# Scaling migrations and replications of Virtual Network Functions based on network traffic forecasting

Francisco Carpio *, Wolfgang Bziuk, Admela Jukan

*Institute of Computer and Network Engineering, Technische Universität Braunschweig, 38106 Braunschweig, Germany*

## ARTICLE INFO

## ABSTRACT

Migration and replication of virtual network functions (VNFs) are well-known mechanisms to face dynamic resource requests in Internet Service Provider (ISP) edge networks. They are not only used to reallocate resources in carrier networks, but in case of excessive traffic churns also to offload VNFs to third party cloud providers. We propose to study how traffic forecasting can help to reduce the number of required migrations and replications when the traffic dynamically changes in the network. We analyze and compare three scenarios for the VNF migrations and replications based on: (i) the current observed traffic demands only, (ii) specific maximum traffic demand value observed in the past, or (iii) predictive traffic values. For the prediction of traffic demand values, we use an LSTM model which is proven to be one of the most accurate methods in time series forecasting problems. Based on the traffic prediction model, we then use a Mixed-Integer Linear Programming (MILP) model as well as a greedy algorithm to solve this optimization problem that considers migrations and replications of VNFs. The results show that LSTM-based traffic prediction can reduce the number of migrations up to 45% when there is enough available resources to allocate replicas, while less cloud-based offloading is required compared to overprovisioning.

## 1. Introduction

Internet Service Providers (ISP) recognize Network Function Virtualization (NFV) as a key concept to reducing capital and operational expenditures. In NFV, service provisioning is achieved by concatenating Virtual Network Functions (VNFs) in a specific sequence order, defined as Service Function Chains (SFCs). The placement of VNFs is a well known problem in the community which can follow different optimization objectives, such as network load balancing and end-to-end delay. Once VNFs are deployed in the network, the dynamic traffic demand patterns require either reallocation or scaling of VNFs to pursuing different objectives. Moreover, part of the workload may need to be migrated to the cloud due to, for instance, non-optimal deployments or insufficient resources within physical servers of the ISP.

The migration and replication of VNFs is a problem widely studied from different perspectives to date. In all studies, when performing migrations in runtime, it was shown that the active flows need to be rerouted causing service disruptions. The use of replications, on the other hand, requires extra server resources, due to virtualization overhead, and extra network resources, due to state synchronization tasks. From an ISP-centric point of view, the use of third party clouds for a possible migration or replication of VNFs has an impact not only on the performance of the system but also on the monetary costs for the

ISP when using third-party cloud services. For these reasons, accurate prediction of future resource utilization or traffic demand values is the key for ISP to better proactively allocate their resources.

We propose to study how traffic forecasting, in general, can help us to reduce the number of migrations and replications in ISPs, as well as the related placements in third-party clouds. We formulate the placement problem as a Mixed-Integer Linear Programming (MILP) model and solve the placement in two phases, the latter one focused on migrations and replications to be able to better understand their effects. We analyze and compare three scenarios for the VNF migrations and replications based on: (i) the current observed traffic demands only, (ii) specific maximum traffic demand value observed in the past, or (iii) predictive traffic values. In the latter case, we specifically use LSTM networks for traffic predictions. The placement model also considers the impact of migrations on the service delays due to service interruptions and the impact of replications on the network and server resource utilization due to virtual machine (VM) overhead and synchronization traffic. Since the MILP model cannot be used as online solution, we propose a greedy algorithm for that purpose and analyze its performance.

---

The rest of the paper is organized as follows. Section 2 presents related work and our contribution. Section 3 describes the reference scenario. Section 4 formulates the optimization model. Section 5 describes the online heuristic approaches. Section 6 analyzes the performance of the model and heuristics and Section 7 concludes the paper.

