INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT
Int. J. Network Mgmt 2016; **00**:1–23
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nem

Stochastic and Exact Methods for Service Mapping in Virtualized Network Infrastructures

Francesco Liberati[1]*, Alessandro Giuseppi[2], Antonio Pietrabissa[2], Vincenzo Suraci[1], Alessandro Di Giorgio[2], Marco Trubian [3], David Dietrich[4], Panagiotis Papadimitriou[5], Francesco Delli Priscoli[2]

[1]*SMART Engineering Solutions & Technologies, eCampus University, Via Isimbardi 10, 22060 Novedrate (CO), Italy*
[2]*DIAG Department, University of Rome "La Sapienza", Via Ariosto 25, 00185 Rome, Italy*
[3]*Department of Informatics, Universitá degli Studi di Milano, Via Comelico 39/41, 20135 Milano, Italy*
[4]*Institute of Communications Technology (IKT), Leibniz Universität, Hannover Appelstr. 9A, 30167 Hannover, Germany*
[5]*Department of Applied Informatics, University of Macedonia, Egnatia 156, 54621 Thessaloniki, Greece*

SUMMARY

This paper presents a stochastic algorithm for virtual network service mapping in virtualized network infrastructures, based on Reinforcement Learning (RL). An exact mapping algorithm in line with the current state of the art and based on integer linear programming is proposed as well, and the performances of the two algorithms are compared. While most of the current works in literature report exact or heuristic mapping methods, the RL algorithm presented here is instead a stochastic one, based on Markov decision processes theory. The aim of the RL algorithm is to iteratively learn an efficient mapping policy, which could maximize the expected mapping reward in the long run. Based on the review of the state of the art, the paper presents a general model of the service mapping problem and the mathematical formulation of the two proposed strategies. The distinctive features of the two algorithms, their strengths and possible drawbacks are discussed and validated by means of numeric simulations in a realistic emulated environment. Copyright © 2016 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Network functions virtualization; service mapping; Markov decision process; reinforcement learning; virtual network functions

1. INTRODUCTION

1.1. Background and Aim

Network Function Virtualization (NFV) [1] refers to the substitution of dedicated network function devices with functionally equivalent software modules running on commercial-off-the-shelf (COTS) hardware, through software virtualization techniques. NFV is expected to bring

*Correspondence to: eCampus University, Via Isimbardi 10, 22060, Novedrate CO, Italy. E-mail: liberatifnc@gmail.com

Contract/grant sponsor: This work was supported in part by the European Commission through the ICT-2013.1.1 FP7 T-NOVA project; contract/grant number: 619520.

significant advantages, including [1]: i) increased capital and operational efficiencies; ii) rapid service deployment and customization/reconfiguration of Virtual Network Functions (VNFs) [2] and the underlying hardware; iii) a more stimulating service ecosystem; iv) increased flexibility in management of network functions, etc. Additionally, the sharing of infrastructure resources, workload migration and powering down is expected to bring relevant energy savings. Aside of the new capabilities offered, NFV demands for improved or entirely new orchestration and management functions, business models and architectures [3]. Since NFV allows for dynamic coupling of VNFs and the supporting hardware, new orchestration logic is needed to govern the process of services instantiation, termination and overall life-cycle management. In particular, this paper focuses on the problem of *service mapping*, that is, the problem of intelligently assigning network infrastructure resources to Network Service (NS)[†] to be instantiated, as detailed in Section 3. The aim of the paper is to propose and test two different service mapping algorithms: an innovative one, based on a stochastic learning technique, and a state of the art one based on Integer Linear Programming (ILP). The two algorithms are compared and their performance is assessed on a realistic simulation base. The results presented in this paper are the outcome of the research performed in the context of the European FP7 project T-NOVA [4].

1.2. Main Contributions

The main contributions of the paper are:

1. The proposition of a general modeling framework for the service mapping problem, based on graph theory, and the subsequent derivation of a Markov decision process model, which is at the base of the Reinforcement Learning (RL) algorithm presented.
2. The proposition of a novel RL algorithm for service mapping, providing a solution that seeks to maximize the performances in the long run, whereas previous works in literature have mainly focused on exact or heuristic methods maximizing performance indicators at current time.
3. The proposition of a second, exact method for service mapping, based on ILP, and the comparison of the characteristics of the two algorithms and the performances achieved. The simulations are based on realistic settings and data taken from a specialized database for testing algorithms in NFV environments.

Previous efforts of the authors and the T-NOVA project in the field of service mapping can be found in [5] and [6], providing: i) a first discussion of the reference scenario and modeling framework; ii) a high-level discussion of the two mapping strategies proposed, with in particular a first proposition of the RL method, limited to the case of single VNF mapping, and only outlying a possible way of extending it to the complete NS mapping case; iii) early simulation results mainly focused on the ILP method for what concerns complete NS mapping. The key novelties of this paper with respect to [5] and [6] are: i) an extended state of the art in service mapping methods; ii) a radical improvement of the RL algorithm, with an improved formulation of the state space and the reward function, and a deeper extension and investigation of the RL method in the general case of complete NS mapping (i.e. mapping of NS composed by more than one VNF); iii) an improvement of the ILP algorithm,

[†] A NS is a service composed by one or different specific VNFs [2].

DOI: 10.1002/nem

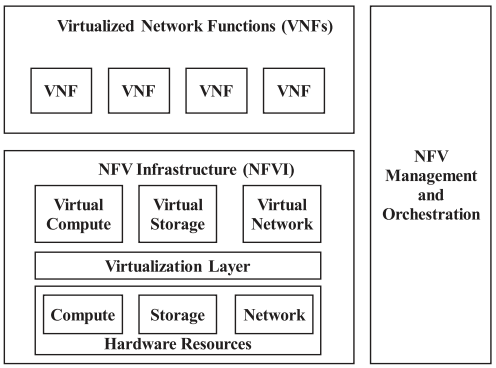

Figure 1. ETSI reference framework for NFV (picture from [1]).

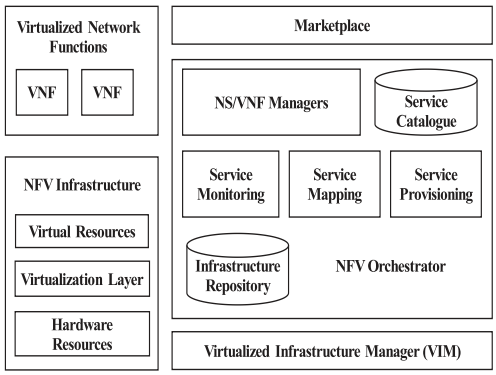

Figure 2. T-NOVA reference architecture.

to increase load balancing capabilities; vi) an actual comparison of the two algorithms running on the same simulated test scenarios (a delicate task due to the entirely different nature of the two algorithms).

1.3. Paper Organization

The remainder of the paper is organized as follows: Section 4 presents a review of the state of the art in service mapping; Section 2 deals with the reference scenario and architecture considered in this paper; Section 3 defines and models the service mapping problem; Section 5 and 6 detail, respectively, the design of the RL and the ILP service mapping algorithms; Section 7 presents and discusses simulation results; Section 8 concludes the paper and outlines future works.

2. REFERENCE SCENARIO AND ARCHITECTURE

This section presents the reference scenario and the NS orchestration architecture developed in the T-NOVA project [4] and assumed for the design of the mapping algorithms. The interested reader is referred to [1], [2], [5], [6], [7] for more details. The assumed NFV architecture is in line with the European Telecommunications Standards Institute (ETSI) high level framework for NFV, as reported in Figure 1: NS provisioning leverages virtual compute, storage and network resources made available by the virtual infrastructure manager through control of physical hardware resources. Key actors and systems relevant to frame the discussion in this paper include:

1. VNF/NS Function Providers: the actors responsible for VNFs and NSs provisioning, and for maintenance of the related *service descriptors* [2], i.e. the templates reporting VNFs/NSs data essential for correct orchestration.
2. Infrastructure Repository: a system which stores and makes available information on the physical network infrastructure supporting virtualization. Such information is abstracted by the virtualized infrastructure manager, described below. Together with service catalogs, described below, this is the main system that the service mapping module has to interact with in order to gather the needed inputs for solving the problem.
3. Marketplace: the system in charge of contract management, purchase of NSs, Service Level Agreement (SLA) management, billing, etc.

DOI: 10.1002/nem

4. NFV Orchestrator: the system responsible for the technical VNF/NS life-cycle management (request receipt, NSs on-boarding, instantiation of VNF managers, service reconfiguration, scaling, events management, performance measurement, service termination, etc.). The orchestrator hosts, among the others, the service mapping module, which has been designed in a flexible way in order to allow the integration of different service mapping algorithms.
5. NS/VNF Catalogs: NS/VNF repositories including key data for service orchestration.
6. NS/VNF Manager: system managing the complete life-cycle of a VNF/NS instance.
7. Virtualized Infrastructure Manager (VIM): the actor responsible for control and management of compute, storage and network resources. It implements the decision of the mapping algorithm. It further provides the information on resources' occupancy, events notifications, performance, etc., which are fed into the infrastructure repository.

The functional architecture involving such actors and systems is represented in Figure 2. The interested reader may find detailed information about the above NFV architecture in [7].

3. PROBLEM DESCRIPTION AND MODEL

In this section, the service mapping problem is modeled and stated. In the following, $|\cdot|$ denotes the cardinality of a set and $\{x_i\}_{i\in I}$ the set of the elements x_i, for i varying in the set of indices I.

