function placement for dynamic service chaining in metro-area networks," in *Proc. Int. Conf. Opt. Netw. Design Modeling (ONDM)*, May 2018, pp. 136–141.

[13] X. Wang, C. Wu, F. Le, A. Liu, Z. Li, and F. Lau, "Online VNF scaling in datacenters," in *Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD)*, Jun./Jul. 2016, pp. 140–147.

[14] A. Leivadeas, M. Falkner, I. Lambadaris, and G. Kesidis, "Resource management and orchestration for a dynamic service chain steering model," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[15] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 3, pp. 543–553, Sep. 2017.

[16] J.-J. Pedreno-Manresa, P. S. Khodashenas, M. S. Siddiqui, and P. Pavon-Marino, "On the need of joint bandwidth and NFV resource orchestration: A realistic 5G access network use case," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 145–148, Jan. 2018.

[17] X. Chen, Z. Li, Y. Zhang, R. Long, H. Yu, X. Du, and M. Guizani, "Reinforcement learning–based QoS/QoE-aware service function chaining in software-driven 5G slices," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 11, p. e3477, 2018.

[18] P. Reichl, S. Egger, R. Schatz, and A. D'Alconzo, "The logarithmic nature of QoE and the role of the Weber-Fechner law in QoE assessment," in *Proc. IEEE Int. Conf. Commun.*, May 2010, pp. 1–5.

[19] M. Fiedler, T. Hossfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," *IEEE Netw.*, vol. 24, no. 2, pp. 36–41, Mar./Apr. 2010.

[20] R. Li, Z. Zhao, Q. Sun, I. Chi-Lin, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," 2018, *arXiv:1805.06591*. [Online]. Available: https://arxiv.org/abs/1805.06591

[21] R. Alvizu, S. Troia, G. Maier, and A. Pattavina, "Machine-learning-based prediction and optimization of mobile metro-core networks," in *Proc. IEEE Photon. Soc. Summer Top. Meeting Ser. (SUM)*, Jul. 2018, pp. 155–156.

[22] D. C. Park, M. A. El-Sharkawi, R. J. Marks, L. E. Atlas, and M. J. Damborg, "Electric load forecasting using an artificial neural network," *IEEE Trans. Power Syst.*, vol. 6, no. 2, pp. 442–449, May 1991.

[23] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.

[24] N. M. Nasrabadi, "Pattern recognition and machine learning," *J. Electron. Imag.*, vol. 16, no. 4, 2007, Art. no. 049901.

[25] D. Kroese and R. Rubinstein, *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. New York, NY, USA: Springer-Verlag, 2004.

[26] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.

[27] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, May 1996.

[28] P. Öhlén, B. Skubic, A. Rostami, M. Fiorani, P. Monti, Z. Ghebretensaé, J. Mårtensson, K. Wang, and L. Wosinska, "Data plane and control architectures for 5G transport networks," *J. Lightw. Technol.*, vol. 34, no. 6, pp. 1501–1508, Mar. 15, 2016.

[29] Technical Specification Group Services and System Aspect, *Service Requirements for Next Generation New Services and Markets*, document TS22.261, 3GPP, 2017.

[30] P. Teli, M. V. Thomas, and K. Chandrasekaran, "Big data migration between data centers in online cloud environment," *Procedia Technol.*, vol. 24, pp. 1558–1565, Jan. 2016.

[31] S. Fichera, R. Martínez, B. Martini, M. Gharbaoui, R. Casellas, R. Vilalta, R. Muñoz, and P. Castoldi, "Latency-aware resource orchestration in SDN-based packet over optical flexi-grid transport networks," *J. Opt. Commun. Netw.*, vol. 11, no. 4, pp. B83–B96, Apr. 2019.

[32] M. I. Biswas, G. Parr, S. McClean, P. Morrow, and B. Scotney, "An analysis of live migration in openstack using high speed optical network," in *Proc. SAI Comput. Conf.*, Jul. 2016, pp. 1267–1272.

[33] S. Troia, G. Sheng, R. Alvizu, G. A. Maier, and A. Pattavina, "Identification of tidal-traffic patterns in metro-area mobile networks via matrix factorization based model," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2017, pp. 297–301.