## 2. Related work and our contribution

### 2.1. VNF placement, migrations and replications

Significant amount of previous work has focused on the placement of virtual resources for VNFs [1], specially with variants of the joint optimization placement problem with different objectives. For instance, in [2], a resource allocation solution is proposed for optimizing energy efficiency, while considering delay, network and server utilization. [3] proposed models to finding the optimal dimensioning and resource allocation with latency constraints in mobile networks. [4] studied how to optimize the VNF placement and traffic routing while considering reliability and end-to-end delays. In [5], the authors propose to solve a joint decision problem when placing VNFs considering multiple real-world aspects in order to deal with highly varying traffic requests. Within the placement problem topic, migration and replications of VNFs are known as specific sub-problems that need to be solved in the context of resource and service management.

Regarding migrations, since VNFs are commonly running over VMs, there is the possibility of migrating VMs entirely [6] or migrating only the internal states of VNFs [7] to new VMs. In this regard, while the interruption and rerouting of active flows is possible [8], there is always a service downtime duration that will vary depending on the path latencies [9]. Some authors, like in [10], propose a dynamic placement scheduler to minimize the end-to-end latencies when performing migrations. In [11], a trade-off was found between the power consumption and QoS degradation to determine whether a migration is appropriate in order to minimize its negative impact due to the service interruptions.

On the other hand, replications have been primarily used to provide service reliability [12,13], whereby minimization of the number of required replicas [14] is one of the main objectives. In addition, replications need to be studied in the context of reduction of end-to-end service delays [15], load balancing on the network links [16] or to load balance the server utilization [17]. Studies combining both migrations and replications have also been carried out, e.g., [18], where a balancing between the number of migrations and replications is proposed in order to maximize the network throughput and minimize the delay. In our previous work [19], we proposed an optimization method to deriving a trade-off between migrations and replications while improving server, network load balancing and QoS. Unlike migrations, replications need to consider the impact on traffic synchronization between VNFs, which is an important issue that adds considerable traffic overhead in the network [20].

### 2.2. Traffic forecasting and VNF resource requirement predictions

While NFV provides network operators more flexibility to instantiate VNFs at runtime, the dynamic change of network states due to the highly variant traffic load at the edge requires prediction mechanisms to proactively adapt the placement of VNFs accordingly. To address this issues, two approaches have been proposed, one by predicting the resources that VNFs will require based on past utilization [21] while the other one by using traffic forecasting (predictions) techniques to calculate how much resources the VNFs will need to serve that traffic correspondingly [22]. In both cases, a more traditional approach uses either the statistical analysis of time series, or machine learning. Examples of the statistical analysis can be found, for instance, in [23] where the authors introduce a mechanism based on Fourier-Series to determine upcoming demands to perform online VNF scaling. In [24],

the authors also use Fourier-Series with the same purpose but, in this case, with the objective of reducing blocking probability. A slightly different approach in this area is proposed in [25] where a method is used based on linear regression to predict traffic and to scale VNFs in order to improve service availability. Yet another example in [26] uses a fractional Brownian motion (fBm) traffic model to learn traffic parameters in order to predict time-varying VNF resource demand.

Most of the recent work in this area, however, include machine learning based methods. In the area of predicting resource requirements, [27] uses Feedforward Neural Networks (FNN) to predict future requirements of VNFs based on its past utilization and the influence from neighbor VNFs. With a similar objective, [28] uses a Bayesian learning approach to learn from historical resource usage data from VNFs and predict future resource reliability. Another example in [29] uses an specific type of Recurrent Neural Network (RNN) which is based on attention and embedding techniques jointly with Long Short Term Memory (LSTM) model to predict CPU utilization from VNFs with high accuracy.

For traffic forecasting with ML, [30] uses both RNN and Deep Neural Networks (DNN) to forecast traffic changes and prove that these methods can improve delay when provisioning new resources to VNFs as compared to threshold-based methods. Since one of the main objectives when predicting traffic is to determine when to scale VNFs, as discussed in [31]. Here, it is proposed the use of Multilayer Perceptron (MLP) to predict the required number of VNFs in relation with the network traffic to scaling the deployment of VNFs.