3.1. Problem Modeling

The service mapping problem is modeled based on graph theory; the proposed modeling framework subsumes the key findings in literature and is in line with the ETSI indications [1]. The following *directed* graphs are considered [5]:

- Network Infrastructure Graph. $\mathcal{G}(NI) = \{\mathcal{V}^{NI}, A^{NI}\}$. $\mathcal{V}^{NI}$ and A^{NI} are, respectively, the sets of vertices and edges of the Network Infrastructure (NI) graph. Each vertex represents a Point of Presence (PoP) while the edges stand for the links between the PoPs. A network PoP [2] is the location where a NS is implemented[‡].
- Network Service Graph. $\mathcal{G}(NS) = \{\mathcal{V}^{NS}, A^{NS}\}$. $\mathcal{V}^{NS}$ and A^{NS} are, respectively, the sets of vertices and edges of the generic NS graph. The vertices and edges represent, respectively, VNFs and the virtual links in between.

The reader may find in the ETSI documents the detailed representation of the above-mentioned NS and VNF forwarding graphs. The above graphs provide a basic representation of the NI and NS topologies; they are further enriched with the information on the current availability of infrastructure resources, and the information on the resources required by the services to be mapped:

- RA_r^{PoP} represents the aggregate availability of resources of type $r \in R$ at $PoP \in \mathcal{V}^{NI}$. R denotes the set of network resource types (e.g., memory, CPU, bandwidth, etc.).
- $RA_r^{(PoP_s, PoP_d)}$ captures the availability of resource of type r on the physical link between $PoP_s \in \mathcal{V}^{NI}$ and $PoP_d \in \mathcal{V}^{NI}$.
- RR_r^{VNF} is the amount of resource of type r required by the $VNF \in \mathcal{V}^{NS}$.

[‡]The terms PoP and Datacentre (DC) are used interchangeably throughout this paper.

DOI: 10.1002/nem

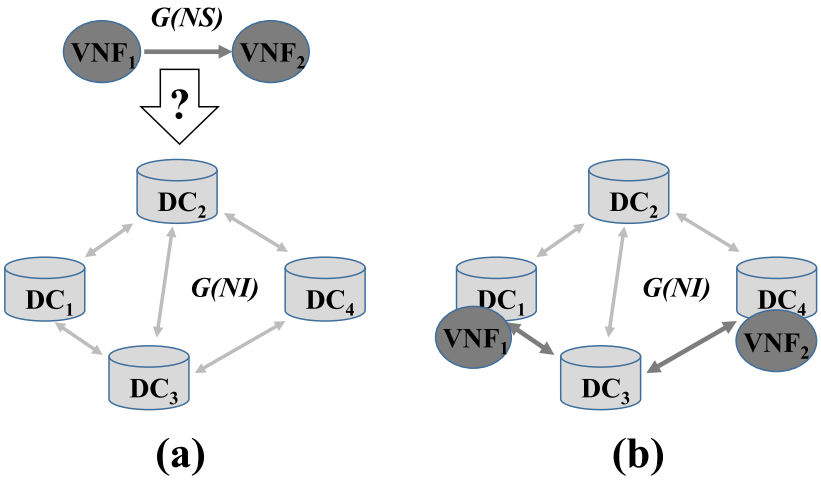

Figure 3. Example of first level service mapping problem (a) and solution (b) (picture from [5]).

- $RR_r^{(VNF_s,VNF_d)}$ is the amount of resource of type r required by the virtual link between $VNF_s \in \mathcal{V}^{NS}$ and $VNF_d \in \mathcal{V}^{NS}$.

Depending on the resource type, the above parameters can be either real numbers (e.g., for memory, CPU, bandwidth requirements) or integers (e.g., to model presence or absence of resources, the number of resources available or required, etc.). The proposed RL-based mapping algorithm will also make use of the knowledge of the type of NS/VNF mapped. Therefore, it is assumed in the following that the NSes and the VNFs belong, respectively, to the set T_{NS} of NS types, and to the set T_{VNF} of VNF types. In practice, T_{NS} and T_{VNF} can be either exactly determined, when their cardinality is small, or designed based on quantization of the relevant NSs/VNFs parameters.

The adopted modeling philosophy is general enough to capture the commonly encountered resource requirements or availability specifications, like maximum tolerated link delay, minimum bandwidth required, constraints on NS endpoints location, etc. The focus of this paper is on the so called *first level service mapping* problem; that is, the problem of mapping the VNFs composing a

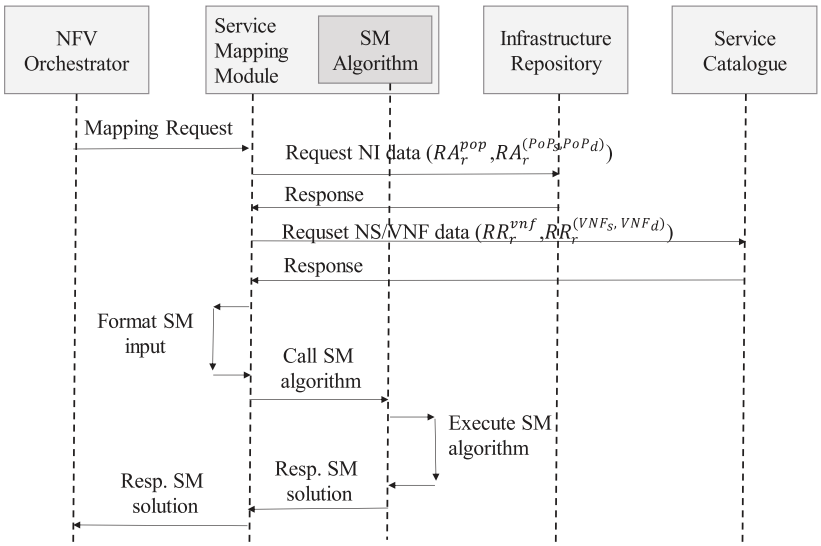

Figure 4. Service mapping sequence diagram (picture from [5]).

DOI: 10.1002/nem

NS to the available PoPs in the network infrastructure (see an example in Figure 3). The sequence diagram in Figure 4 details the basic interaction flows underpinning the provisioning of NS mapping services.

3.2. Problem Statement

Based on the above modeling notation, this section defines the first-level service mapping problem the paper aims to address.

Problem 1 (*First Level Service Mapping*).
Given:

- *a network infrastructure* $\mathcal{G}(NI) = (\mathcal{V}^{NI}, A^{NI})$*, with knowledge of currently available PoP resources* RA_r^{PoP} *and link resources* $RA_r^{(PoP_s, PoP_d)}$*,*
- *a NS instantiation request, with knowledge of: the NS topology* $\mathcal{G}(NS) = \{\mathcal{V}^{NS}, A^{NS}\}$*, the NS node requirements* RR_r^{VNF} *and the NS link requirements* $RR_r^{(VNF_s, VNF_d)}$*,*
- *a service mapping reward criterion, which associates a reward to each mapping outcome,*

find a feasible mapping of the VNFs composing the NS to the infrastructure PoPs, and a feasible mapping of the NS virtual links to PoP paths, such that all the node and link constraints are satisfied and the mapping reward criterion is optimized.◁

In the next two sections, the proposed RL and ILP formulations of the service mapping problem are detailed. The ILP formulation is aligned with the state-of-the-art exact methods found in literature, and serves here also as a benchmark of the performances achieved by the RL algorithm.

4. RELATED WORK

NFV is being increasingly investigated by the research and the industrial communities. Regarding service mapping, over the past years the attention has been focused on DC networks. Works such as [8], [9] and [10] discuss cloud platform implementations that allow to assign clusters of NSs to DCs taking into account performance guarantees, but without consideration of the functionalities of a service chain. The authors in [11] discuss service composition considering NSes chaining. Similarly, [12] discusses a heuristic mapping algorithm for assigning Virtual Machines (VMs) to servers in a DC. The paper addresses the case of NSes composed by multiple VNFs, pointing out that the typical case addressed in literature regards instead the mapping of single VNFs. Interestingly, [12] discusses the relation of the proposed automated service mapping procedure with the existing cloud management systems, with particular regard to OpenStack [13].

An heuristic solution for the mapping of chained VNFs is proposed in [14] that uses a transformation of network topology graphs into simplified trees to reduce the complexity of the decision problem. From the methodological perspective, the above works are mainly based on heuristic algorithms that seek to optimize DC networks key performance indicators, such as inter-rack traffic.

Other fundamental works (see e.g., [9]) address the problem of NS modeling, with the aim of solving the inefficiencies implied by the previously adopted models, such as the hose, VOC (Virtual Oversubscribed Cluster) and pipe models [9]. The technique developed in [9], named TAG (Tenant Application Graph), avoids the overprovisioning inefficiencies stemming from the use of the other modeling techniques by accurately capturing the bandwidth requirements for the VMs to

DOI: 10.1002/nem

be deployed. The TAG is a graph-based model in which VMs, or tiers of VMs, are associated with the nodes of the graph, and VM ingress and egress link requirements (e.g., bandwidth requirements) are modeled by directed edges in the graph. The NS models used in this paper are an extension of the TAG model. Subsequent works have started to investigate the problem of mapping NSes (i.e., chains of VNFs) in a wide area context, considering the infrastructure resources available in a network of DC with multiple PoPs.