[34] *The Big Data Challenge*, TIM, Rome, Italy, 2014. [Online]. Available: https://dandelion.eu/datamine/open-big-data/

[35] R. Alvizu, X. Zhao, G. Maier, Y. Xu, and A. Pattavina, "Energy aware optimization of mobile metro-core network under predictable aggregated traffic patterns," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–7.

[36] R. Alvizu, X. Zhao, G. Maier, Y. Xu, and A. Pattavina, "Energy efficient dynamic optical routing for mobile metro-core networks under tidal traffic patterns," *J. Lightw. Technol.*, vol. 35, no. 2, pp. 325–333, Jan. 15, 2017.

[37] I. Chlamtac, A. Ganz, and G. Karmi, "Lightpath communications: An approach to high bandwidth optical WAN's," *IEEE Trans. Commun.*, vol. 40, no. 7, pp. 1171–1182, Jul. 1992.

[38] A. Blum and R. L. Rivest, "Training a 3-node neural network is NP-complete," in *Proc. Adv. Neural Inf. Process. Syst.*, 1989, pp. 494–501.

[39] M. Bianchini and F. Scarselli, "On the complexity of neural network classifiers: A comparison between shallow and deep architectures," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 8, pp. 1553–1565, Aug. 2014.

[40] B. Zhou, M. A. Bassiouni, and G. Li, "Routing and wavelength assignment in optical networks using logical link representation and efficient bitwise computation," *Photon. Netw. Commun.*, vol. 10, no. 3, pp. 333–346, Nov. 2005.

[41] W. Van Heddeghem, F. Idzikowski, W. Vereecken, D. Colle, M. Pickavet, and P. Demeester, "Power consumption modeling in optical multilayer networks," *Photon. Netw. Commun.*, vol. 24, no. 2, pp. 86–102, Oct. 2012.

[42] A. Gupta, M. Tornatore, B. Jaumard, and B. Mukherjee, "Virtual-mobile-core placement for metro network," 2018, *arXiv:1801.05537*. [Online]. Available: https://arxiv.org/abs/1801.05537

SEBASTIAN TROIA received the B.Sc. and M.Sc. degrees in telecommunications engineering from the Politecnico di Milano, in 2013 and 2016, respectively, and the Ph.D. degree in information technology from the BONSAI LAB, Politecnico di Milano, in November 2016. His research interests are related to machine learning algorithms for communications networks,software-defined networking, network orchestration–automation–optimization, SD-WAN, and optical multipath routing.

RODOLFO ALVIZU received the degree in telecommunications engineering from the National Experimental University of the Armed Forces, Venezuela, in 2008, the M.Sc. degree in electronics engineering from Simon Bolivar University (USB), Venezuela, in 2011, and the Ph.D. degree in information technologies from the Politecnico di Milano, Italy, in 2017. From 2011 to 2013, he was an Assistant Professor with USB. In 2016, he has co-founded SWAN Networks, a startup developing automation and optimization software based on SDN. In 2017, SWAN networks became spin-off of the Politecnico di Milano. His research interests are related to software-defined networking, network orchestration–automation–optimization, SD-WAN, and optical multipath routing.

GUIDO MAIER received the Laurea degree in electronic engineering and the Ph.D. degree in telecommunication engineering from the Politecnico di Milano, Italy, in 1995 and 2000, respectively. From 1995 to February 2006, he was a Researcher with CoreCom (research consortium supported by Pirelli and PoliMi in Milan, Italy), where he achieved the position of the Head of the Optical Networking Laboratory. In March 2006, he joined the Department of Electronics, Information and Bioengineering, Politecnico di Milano, as an Assistant Professor, where he became an Associate Professor, in January 2015. He has participated in several joint research projects with industrial partners, such as Pirelli, Telecom Italia, Fastweb, and RFI. He has been involved in the IST-FP6 and FP7 European Projects, such as MUPBED, NOBEL2, e-Photon/ONe+, BONE, STRONGEST, and (IRSES) MobileCloud. He has authored or coauthored over 90 articles published in proceedings of international conferences and 35 articles published in international journals on networking and optical networks.