### 2.3. Our contribution

So far, we lack studies on how traffic prediction can be used to minimize migrations and replications of VNFs. To this end, we contribute with studying how traffic forecasting can help on reducing the number of migration and replication of VNFs by optimizing their placement in a proactive manner. This is motivated especially by three previously mentioned studies, [5,30,31], that showed the need to consider highly varying traffic requests when placing VNFs in 5G networks and the role that traffic forecasting plays in placement and scaling of VNFs. We analyze this problem from an ISP point of view by using a MILP generic multipath based model comparing three scenarios: (i) when VNFs are placed only considering current observed traffic demands, (ii) when VNFs are placed considering the 80% of the specific maximum traffic demand value and (iii) when VNFs are placed considering predicted traffic values. For traffic forecasting, we use an LSTM model which is proven to be one of the most accurate methods in time series forecasting problems. The placement model also considers the impact of migration of VNFs have on the service delays due to service interruptions, considering individual delays per each traffic demand on a per-path basis, i.e., individually per each path. Regarding replications, we consider their impact on the network and server resource utilization due to VM overhead and synchronization traffic used for maintaining states. Additionally, we propose a greedy algorithm as online solution for the MILP model and we compare it to basic random- and first-fit approaches. Finally, we contribute to by showing that traffic prediction can reduce the number of migrations when enough available resources to allocate replicas, while also reducing the utilization of the cloud.

## 3. Reference scenario

We assume that an ISP owns the network infrastructure close to the end users where it install small groups of servers for the NFV Infrastructure. We also assume that the ISP uses the cloud as a third party to offload VNFs when, for instance, its own infrastructure cannot deploy new VNFs. Our model follows a two phase optimization process, in order to study the impact of migrations and replications of VNFs have on the ISP network while minimizing the utilization of the cloud.

### 3.1. Optimization scenarios and assumptions

Since our approach to optimizations is carried out from the point of view of an ISP who owns the physical server infrastructure, given a certain network topology with certain number of servers located in network nodes, we assume that all nodes of that topology have direct links to a third party cloud server. The specific resulting resource utilization from the links connecting to the cloud and the cloud servers are not considered in the analysis, but the geographic location of the cloud servers for service delay is.

The optimization is divided in two phases. During the first one, the model minimizes the placement of VNFs in the cloud, so the ISP network is as much utilized as possible, and also by minimizing the number of VNF replicas at certain time step $t$. After that, a second placement is carried out at time $t + \Delta t$ while considering initial placement of VNFs that took over during the first phase. In this case, minimizing the migration of VNFs from the first placement is also added to the objective altogether with the minimization of replications and cloud VNFs. Since the traffic demands, and, therefore, the amount of resources allocated by VNFs vary over time, during the first phase at time $t$, a certain traffic bandwidth is considered which is different from the one considered during the second phase after $\Delta t$. The main objective is, therefore, to study how migrations and replications can be minimized in the network while at the same time also reducing the usage of the cloud. This is done while comparing three different scenarios when optimizing during the first phase: (i) considering the current observed traffic demands at time $t$, (ii) considering the 80% of the maximum traffic demand values can have and (iii) considering the predicted traffic demands at time $t + \Delta t$.

For the sake of simplicity, we consider a VNF instance maps 1:1 to a VM where some server resources are reserved to the VM independently of the processed traffic. We define the end-to-end service delay, as the sum of propagation delay (time for the data to travel through the fiber), processing delay (time for the VNF to process the data) and service interruption delays caused by migrations. These delays will be explained in detail in the next section, however, let us shortly focus on the migration process in order to better understand its impact on the service delay. We assume a migration occurs when a VNF is reallocated into a new location and, still, there are active flows being served, causing, in this way, an interruption of the service. So, we omit here the case of cold migrations. Most of the migration process occurs without affecting the perceived delay by the end user since, before performing a migration, a new VNF instance is deployed in a new location and its state is synchronized with the old instance. However, we consider there is always a short interruption of the service until the active flows commute to the new VNF [9]. In this sense, the service delay can be interpreted as a worst case delay. In our model, we consider a multipath based approach where every SFC can use multiple paths, whereby each path can exhibit different delays due to different links and VNFs are traversed. On the other hand, we make use of replications to address scalability but without introducing delays due to the replication process does not stop the service. But, we do consider the synchronization traffic between replicas in order maintain their states synchronized.