In the recent work [15], Riggio et al. introduce an algorithm for mapping VNF chains to a virtualised network infrastructure, focusing on the case of WLANs. The paper proposes a general NS modeling framework, compliant with the ETSI modeling approach [15]. The paper address both the problems of VNF node mapping to substrate network nodes and VNF virtual link mapping to network paths. The mapping strategy proposed in [15] sequentially visits the VNFs to be mapped, starting from the one with maximum connectivity degree. Each visited VNF is then mapped to the network nodes which are highest on a cost metric scale which accounts for the node residual resources available, and the congestion level of the path connecting the network node with the one hosting the previously mapped parent VNFs. The VNF virtual link mapping problem is solved based on a shortest path computation. Results presented in [15] report performance metrics widely used in the literature, such as NS acceptance rates and node/link utilization levels. Reference [16] presents another interesting and advanced contribution. The authors propose a service mapping algorithm based on ILP, in which the NSes to be mapped and the network infrastructure are modeled as undirected graphs, with specification of topology and node/link resource availability and requirements. The paper also investigates the so called "lookahead" mapping feature (previously introduced by the references in [16]), according to which more NSs are embedded at the same time. The efficiency of the solution increases as the number of services simultaneously mapped increases (more service requests can be accommodated and, at the same time, network resources are better exploited, at the expenses of increased computation times).

Dealing with ILP methods, [17] discusses an advanced ILP service mapping algorithm for multi-domain service mapping, focusing on the problem of limited information disclosure among service providers and infrastructure operators. MIDAS [18] proposes an architecture for the coordination of on-path flow processing setup, assuming the wide-scale deployment of middleboxes in the network. MIDAS relies on Multi-Party Computation to partition NSes among multiple providers in a distributed manner. References [19] and [20] also propose other ILP formulations, the first utilizing also the Gomory-Hu Tree Transformation on the substrate network graph to reduce the complexity of the problem and the latter takes also into account, in the cost function, the reliability of the network infrastructure. In [20] the authors also propose a game-theory based heuristic algorithm that shows higher scalability and differentiates itself from other works in literature also for the fact that it first maps the virtual links and only after deals with the nodes. References [21] and [22] discuss the case in which a NS chain consisting of several VNFs can be realized in different ways (the process of choosing the actual implementation "shape" for the requested NS being referred to as "service decomposition"). The authors propose a mapping algorithm which also integrates a service decomposition phase, allowing to select the most suitable service configuration at run time. A heuristic and an ILP model for solving the problem are presented and discussed. Another contribution that deals with the topology of services is reference [23], where the authors propose an heuristic solution to the problem of joint design and mapping of chains of VNFs, minimizing

DOI: 10.1002/nem

the total consumed bandwidth. Reference [24] presents a context-free language to build a model of NS chaining requests. A mixed integer quadratic programming mapping strategy is also presented, and three different objective functions are evaluated using Pareto techniques: (i) maximization of available data rate on substrate network links, (ii) minimization of used infrastructure nodes and (iii) minimization of the path latencies.

Soft computing optimization techniques have been proposed as well. The early work [25] provides an interesting example, proposing a mapping algorithm based on binary PSO (Particle Swarm Optimization). Five different target functions are proposed (the relevant ones for the following discussion being: minimization of used network nodes, minimization of used network links, minimization of the mapping cost). Virtual link mapping relies on shortest path computation, with the cost matrix being given by the available link resources. The interested reader is referred to references in [25] for a discussion of other soft computing, approximate, evolutionary optimization techniques which have been applied to the service mapping problem.

Other interesting contributions are [26], which proposes a decentralized auction algorithm based on consensus and highlights the advantages of such method with respect to centralized control schemes, and [27] where two heuristic algorithms are presented to map VNFs robustly against faults or disasters that may affect the network topology.

Reference [28], one of the recent works on service mapping, presents a modular virtualization architecture that enables policy-based management of VNFs. An information model is introduced to describe and abstract network resources and VNFs. The paper reports also an interesting discussion about the possible synergies between NFV architectures and Software-Defined Networks (SDN). A simple shortest path virtual link mapping algorithm is proposed and tested in a small-scale testbed, together with the proposed policy based VNF orchestrating framework. The authors show that including a rule-based framework allows to orchestrate the VNFs mapping in such a way as to preserve the SLAs defined for the different clients.

Beyond the above-described works on service mapping, the interest reader is referred to [29] and [30] for an additional discussion of consolidated state of the art.

5. SERVICE MAPPING BASED ON REINFORCEMENT LEARNING

This section presents a formulation of the service mapping problem based on Markov decision process theory. Then, based on such reformulation, the proposed RL algorithm is presented.

5.1. Markov Decision Process Modelling of the Service Mapping Problem

The service mapping control problem can be reformulated as a Markov decision process. The reference control logic is illustrated in Figure 5. At each time k a NS instantiation request is made, the service mapping module plays the role of the controller, and acts based on a mapping policy $\pi(s(k))$, which decides the mapping action $a(k)$ based on the current state of the system $s(k)$ and the received service request $w(k)$. As a result of the control action, a mapping reward $r(k)$ is received, modeling the mapping success or failure, and the system transits to a new state $s(k+1)$.

The design of the proposed RL algorithm is based on Markov decision process modeling of the service mapping problem. A Markov decision process is a tuple $\{S, A, T, r\}$, in which S defines a discrete and finite state space, A defines the discrete and finite set of possible mapping

DOI: 10.1002/nem

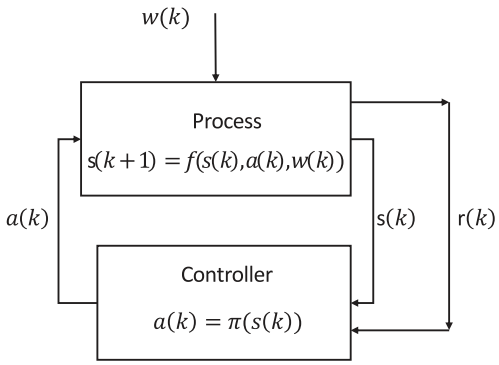

Figure 5. High level control scheme for the proposed service mapping algorithm (picture from [5]).

actions, T is the matrix of the state space transition probabilities and r is a reward function. Dynamic programming methods can be used to solve the Markov decision process by finding an optimal policy which maximizes the cumulative mapping reward in the long run. Since the dynamic programming approach cannot be implemented in a real scenario, due to scalability limits (the well-known "curse of dimensionality" [31]), this paper proposes an on-line strategy for the iterative derivation of a sub-optimal mapping policy by means of a RL algorithm.

Each element of the tuple is defined in the following subsections, focusing on the atomic action of mapping a single VNF to a PoP. In Section 5.2, the complete service mapping RL algorithm will be detailed, which fulfills the mapping of all the nodes and links of the NS to be instantiated.

5.1.1. State Space S Giving a state space formulation for Problem 1 implies accepting a trade-off between modeling accuracy and scalability of the mapping algorithm. In the following, an aggregate formulation for the state space will be considered, to let the algorithm scale to realistic scenarios. The state space S is chosen as a quantized representation of the occupancy level of the infrastructure resources, with the addition of information on the NS that is going to be mapped next. Let us denote with s_i^j the quantized occupancy level of the resource of type j in PoP i (e.g. 10%, 20%, etc.). The state of resources occupancy at the i-th PoP is thus captured by the vector $s_i = [s_i^1, s_i^2, , s_i^{|R|}]^T$ ($|R|$ is the number of different resource types). Hence, the overall state space S will be given by the collection of all the possible above resources occupancy vectors and, as said, the information on the NS being mapped.

$$S = \{s_i, NS, i = 1, ..., |\mathcal{V}^{NI}|, NS \in T_{NS}\} \tag{1}$$

It is assumed that, at each mapping time k, the controller avails of a measurement of the state of the system $s(k)$. Such feedback information is available in practice through the infrastructure repository, which makes possible to measure the occupancy level of the machines inside the PoPs. The knowledge of $s(k)$ is fundamental to understand whether a feasible solution to the mapping problem exists or not, and to have a feedback on the outcome of the mapping actions. The number of occupancy levels considered should take in to account the number of PoPs and resources considered in the scenario, as the number of states is equal to $L^{|R|\cdot|\mathcal{V}^{NI}|} \cdot T_{NS}$, where L is the number of quantization levels, $|R|$ is the number of resource types and T_{NS} is the number of NS types. The informative power of the state decreases as the number of occupancy levels is lowered, but its most important information, which is the identification of the almost saturated PoPs where the VNFs should in general not be mapped, is preserved even with only two levels and a high quantization

DOI: 10.1002/nem

threshold. In this case the state elements would tell if the corresponding PoP is close to the saturation of its resources or not. A higher number of levels could lead to marginally better performances but will increase both the spatial and the temporal complexity of the problem, as it will be clarified in the following.

5.1.2. Action Space A The action space defines the set of possible mapping decisions for the generic VNF composing the service. Since the algorithm deals with first level service mapping, possible actions consist in the choice of one of the $|\mathcal{V}^{NI}|$ PoPs where to map the service. Let us introduce the Boolean decision variable $a_i = \{0, 1\}$, which is equal to one if and only if the current NS is to be mapped at PoP i. Then the action space will be given by the collection of the above decision variables.

$$A = \{a_i, i = 1, ..., |\mathcal{V}^{NI}|\} \tag{2}$$

The mapping actions are decided based on a stochastic, iteratively learned policy (conversely to the ILP algorithm, which provides a deterministic approach to service mapping).

5.1.3. Transition Matrix T The generic entry $t(s, a, s')$ of the transition matrix T defines the probability of a system transition from state s to state s', when action a is taken. Even by assuming typical stochastic distributions for the service arrivals and terminations (e.g., Poisson arrival rates and exponential dwelling times), it is not easy to derive an exact transition matrix for the problem in question (see e.g. an example of derivation of T in [32]), since an aggregated state space formulation has been chosen to favor the scalability of the algorithm. On the one hand, this precludes the possibility of computing explicitly the optimal policy based on the Bellman iteration [33], on the other hand, the proposed mapping algorithm will make use of model-free learning techniques which do not rely on the knowledge of T. Also, the computation of the optimal policy would be infeasible for the realistic scenarios dealt with in this paper.