### 3.2. Migrations and replications

To better understand the model, let us now illustrate an example (shown in Fig. 1(a)) of an SFC providing service to traffic demands $\lambda 1$ and $\lambda 2$ with two chained VNFs, $v1$ and $v2$ instantiated in server $x1$, at node $n1$, and server $x2$ at node $n2$, respectively. Depending on the functionality, every VNF can be of a different type $t$, however for simplification in this example, we assume all VNFs are of the same type $t$, so they all require the same amount resources. The service delay is calculated as the sum of propagation delays, processing delays and service interruption delays. As an example, assuming $D_l$ is the propagation delay of a link $l$ and $d^{\text{pro}}_{x,v}(\lambda)$ is the processing delay



(a) First placement



(b) Migration during second placement
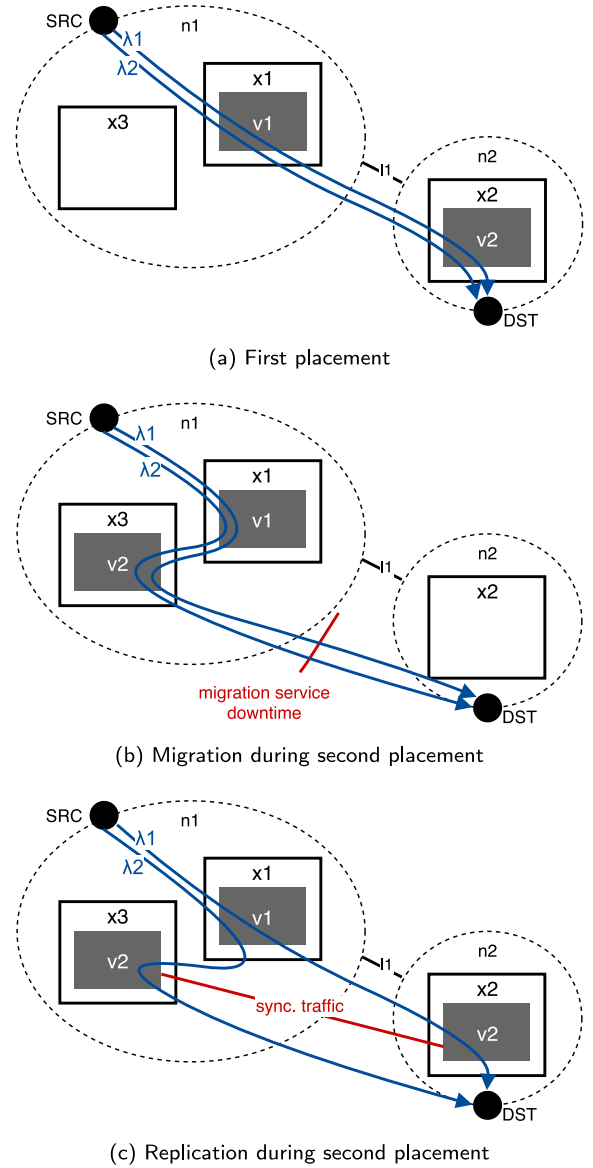


(c) Replication during second placement

**Fig. 1.** Examples of different possible scenarios for the model.

experienced by a traffic demand $\lambda$ traversing a VNF $v$ on a server $x$, then, the delay for traffic demand $\lambda_1$ using that specific path $p$ is $\hat{d}^{\lambda 1}_p = D_{l1} + d^{\text{pro}}_{x1,v1}(\lambda 1) + d^{\text{pro}}_{x2,v2}(\lambda 1)$. In this phase, which is taken as the initial placement for the second phase, we do not consider delays caused by service interruptions, since there are no migrations yet.