5.1.4. Reward Function r The reward function r specifies the reward deriving from each mapping action. Several formulations of the reward function can be chosen to steer the controlled system towards the desired performance (e.g., costs minimization, reward maximization, load balancing, maximization of the acceptance rate, etc.). A basic and sufficiently general formulation is given in the following, in which the mapping reward for a VNF depends on the mapping outcome (acceptance, if the proposed mapping leads to a feasible network configuration, rejection otherwise) and on the VNF/NS type ($t_{VNF} \in T_{VNF}$, $t_{NS} \in T_{NS}$, to model possible different rewards associated to different services).

$$r_{k+1} = \begin{cases} r(t_{VNF}, t_{NS}) & \text{if the mapping succeeded} \quad (3a) \\ 0 & \text{otherwise} \quad (3b) \end{cases}$$

Different choices for r can be made, for instance: by associating a unitary reward to each NS mapping, the modeled problem would be the one of maximizing the overall cumulative number of services allocated; by assigning different rewards to each NS type, a priority order for the allocation of the various NSes can be enforced, not necessarily linked or proportional to their resource requirements.

DOI: 10.1002/nem

5.2. *Proposed Service Mapping Algorithm Based on Reinforcement Learning*

The proposed RL algorithm is sequential, in the sense that the VNFs and their virtual links are sequentially mapped. No specific visiting order is considered in this paper, since currently available NS are characterized by very simple topologies (investigating improved visiting orders, as in [15], will be considered in the future to further increase performances). The core of the algorithm is a VNF stochastic mapping policy, detailed below, which uses RL to solve the Markov decision problem described in the previous sections. In particular, the standard Q-Learning [33] algorithm is used to iteratively estimate the state-action value function $Q_\pi(s,a)$, that is, the *value function* that expresses the future expected cumulative reward achieved by the system when starting from state s, performing action a, and following policy π thereafter. Based on the knowledge of Q_π, the following ϵ-greedy policy is considered, as customary in RL applications [33].

$$\pi(s,a) = \begin{cases} 1-\epsilon & \text{if } a = argmax\{Q_\pi(s,a)\} \quad (4a) \\ \epsilon/(|A|-1) & \text{otherwise} \quad (4b) \end{cases}$$

where $\pi(s,a)$ represents the probability of taking mapping action a when in state s. According to (4a), at the generic mapping time, the controller will select with probability $1-\epsilon$ the action yielding the maximum expected reward, according to the current value of Q_π (i.e., $a = argmax\{Q_\pi(s,a)\}$); otherwise, it will select a random action. The inclusion of some randomness through the parameter ϵ allows for exploration and increased speed in the learning phase [33]. In the following, a dynamic $\epsilon-$greedy policy is considered, in the sense that the randomness in the action selection is progressively lowered as the number of iterations goes up. This behavior models the need, intrinsic in RL, to balance the exploitation of the knowledge obtained from the past experience, represented by the Q_π matrix, and the exploration of new state-action pairs, which is crucial especially in the first iterations when the agent is still not familiar with the problem. The speed of decaying for this parameter should be inversely proportional to the size of the Q_π matrix, and should be tuned accordingly.

The Q action-value function is iteratively updated according to the Q-Learning strategy, as mapping events arrive.

$$Q_\pi(s_k,a_k) \leftarrow Q_\pi(s_k,a_k) + \alpha_k[r_{k+1} + \gamma \max_a Q_\pi(s_{k+1},a) - Q_\pi(s_k,a_k)] \quad (5)$$

In (5), k is the discrete time, α_k is the so called *learning rate* at time k and $\gamma \in [0,1]$ is a *discount factor*, weighting the current rewards versus the future ones representing how much of the current reward the agent is supposed to risk for an higher future expected reward. The parameter α_k represents how the current reward has to be weighted with respect to the previous experiences, stored in the previous value of $Q_\pi(s,a)$ before the update. To guarantee the convergence of the algorithm this parameter should be chosen in such a way that $\sum_k \alpha_k = \infty$ and $\sum_k \alpha_k^2 < \infty$ [34]. To ensure those two conditions, while guaranteeing an high learning rate for the first iterations, where the agent is still exploring the state-action pairs, the sequence of α_k has been chosen as $\alpha(k) = \alpha_0/(1+\lfloor k/T \rfloor)$. The symbol $\lfloor \cdot \rfloor$ denotes rounding to the lower integer while the parameter α_0 represents the starting value of the sequence, which should be chosen close to one and is lowered every T iterations, where T is proportional to the dimension of the matrix Q_π. The above mentioned

DOI: 10.1002/nem

convergence condition for α_k is respected as it can be shown that $\sum_k \alpha_k^2 = T * \zeta(2) = T * \pi^2/6$, where ζ is Riemann's Zeta function.

Equation (5) can be seen as a feedback rule in which α_k is a gain factor and $[r_{k+1} + \gamma \max_a Q_\pi(s_{k+1}, a) - Q_\pi(s_k, a_k)]$ is the error between a mixed observed/estimated state-action value for the pair (s_k, a_k), i.e. $r_{k+1} + \gamma \max_a Q_\pi(s_{k+1}, a)$, and the last available value of Q_π in such pair, $Q_\pi(s_k, a_k)$.

The above laws (4a-4b) and (5) together define the VNF mapping algorithm adopted. The complete NS mapping algorithm is derived by including a virtual link mapping phase, as follows. Let VNF_i denote the generic i-th VNF to be mapped at current time. Let PA_{VNF_i} and CH_{VNF_i} denote, respectively, the set of parent and children[§] VNFs of VNF_i that have already been mapped before VNF_i. Similarly, let $PA^{link}_{VNF_i}$ and $CH^{link}_{VNF_i}$ denote, respectively, the sets of links connecting the nodes in PA_{VNF_i} and CH_{VNF_i} to VNF_i. Given the above, the NS mapping algorithm has to map the links $PA^{link}_{VNF_i}$ and $CH^{link}_{VNF_i}$ to feasible paths in the infrastructure graph, linking the PoP in which VNF_i has been mapped to the PoPs where the VNFs in PA_{VNF_i} and CH_{VNF_i}, respectively, have been previously mapped. Given the generic link $l \in \{PA^{link}_{VNF_i} \cup CH^{link}_{VNF_i}\}$, the link mapping strategy considered in this work is based on a weighted shortest path computation between the source VNF and the destination VNF connected by the virtual link l. For the shortest path computation, a weighted adjacency matrix W is considered in order to favor balancing of link resources, as follows:

$$W = \{w_{ij} = a^{NI}_{ij} \sum_r OL_r^{(PoP_i, PoP_j)^2}\}_{(i,j) \in A^{NI}} \tag{6}$$

where $OL_r^{(PoP_i, PoP_j)}$ is the % occupancy level of resource r at physical link between PoP_i and PoP_j, $a^{NI}_{ij} \in \{0, 1\}$ is the (i, j) entry of the adjacency matrix of the network infrastructure graph. OL is updated at each mapping event based on the information from the infrastructure repository.

The proposed complete NS mapping algorithm is summarized below.

Algorithm 1 (*Service Mapping Based on Reinforcement Learning*).

Inputs: $\mathcal{G}(NI)$, $\mathcal{G}(NS)$, RR_r^{VNF}, $RR_r^{(VNF_s, VNF_d)}$, RA_r^{PoP}, $RA_r^{(PoP_s, PoP_d)}$ (infrastructure and service parameters; measurement of current infrastructure occupancy level).

Outputs: Mapping of each VNF of the NS to a PoP; mapping of each VNF link to a path between the chosen PoPs.

Procedure:

1: Initialize $Q_\pi(s, a)$.
2: Initialize *cumulativeReward* to 0.
3: Initialize *failureFlag* to *false*.
4: Wait to receive incoming NS activation request.
5: Retrieve NS data: $\mathcal{G}(NS)$, RR_r^{VNF} and $RR_r^{(VNF_s, VNF_d)}$.
6: Determine current state based on (1) and NI measures (i.e. RA_r^{PoP} and $RA_r^{(PoP_s, PoP_d)}$).
7: **for** $i = 1$ to $|\mathcal{V}^{NS}|$ **do**
8: **if** (VNF_i is an endpoint) **then**
9: Compute mapping of VNF_i according to customer request, if given.
10: **else**

[§]Given two linked VNFs in a generic NS, say VNF_i and VNF_j, VNF_i is called parent of VNF_j if VNF_i has an outgoing link towards VNF_j. Similarly, VNF_j is called children of VNF_i.

DOI: 10.1002/nem

11: Compute mapping of VNF_i based on Q-Learning policy (4a-4b).
12: **endif**
13: Check feasibility of resulting PoP resource usage (see (8e) in Section 6).
14: Compute mapping of ingress and egress virtual links of VNF_i based on weighted shortest path - see (6).
15: Check feasibility of resulting physical link resource usage and link delay (see respectively (8d) and (8c) in Section 6).
16: **if** (link feasibility and PoP feasibility) **then**
17: Set positive reward r for processed VNF.
18: **else**
19: Set reward $r = 0$ (failure) and set *failureFlag* to *true*.
20: **endif**
21: Update the state through (1), based on *failureFlag* and processed VNFs.[¶]
22: Update $Q_\pi(s, a)$ function based on (5).
23: **if** (*failureFlag* is *true*) **then**
24: **return** mapping failure notification and **goto** Step 3
25: **endif**
26: **endfor**
27: **if** (*failureFlag* is *false*) **then** [‖]
28: Set positive reward r for processed NS.
29: Update $Q_\pi(s, a)$ function based on (5).
30: Add r to the *cumulativeReward* archived by the agent. [**]
31: **endif**
32: **return** mapping solution and **goto** Step 3

In conclusion, notice that the SM algorithm is only in charge of computing a mapping solution, if one exists, while then they are the infrastructure manager entities that actually implement the mapping solution found by the algorithm. Hence, if the algorithm fails in computing the mapping of one of the VNFs, the SM algorithm returns a mapping failure notification in output (Step 23), but no VNF has to be deallocated from the infrastructure (since none was allocated yet). In this regard, notice also that the state update performed at Step 5 above is an actual *measurement* of the current state of the infrastructure, since it is based on the measurements received from the infrastructure manager (refer also to Fig. 4). On the other hand, the state update in Step 20 is done to take into account the resources that have to be progressively "reserved" for the new VNFs as their mapping is computed by the algorithm.

[¶]If *failureFlag* is true, then the state is rolled back to the original one computed in Step 7 and is taken as the next state in 5 for the Q update. If *failureFlag* is instead false, then the state is updated through (1), based on the state *measured* at step 7 and taking into account the additional resources that have to be reserved for the VNFs processed up to the current time.
[‖]Notice that the Q matrix is updated after both the VNFs and the NS mappings. This is because the aim of the algorithm is to maximize only the reward relative to the NS mappings, but to give a more informative and immediate feedback we also associate a smaller reward, that is not taken in to account in the cumulative reward that is used to evaluate the performances of the agent, for successful VNF mappings and a zero reward to discourage actions that lead to allocation failures.
[**]This cumulative reward represents the sum of all the successful NS mapping rewards and will be used to evaluate and compare the RL and ILP algorithms' performances in the simulations.

DOI: 10.1002/nem

6. SERVICE MAPPING BASED ON INTEGER LINEAR PROGRAMMING

This section details the proposed ILP formulation for the service mapping problem. Two sets of optimization variables are introduced: y_p^h is a binary decision variable equal to one if and only if VNF h is assigned to PoP p, x_{pq}^{hk} is a binary decision variable equal to one if and only if the virtual link (h,k) in graph $\mathcal{G}(NS)$ is mapped onto a physical path of the graph $\mathcal{G}(NI)$ which includes the physical link between PoPs p and q. The ILP formulation can be stated as follows.

Algorithm 2 (*Service Mapping Based on Integer Linear Programming*).

Inputs: $\mathcal{G}(NI)$, $\mathcal{G}(NS)$, RR_r^{VNF}, $RR_r^{(VNF_s,VNF_d)}$, RA_r^{PoP}, $RA_r^{(PoP_s,PoP_d)}$ (infrastructure and service parameters; measurement of current infrastructure occupancy level).

Outputs: Mapping of each VNF of the NS to a PoP; mapping of each VNF link to a path between the chosen PoPs.

Procedure: Find the node and link mapping by minimizing the objective function

$$\alpha \sum_{h\in\mathcal{V}^{NS}} \sum_{p\in\mathcal{V}^{NI}} b_p^h y_p^h + \beta \sum_{r\in R} \sum_{(h,k)\in A^{NS}} \sum_{(p,q)\in A^{NI}} RR_r^{hk} x_{pq}^{hk} \tag{7}$$

subject to the constraints

$$\begin{cases} \sum_{p\in\mathcal{V}^{NI}} y_p^h = 1, & \forall h \in \mathcal{V}^{NS} & (8a)\\ \sum_{(p,q)\in A^{NI}} x_{pq}^{hk} - \sum_{(q,p)\in A^{NI}} x_{qp}^{hk} = y_p^h - y_p^k, & \forall (h,k)\in A^{NS}, \forall p \in \mathcal{V}^{VI} & (8b)\\ \sum_{(p,q)\in A^{NI}} \delta_{pq} x_{pq}^{hk} \le \Delta_{hk}, & \forall (h,k)\in A^{NS} & (8c)\\ \sum_{(h,k)\in A^{NS}} RR_r^{hk} x_{pq}^{hk} \le RA_r^{pq}, & \forall (p,q)\in A^{NI}, \forall r \in R & (8d)\\ \sum_{h\in\mathcal{V}^{NS}} RR_r^{h} y_p^h \le RA_r^{p}, & \forall p \in \mathcal{V}^{NI}, \forall r \in R & (8e)\\ y_p^h \in \{0,1\}, & \forall h \in \mathcal{V}^{NS}, \forall p \in \mathcal{V}^{NI} & (8f)\\ x_{pq}^{hk} \in \{0,1\}, & \forall (h,k)\in A^{NS}, \forall (p,q)\in A^{NI} & (8g)\\ y_{ingressPoP}^{ingressVNF} = 1 & & (8h)\\ y_{egressPoP}^{egressVNF} = 1 & & (8i) \end{cases}$$

◁

Parameter b_p^h in (7) is a measure of the saturation level of PoP p. The objective function (7) is thus a weighted sum of two components: (i) a term that is inversely proportional to the resources available in the various PoPs, (ii) a term that represents the overall link resource usage. The coefficients α and β are the weighting parameters. The presence of the first term discourages the agent to congest the network, which, due to the presence of the random requested end points would degrade the acceptance rates. Constraints (8a) ensure that each VNF h is mapped exactly to one PoP. Conditions (8b) ensure that for a given pair of VNFs h and k assigned respectively to PoPs p and q, the edge (h,k) is mapped to a network infrastructure path connecting p and q. Constraints (8c) ensure that the resulting delays are within the SLA limits. Δ_{hk} represents the maximum tolerated delay for the

DOI: 10.1002/nem

NS#	RR^{VNF}_{CPU}	$RR^{(VNF_s,VNF_d)}_{bandwith}$	Δ_{sd}
NS1	$\begin{bmatrix}60\\54\\24\\42\\42\end{bmatrix}$	$\begin{bmatrix}RR^{(1,2)}_{bandwith}=18\\RR^{(1,3)}_{bandwith}=8\\RR^{(1,4)}_{bandwith}=14\\RR^{(1,5)}_{bandwith}=14\end{bmatrix}$	$\begin{bmatrix}\Delta_{1,2}=151\\\Delta_{1,3}=151\\\Delta_{1,4}=151\\\Delta_{1,5}=151\end{bmatrix}$
NS2	$\begin{bmatrix}144\\72\\135\\54\\81\\81\end{bmatrix}$	$\begin{bmatrix}RR^{(1,2)}_{bandwith}=24\\RR^{(1,3)}_{bandwith}=45\\RR^{(3,4)}_{bandwith}=18\\RR^{(3,5)}_{bandwith}=27\\RR^{(1,6)}_{bandwith}=27\end{bmatrix}$	$\begin{bmatrix}RR^{(1,2)}_{\delta}=151\\RR^{(1,3)}_{\delta}=151\\RR^{(3,4)}_{\delta}=151\\RR^{(3,5)}_{\delta}=151\\RR^{(1,6)}_{\delta}=151\end{bmatrix}$
NS3	$\begin{bmatrix}384\\216\\336\\288\\240\end{bmatrix}$	$\begin{bmatrix}RR^{(1,2)}_{bandwith}=72\\RR^{(1,3)}_{bandwith}=45\\RR^{(1,4)}_{bandwith}=96\\RR^{(1,5)}_{bandwith}=80\end{bmatrix}$	$\begin{bmatrix}RR^{(1,2)}_{\delta}=151\\RR^{(1,3)}_{\delta}=151\\RR^{(1,4)}_{\delta}=151\\RR^{(1,5)}_{\delta}=151\end{bmatrix}$

Table I. Main parameters of the simulated NSes.

link between VNFs h and k, while δ_{pq} represents the delay associated to the link between PoPs p and q. Constraints (8d) guarantee that inter PoP link resource limits are not violated, whereas constraints (8e) account for node resource limits for each PoP. Conditions (8f) and (8g) express the binary domain constraints for the variables used. Conditions (8h) and (8i) model the possibility to request the allocation of the ingress and egress VNFs to specific PoPs.

7. SIMULATION RESULTS

7.1. Simulation Setup

The numerical simulation setup relies on the benchmark set for the virtual network mapping problem available in [35]. The set provides realistic NS and NI topologies to replicate various NFV scenarios, scaling from small/medium applications to very large ones. The methodology used to generate the dataset in [35] is explained in [36]. Since the present paper focuses primarily on presenting a new approach to service mapping and showing experimentally its validity, we have focused on just one of the networks available in the above dataset, which is representative of todays medium NFV applications, both in terms of network and services dimensions (i.e. number of PoPs and number of VNFs, respectively). The test setup is as follows: three NS types are considered, with characteristic parameters as reported in Table I. The three NSes represent, respectively, "light", "medium" and "heavy" NSes in terms of resources requirements. The considered resources are CPU and link bandwidth. Also, for the services selected from [36] it holds $RR^{(i,j)} = RR^{(j,i)}$, but the framework presented here does not require this limitation in general. A constraint on the maximum tolerated delay for each VNF link is included as well. The simulated network infrastructure is constituted by 20 PoPs (see Figure 6). Such network is representative of the complexity of real virtual network mapping problems: reference [36] describes how the network infrastructure was derived from realistic networks using the nem-0.9.6 tool [37]. Also, the chosen NSes are composed