For the second phase, the traffic demands change, so the current VNFs in the network have the possibility to be either migrated or replicated. An example is shown in Fig. 1(b), where VNF $v2$ is migrated from server $x2$ to server $x3$. From the delay perspective, here the processing delay of VNF $v2$ is calculated considering the server $x3$ instead, and because a service interruption occurred and, therefore, active flows are rerouted, a service downtime delay is added: $\hat{d}^{\lambda 1}_p = D_{l1} + d^{\text{pro}}_{x1,v1}(\lambda 1) + d^{\text{pro}}_{x3,v2}(\lambda 1) + d^{\text{dwt}}_s$. Another example is shown in Fig. 1(c), where instead of migrating, the VNF $v2$ is replicated into server $x3$ and only traffic demand $\lambda 2$ is routed to the new replica location. In this case, there is synchronization traffic added between both VNFs $v2$ to maintain their states synchronized. In terms of delays, there is no change for $\lambda 1$ compared to the first placement and $\lambda 2$ will experience $\hat{d}^{\lambda 2}_p = D_{l1} + d^{\text{pro}}_{x1,v1}(\lambda 2) + d^{\text{pro}}_{x3,v2}(\lambda 2)$, which considers the processing delay of $v2$ on server $x3$. In this case, service downtime is not considered since
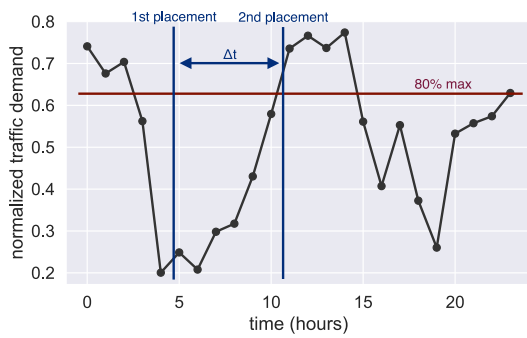
**Fig. 2.** Normalized traffic demand example.

the original VNF is still in original location, so no interruption of the service occurs.

### 3.3. Traffic demand model and time series forecasting

We assume that every source destination pair of nodes within the ISP network generates a certain number of traffic demands with specific bandwidth. The traffic demands data samples are generated using a lognormal distribution with a time-varying mean and variance, which simulates the behavior of common traffic patterns in the internet [32, 33]. The time-varying mean values are obtained using superposition of sinusoidal functions, i.e.:

$$y(t) = \alpha + \sum_{k=1}^{n} \beta_k \cdot \sin(\omega_k^t + \phi_k) , \tag{1}$$

where $\alpha$ is a constant amplitude, $\beta_k$ and $\phi_k$ are frequency dependent constants, and $n$ the number of frequency components, in our case equal to 2. We generate 24 data samples per time period simulating one day. An example of a resulting function is shown in Fig. 2.

The three optimization scenarios previously introduced differ on the considered traffic demand values during the first placement. So, in the first scenario, the VNFs are allocated based on the observed traffic at that specific time step $t$. In the second scenario the VNFs are allocated assuming the demands values are at the 80% of the maximum traffic demand value, instead of considering the real observed values. This let us consider this case as the most conservative one, since an overprovisioning of resources will occur in most of the cases. In the third scenario, the VNFs are allocated considering the predicted traffic demand values after $\Delta t$ instead of the observed ones. Then, the resulting placement from all three scenarios during this first placement is used as initial condition for the optimization of the second placement at $t + \Delta t$, respectively, where in all three cases only the real observed values are considered.

For the last scenario, a time series forecasting problem is modeled where a certain number of time periods (i.e. days, where each day consists on 24 time steps) $D-1$ are used for training and one time period for evaluation. We specifically use one LSTM network for every traffic demand with input and output sizes of 1 unit and 8 units in a hidden layer. The model uses Rectified Linear Unit (ReLU) as the activation function and is fit with Adam optimizer and optimized using the mean squared error (*mse*) loss function. The batch size for the model is 4 and the validation data is 10% of the total. The number of epochs is not constrained, instead an early stopping function is used with a minimum delta of 0.001 and a patience of 10 epochs. Specific parameters are later described during the evaluation of the model.