DOI: 10.1002/nem

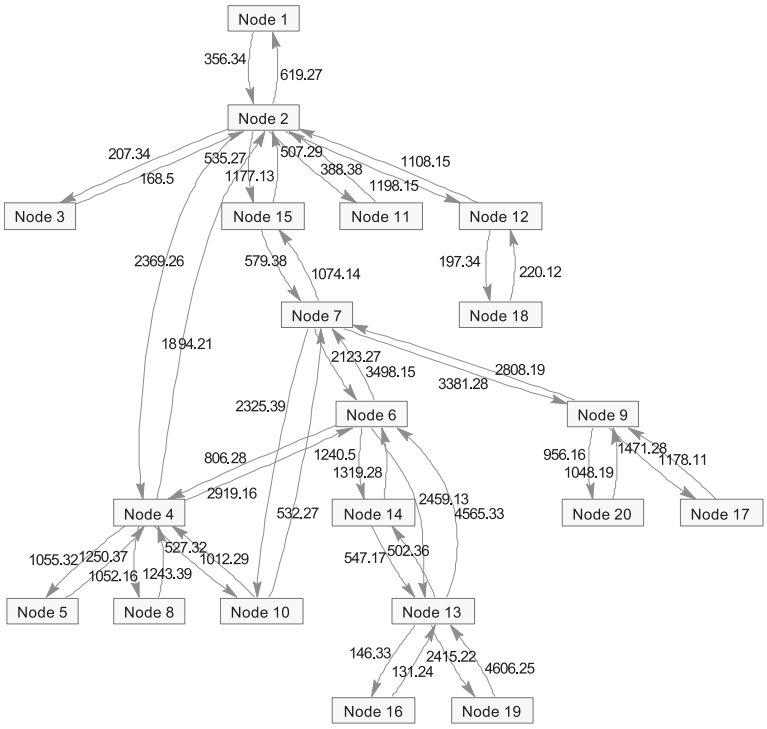

Figure 6. 20-PoP network infrastructure with indication of link bandwidth and delay characteristics (bandwidth and delay values are separated by a point).

by VNFs associated to the following four kind of "slices": Web slice, Stream slice, P2P slice, VoIP slice [36].

The simulation scenario is generated as follows: service arrivals are generated according to a Poisson distribution of diverse arrival rate λ. Services dwell times are generated according to an exponential distribution of parameter μ. Based on Littles law [38], the arrival rate is varied in the simulations in order to vary the service load to the system, referred to as "load factor", LF. Specifically, the load factor is defined as the ratio of the amount of resources that would be occupied by the NSes, at a generic time, in a stationary environment and assuming that all the NSes were accepted (i.e. ignoring mapping constraints), to the total amount of resources available in the empty infrastructure. The load factor is computed for each resource type. It can be thought of as a measure of the average level of demand of a resource type, in a stationary environment (i.e. when the NS arrival rates and dwell times have a stationary distribution). The load factor can be greater than one, meaning that more resources are requested that the ones physically available in the infrastructure (conversely, notice that mapping may fail also when the load factor is lower than one). In the present case, the load factor associated to a given resource type can be computed as $LF_r = \frac{\sum_{i=1}^{|T_{NS}|} RR_r^{NS_i}/(\lambda_i \mu_i)}{\sum_{i \in \mathcal{V}} RA_r^{PoP_i}}$, where T_{NS} is the set of NS types, $RR_r^{NS_i}$ is the total amount of resources of type r required by NS_i and, by definition of the respective distributions, $1/\lambda_i$ is the mean arrival rate of the i-th type of service and $1/\mu_i$ is the mean dwell time. Thus, according to the Little law[††], $RR_r^{NS_i}/(\lambda_i \mu_i)$ is the average demand of resource type r by NSes of type i. For

[††]The Little law (see e.g. [38]) states that in stationary conditions the average number of elements in a system is given by the product of the average elements arrival rate and the average elements dwell time into the system.

DOI: 10.1002/nem

simplicity, λ and μ are assumed equal for all the NS types, μ is kept constant at 0.001 and λ is varied by inverting the above formula for LF, thus allowing to test the algorithm in different desired demand-level scenarios. At each birth time, the type of the NS associated to the service request is determined by random extraction. Also, for each service request, two endpoints are randomly associated to two VNFs and forced to be placed in two randomly selected PoPs, as modeled by constraints (8h) and (8i). The deriving service request is then processed by the proposed Algorithms 1 and 2. At each service request or termination event, the state of the network is consistently updated according to the parameters of the NS in question, and the decision taken by the control algorithm. Each of the simulation results reported below are obtained by compiling the results across many test runs, each spanning over a time horizon of 10000 time steps. Finally, the following RL parameters have been selected: ϵ in (4a-4b) has been set to 0.9, halved every 1000 iterations. In (5) the discount factor γ has been set to 0.89 and kept constant, while the learning rate has been chosen as $\alpha(k) = 0.9/(1 + \lfloor k/5000 \rfloor)$. The choice of the time horizon T is always such that the RL algorithm archives convergence even at the lowest load factors. This is proven by the fact that the acceptance rates of the various services do not show significant differences for values of T higher than 6000 and the cumulative reward increases almost linearly with T.

The state space is quantized selecting for each PoP only two occupancy levels. The above choice of simulation parameters leads to a Q matrix of dimensions $2^{20} \cdot 3 \cdot 20$, which with double precision corresponds to approximately 480MB of memory (in general, the Q matrix has dimensions $L^{|NI|} \cdot |T_{NS}| \cdot |NI|$).

In the following, two simulations are performed, aimed at comparing the performances of the two algorithms when:

1. The maximization of the total service acceptance rate is sought.
2. The maximization of the total mapping reward is sought.

In all the simulations the rewards associated to the whole NS mapping have been chosen as the sum of the rewards associated to the respective VNFs. Finally, the entire simulation scenario has been implemented in Matlab 2011b.

7.2. Maximization of total service acceptance

The objective of this simulation is to compare the performance achieved by the two algorithms when a unitary reward is returned for the successful mapping of a NS, independent of the type of the NS. This scenario is therefore equivalent to seek the maximization of the total allocations, as the cumulative reward can be thought of as the number of allocated services.

Figure 7 reports the total cumulative reward achieved. As expected, for low load factors, rewards increase linearly. After a load factor of 0.8, the total reward growth decreases because of congestion in the network. The decrease is more evident for the ILP algorithm. For the first three values of the load factor, the two algorithms achieve comparable performances, with the ILP algorithm slightly outperforming the RL algorithm. This is because of the exact nature of the ILP algorithm. Then, as the resource consumption in the infrastructure increases, and efficient long-term resource management becomes more critical, the RL algorithm outperforms the ILP one, accepting up to 10% more requests for the highest load factor considered. The same behavior is seen also in the total acceptance rates, as shown in Figure 8.

DOI: 10.1002/nem

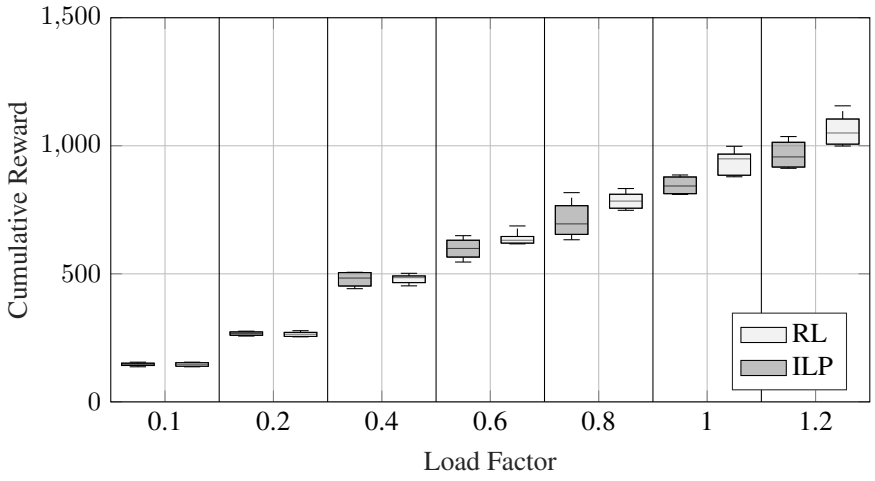

Figure 7. Total cumulative reward (unitary NS rewards).

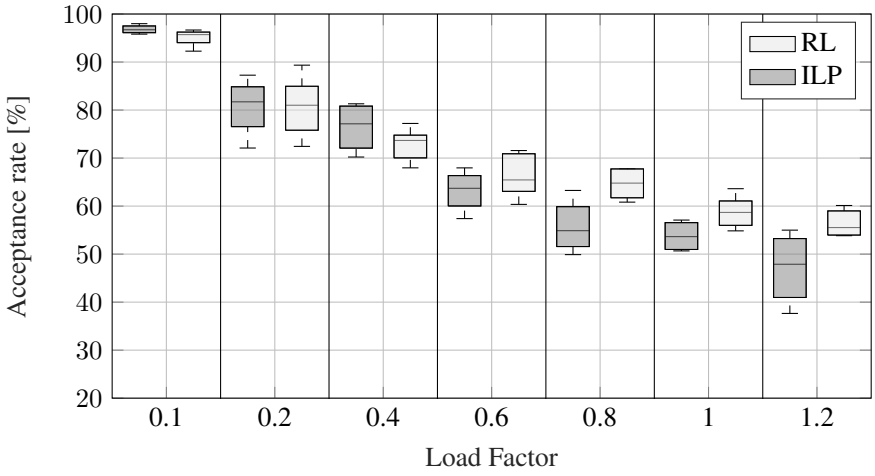

Figure 8. Total acceptance rate (unitary NS rewards).

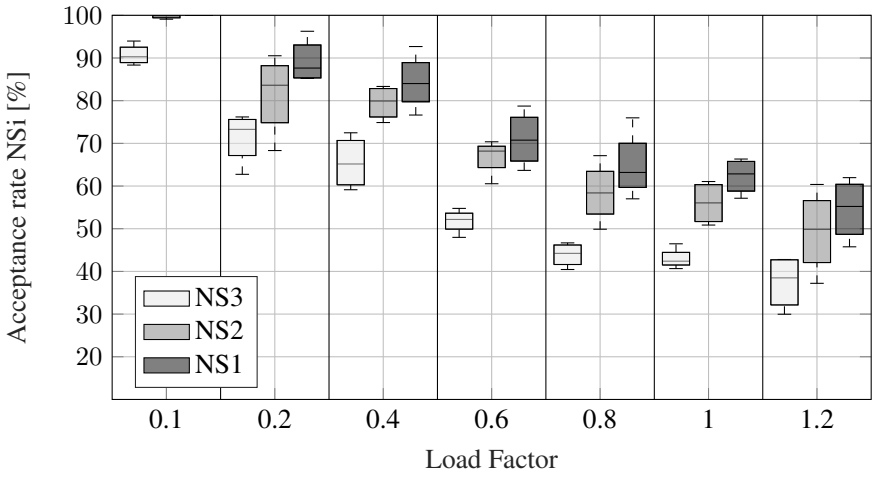

Figure 9. NSes acceptance rates achieved by the ILP algorithm (unitary NS rewards).

The distinct acceptance rates for the three different NS types are reported in the following Figures 9 and 10. As expected, the percentages are decreasing as the load factor increases. Also, the higher

DOI: 10.1002/nem

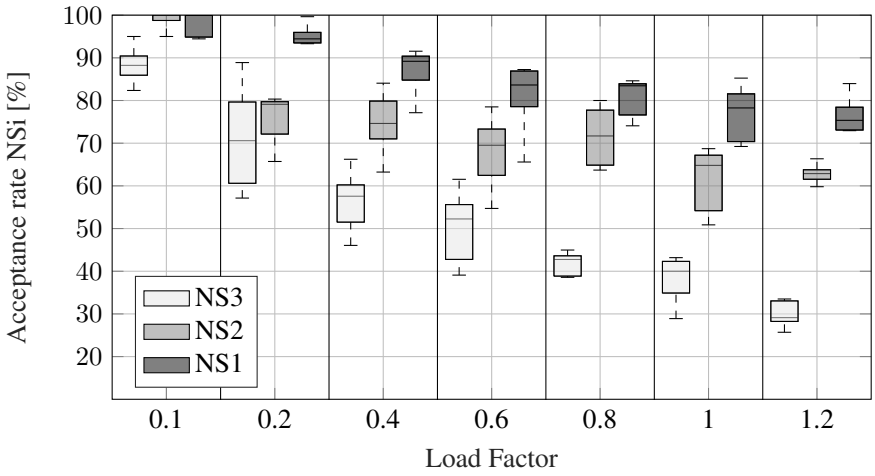

Figure 10. NSes acceptance rates achieved by the RL algorithm (unitary NS rewards).

the resource requirement characterizing a NS type is, the lower is the resulting acceptance rate. We can notice in Figure 9 how the ILP controller decreases the three acceptance rates uniformly, while in Figure 10 it is clear that the proposed RL controller favors the services with lower resource requirements, and penalizes the heaviest ones. This behavior is expected, since the RL controller, in order to archive a maximum reward over the whole time horizon, prefers to reserve resources for the more profitable (in terms of reward/resource ratio) types of NSes. On the other hand, the ILP solution does not take into account the future requests at all, and hence is not able to take a similar decision.

7.3. Maximization of cumulative mapping reward

The objective of this second simulation is to compare the performance achieved by the two algorithms when different rewards are returned for the successful mapping of the three different types of NSes. In particular, the rewards are chosen as 1, 4 and 1 for NS1, NS2 and NS3, respectively, thus making the second type of NS the one with the best reward/resource ratio. The outcome in terms of acceptance rates for the RL controller is reported in Figure 11. It can be noticed that the

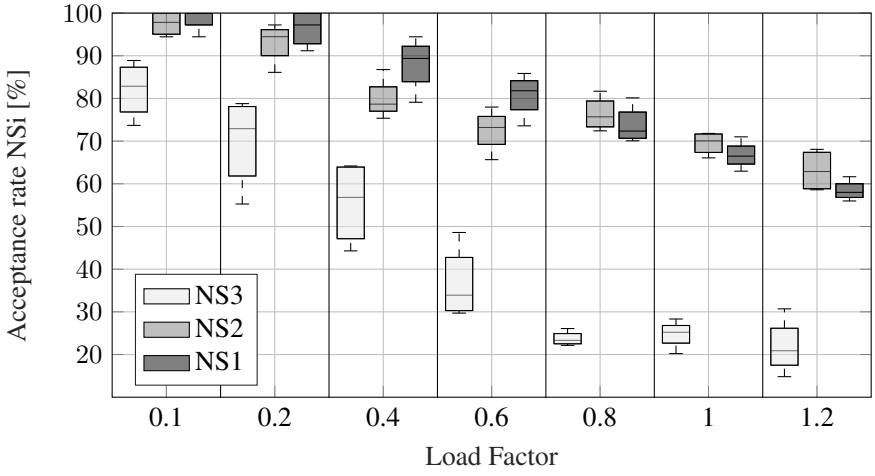

Figure 11. NSes acceptance rates achieved by the RL algorithm (heterogeneous NS rewards).

DOI: 10.1002/nem

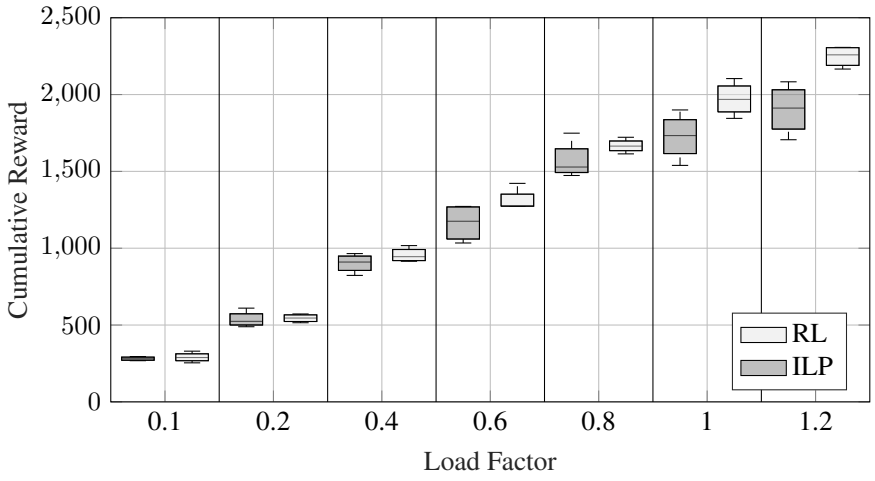

Figure 12. Total cumulative reward (heterogeneous NS rewards).

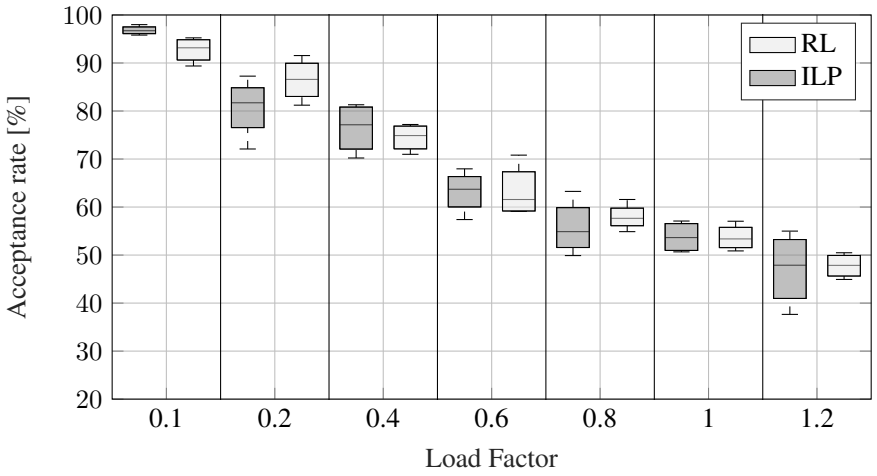

Figure 13. Total acceptance rate (heterogeneous NS rewards).

acceptance rate curve for the RL controller changes, especially in the second half of the figure, where the acceptance rate of NS2 (the NS with highest reward associated) increases at the expenses of the acceptance rate of NS1 and NS3. The acceptance rates achieved by the ILP algorithm do not change significantly from the previous simulation, and are hence omitted. This is again due to the fact that the ILP controller minimizes its target function instant by instant (i.e., request by request) and it is not affected by the different rewards. Considering the cumulative reward shown in Figure 12, in this simulation it can be noticed that the RL algorithm spaces the ILP one faster, and for the highest load factor archives an average reward 18% higher than the ILP-based controller. It is interesting to notice in Figure 13 how the cumulative acceptance rate for the two controllers is very similar, highlighting how the problem this time could not be reduced to the maximization of the overall allocations, but favors the agent that is able to choose which services to allocate more.