## 4. Problem formulation

We model the network as $\mathbb{G} = (\mathbb{N} \cup \mathbb{X}, \mathbb{L})$ where $\mathbb{N} = \{1, \dots, N\}$ is a set of nodes, $\mathbb{X} = \{1, \dots, X\}$ is a set of servers and $\mathbb{L} = \{1, \dots, L\}$ is a set

of directed links. Specifically, $\mathbb{X}_n$ is a subset of servers $x \in \mathbb{X}$ attached to node $n \in \mathbb{N}$. We denote the set of all SFCs as $\mathbb{S} = \{1, \dots, S\}$, where a specific SFC $s \in \mathbb{S}$ is an ordered set of VNFs $\mathbb{V}_s = \{1, \dots, V_s\}$, each VNF being of type $t$, $t \in \mathbb{T}$, $\mathbb{T} = \{1, \dots, T\}$, where $v \in \mathbb{V}_s$ is the $v^{\text{th}}$ VNF in set $V_s$. Table 1 summarizes the notations. It should be noted that the model is written such that it can be efficiently used in optimization solvers. For instance, the big M method is avoided when possible or its value is minimized in order to avoid numerical issues with the solver.

### 4.1. Objective function

We define the joint optimization problem as the minimization of the sum of the number of migrations and replications, and number of functions allocated in the cloud, i.e.,

$$minimize: \quad \sum_{s \in \mathbb{S}} \sum_{v \in \mathbb{V}_s} \left[ W_m \sum_{x \in \mathbb{X}} F_x^{v,s}(1 - f_x^{v,s}) \right. \tag{2a}$$

$$+ W_r \left[ \left( \sum_{x \in \mathbb{X}} f_x^{v,s} \right) - 1 \right] \right] \tag{2b}$$

$$+ W_c \sum_{x \in \mathbb{X}_C} \sum_{s \in \mathbb{S}} \sum_{v \in \mathbb{V}_s} f_x^{v,s} , \tag{2c}$$

where the variable $f_x^{v,s}$ specifies if a VNF $v$ from service chain $s$ is allocated in server $x$. Since the optimization process follows two different phases, after the first placement we take the value of variables $f_x^{v,s}$ and convert them into the input parameters $F_x^{v,s}$ for the next placement step, i.e.

$$\forall s \in \mathbb{S}, \forall v \in \mathbb{V}_s, \forall x \in \mathbb{X} : f_x^{v,s} \Rightarrow F_x^{v,s}. \tag{3}$$

The parameter $F_x^{v,s}$ determines if a VNF $v$ of a service chain $s$ was placed on server $x$ during the initial placement. So, the objective function is divided into three different terms, where the first one (2a) counts the number of migrations, the second one (2b) counts the number of replications and the third one (2c) counts the number of functions allocated in cloud servers (here only $\mathbb{X}_C$ subset is considered). Every term is then weighted by $W_m$, $W_r$ and $W_c$, respectively, which allows us to independently evaluate and compare the effect of each one of the terms. The same objective function is used for both optimization phases, with the only difference being that during the first placement, $F_x^{v,s}$ is in all cases 0, so only replications and number of functions in the cloud are minimized. We next follow up with the definition of constraints.

### 4.2. General constraints

The general constraints are related to the traffic routing, the VNF placement and the mapping between VNFs and paths.

#### 4.2.1. Routing

For a given network, the input set $p \in \mathbb{P}_s$ is the set of all pre-calculated paths for SFC $s$. The binary variable $z_p^{\lambda,s} = 1$ indicates, that a traffic demand $\lambda \in \Lambda_s$ of the SFC $s$ is using path $p \in \mathbb{P}_s$. The first routing constraint specifies that each traffic demand $\lambda \in \Lambda_s$ from SFC $s \in \mathbb{S}$ has to use only one path $p \in \mathbb{P}_s$, i.e.:

$$\forall s \in \mathbb{S}, \forall \lambda \in \Lambda_s : \sum_{p \in \mathbb{P}_s} z_p^{\lambda,s} = 1. \tag{4}$$

Then, the next constraint takes the activated paths from the variable $z_p^{\lambda,s}$ and activates the path for a certain SFC $s$:

$$\forall s \in \mathbb{S}, \forall p \in \mathbb{P}_s, \forall \lambda \in \Lambda_s : z_p^{\lambda,s} \leq z_p^s \leq \sum_{\lambda' \in \Lambda_s} z_p^{\lambda',s} . \tag{5}$$

This forces $z_p^s$ to be 1 when at least one traffic demand is using path $p$, whereas the right side forces to $z_p^s$ to be 0 when no traffic demand $\lambda$ is using path $p$.