8. CONCLUSIONS

This paper has presented a new algorithm for service mapping in virtualized network infrastructures, based on Markov decision process modeling and Reinforcement Learning (RL) theory. A state of

DOI: 10.1002/nem

the art Integer Linear Programming (ILP) formulation has been also presented, for comparison purposes. The proposed formulation of the Markov decision process modeling of the service mapping problem is more scalable compared to the previous formulations in [5] and [6]. Both the formulations allow for the possibility of choosing the reward function most tailored to the control objectives. The simulations highlighted how the proposed RL controller is able to outperform the ILP one, when the long term efficient management of the available resources is more important, such as in the cases of high load factors and in the presence of prioritized services. The strengths of the proposed RL algorithm are: i) the learning ability, which allows to control systems whose dynamics are uncertain or time varying, and ii) the maximization of performances in the long run, appearing as a promising and differentiating feature with respect to most of the works currently present in literature, which are either based on exact optimization algorithms or on heuristics, iii) the fact that each iteration is composed of elementary computations, allowing a potentially higher speed compared to optimization methods. On the other hand, the strengths of the ILP algorithm are: i) it relies on an exact method, implying no learning times; it is very effective in small/medium scenarios, only limited by the computational power on the machine it runs on and the optimization solver used, ii) optimizing instant by instant can still be the optimal choice in certain scenarios, for example in cases where the load factor is low or the resources available are less scarce. Future improvements can be sought in exploring other reinforcement learning techniques, such as different state space aggregations, value function and policy approximation (see e.g. [31], [39]).

Finally, to combine the strengths of the two algorithms, an algorithm based on Model Predictive Control (MPC) is at study, as it allows to achieve the optimization of the system performances in a time windows in the future of the current time.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge all the partners of the T-NOVA project for their cooperation and the anonymous reviewers for their valuable comments.

REFERENCES

1. European Telecommunications Standards Institute (ETSI). ETSI GS NFV 001 v1.1.1 - Network Functions Virtualisation (NFV): Architectural Framework, 1-21, 2013.
2. European Telecommunications Standards Institute (ETSI). ETSI GS NFV 003 V1.2.1 - Network Functions Virtualisation (NFV): Terminology for Main Concepts in NFV, 1-13, 2014.
3. European Telecommunications Standards Institute (ETSI). GS NFV-MAN 001 - V1.1.1 - Network Functions Virtualisation (NFV): Management and Orchestration, 1-184, 2014.
4. T-NOVA Website. http://www.t-nova.eu/. Accessed: 2016-12-12.
5. T-NOVA Consortium. Deliverable D3.3 Service mapping, 1-84, 2016. URL http://www-t-nova.eu/.
6. Riera JF, Batall J, Bonnet J, Das M, McGrath M, Petralia G, Liberati F, Giuseppi A, Pietrabissa A, Ceselli A, *et al.*. TeNOR: Steps towards an orchestration platform for multi-PoP NFV deployment. *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, 2016; 243–250, doi:10.1109/NETSOFT.2016.7502419.
7. T-NOVA Consortium. Deliverable D3.01 Interim Report on Orchestrator Platform Implementation, 1-92 2014.
8. Ballani H, Costa P, Karagiannis T, Rowstron A. Towards predictable datacenter networks. *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, ACM: New York, NY, USA, 2011; 242–253, doi:10.1145/2018436.2018465. URL http://doi.acm.org/10.1145/2018436.2018465.
9. Lee J, Lee M, Popa L, Turner Y, Banerjee S, Sharma P, Stephenson B. Cloudmirror: Application-aware bandwidth reservations in the cloud. *Presented as part of the 5th USENIX Workshop on Hot Topics in Cloud Computing*, USENIX: Berkeley, CA, 2013.

10. Guo C, Lu G, Yang S, Kong C, Sun P, Wu W, Zhang Y, Wang H, Lv G. Secondnet: A data center network virtualization architecture with bandwidth guarantees. *ACM CONEXT 2010*, Association for Computing Machinery, Inc., 2010.
11. Gember A, Grandl R, Anand A, Benson T, Akella A. Stratos: Virtual middleboxes as first-class entities. *UW-Madison TR1771* 2012; :15.
12. Oechsner S, Ripke A. Flexible support of VNF placement functions in openstack. *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, 2015; 1–6, doi:10.1109/NETSOFT.2015.7116178.
13. Openstack website. https://www.openstack.org/. Accessed: 2016-12-12.
14. De Oliveira R, Koslovski GP. A tree-based algorithm for virtual infrastructure allocation with joint virtual machine and network requirements. *International Journal of Network Management* 2016; doi:10.1002/nem.1958.
15. Riggio R, Rasheed T, Narayanan R. Virtual network functions orchestration in enterprise WLANs. *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015; 1220–1225, doi:10.1109/INM.2015.7140470.
16. Guerzoni R, Trivisonno R, Vaishnavi I, Despotovic Z, Hecker A, Beker S, Soldani D. A novel approach to virtual networks embedding for SDN management and orchestration. *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014; 1–7, doi:10.1109/NOMS.2014.6838244.
17. Abujoda A, Papadimitriou P. Midas: Middlebox discovery and selection for on-path flow processing. *2015 7th International Conference on Communication Systems and Networks (COMSNETS)*, 2015; 1–8, doi:10.1109/COMSNETS.2015.7098686.
18. Dietrich D, Rizk A, Papadimitriou P. Multi-domain virtual network embedding with limited information disclosure. *2013 IFIP Networking Conference*, 2013; 1–9.
19. Soualah O, Fajjari I, Hadji M, Aitsaadi N, Zeghlache D. A novel virtual network embedding scheme based on Gomory-Hu tree within cloud's backbone. *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016; 536–542, doi:10.1109/NOMS.2016.7502855.
20. Zhu Y, Xu J, Zhang Q, Wang X, Palacharla P, Ikeuchi T. Game theory based reliable virtual network mapping for cloud infrastructure. *2016 IEEE International Conference on Communications (ICC)*, 2016; 1–7, doi:10.1109/ICC.2016.7511145.
21. Sahhaf S, Tavernier W, Colle D, Pickavet M. Network service chaining with efficient network function mapping based on service decompositions. *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, 2015; 1–5, doi:10.1109/NETSOFT.2015.7116126.
22. Sahhaf S, Tavernier W, Colle D, Pickavet M. Network service chaining with efficient network function mapping based on service decompositions. *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, 2015; 1–5, doi:10.1109/NETSOFT.2015.7116126.
23. Ye Z, Cao X, Wang J, Yu H, Qiao C. Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization. *IEEE Network* May 2016; **30**(3):81–87, doi:10.1109/MNET.2016.7474348.
24. Mehraghdam S, Keller M, Karl H. Specifying and placing chains of virtual network functions. *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*, IEEE, 2014; 7–13.
25. Abedifar V, Eshghi M, Mirjalili S, Mirjalili SM. An optimized virtual network mapping using PSO in cloud computing. *2013 21st Iranian Conference on Electrical Engineering (ICEE)*, 2013; 1–6, doi:10.1109/IranianCEE.2013.6599723.
26. Esposito F, Paola DD, Matta I. On distributed virtual network embedding with guarantees. *IEEE/ACM Transactions on Networking* Feb 2016; **24**(1):569–582, doi:10.1109/TNET.2014.2375826.
27. Wang Y, Liu X, Qiu X, Li W. Prediction-based survivable virtual network mapping against disaster failures. *International Journal of Network Management* 2016; **26**(5):336–354, doi:10.1002/nem.1939.
28. Giotis K, Kryftis Y, Maglaris V. Policy-based orchestration of NFV services in software-defined networks. *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, 2015; 1–5, doi:10.1109/NETSOFT.2015.7116145.
29. Fischer A, Botero JF, Beck MT, de Meer H, Hesselbach X. Virtual network embedding: A survey. *IEEE Communications Surveys Tutorials* Fourth 2013; **15**(4):1888–1906, doi:10.1109/SURV.2013.013013.00155.
30. Mijumbi R, Serrat J, Gorricho JL, Bouten N, De Turck F, Boutaba R. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials* Firstquarter 2016; **18**(1):236–262, doi:10.1109/COMST.2015.2477041.
31. Pietrabissa A. A policy approximation method for the UMTS connection admission control problem modelled as an mdp. *International Journal of Control* 2009; **82**(10):1814–1827, doi:10.1080/00207170902774233.
32. Oddi G, Panfili M, Pietrabissa A, Zuccaro L, Suraci V. A resource allocation algorithm of multi-cloud resources based on markov decision process. *2013 IEEE 5th International Conference on Cloud Computing Technology and*

Science, vol. 1, 2013; 130–135, doi:10.1109/CloudCom.2013.24.
33. Sutton R, Barto A. *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.
34. Melo FS. Convergence of Q-learning: A simple proof. *Institute Of Systems and Robotics, Tech. Rep* 2001; .
35. Algorithms and Complexity Group - TU Wien. Benchmark Set for the Virtual Network Mapping Problem , accessed on 21 April 2017. URL https://www.ac.tuwien.ac.at/research/problem-instances/.
36. Inführ J, Raidl GR. *Introducing the Virtual Network Mapping Problem with Delay, Routing and Location Constraints*. Springer Berlin Heidelberg: Berlin, Heidelberg, 2011; 105–117, doi:10.1007/978-3-642-21527-8_14. URL http://dx.doi.org/10.1007/978-3-642-21527-8_14.
37. Magoni D. Network topology analysis and internet modelling with nem. *International Journal of Computers and Applications* 2005; **27**(4):252–259, doi:10.1080/1206212X.2005.11441778. URL http://www.tandfonline.com/doi/abs/10.1080/1206212X.2005.11441778.
38. Little J, Graves S. Little's law. *Building intuition*. Springer, 2008; 81–100.
39. Pietrabissa A, Delli Priscoli F, Di Giorgio A, Giuseppi A, Panfili M, Suraci V. An approximate dynamic programming approach to resource management in multi-cloud scenarios. *International Journal of Control* 0; **0**(0):1–12, doi:10.1080/00207179.2016.1185802.