**Table 1**
Parameters and variables notation.

| Param. | Meaning |
| --- | --- |
| $\mathbb{N}$ | set of nodes: $\mathbb{N} = \{1, \dots, N\}$, $n \in \mathbb{N}$. |
| $\mathbb{X}$ | set of servers: $\mathbb{X} = \{1, \dots, X\}$, $x \in \mathbb{X}$. |
| $\mathbb{L}$ | set of links: $\mathbb{L} = \{1, \dots, L\}$, $\ell \in \mathbb{L}$. |
| $\mathbb{P}$ | set of admissible paths: $\mathbb{P} = \{1, \dots, P\}$, $p \in \mathbb{P}$. |
| $\mathbb{S}$ | set of SFCs: $\mathbb{S} = \{1, \dots, S\}$, $s \in \mathbb{S}$. |
| $\mathbb{T}$ | set of VNF types: $\mathbb{T} = \{1, \dots, T\}$, $t \in \mathbb{T}$. |
| $\mathbb{V}_s$ | ordered set, $v \in \mathbb{V}_s$ is the $v^{\text{th}}$ VNF in set $\mathbb{V}_s$. |
| $\Lambda$ | set of traffic demands: $\Lambda = \{1, \dots, \Lambda\}$, $\lambda \in \Lambda$. |
| $\Lambda_s \subseteq \Lambda$ | subset of traffic demands $\lambda \in \Lambda_s$ for SFC $s \in \mathbb{S}$. |
| $\mathbb{N}_p \subseteq \mathbb{N}$ | subset of ordered nodes in path $p \in \mathbb{P}$. |
| $\mathbb{X}_n \subseteq \mathbb{X}$ | subset of servers attached to node $n \in \mathbb{N}$. |
| $\mathbb{X}_p \subseteq \mathbb{X}$ | subset of ordered servers in path $p \in \mathbb{P}$. |
| $\mathbb{X}_C \subseteq \mathbb{X}$ | subset of servers located at the cloud. |
| $\mathbb{P}_s \subseteq \mathbb{P}$ | subset of admissible paths $p \in \mathbb{P}_s$ for $s \in \mathbb{S}$. |
| $T_p^\ell, T_p^{n,m}$ | binary, 1 if path $p \in \mathbb{P}$ traverses link $\ell \in \mathbb{L}$ and 1 if connects node $n \in \mathbb{N}$ and $m \in \mathbb{N}$ as source and destination path nodes, respectively. |
| $\Gamma_{t(v)}^{\text{pro}}, \Gamma_{t(v)}^{\text{syn}}$ | continuous, load ratio of a VNF of type $t \in V_t$ and traffic ratio for synchronization traffic between two VNFs of type $t \in V_t$, respectively. |
| $\Theta_{t(v)}^s$ | integer, overhead for VNF $v \in \mathbb{V}_s$ of type $t \in V_t$. |
| $C_\ell^{\max}, C_x^{\max}$ | integers, maximum capacity of link $\ell \in \mathbb{L}$ and of server $x \in \mathbb{X}$, respectively. |
| $C_{x,t(v)}^{\text{proq,max}}$ | integer, maximum processing capacity that can be assigned by a server $x$ to a VNF of type $t$ |
| $D_\ell$ | continuous, propagation delay of link $\ell \in \mathbb{L}$. |
| $D_s^{\max}, D^{\text{dwt}}$ | continuous, max. service delay of a SFC $s \in \mathbb{S}$ and service downtime duration caused by a migration, respectively. |
| $D_{t(v)}^{\text{pro,max}}$ | continuous, maximum allowed processing delay for a VNF of type $t$. |
| $D_{t(v)}^{\text{proq}}, D_{t(v)}^{\text{prox}}$ | continuous, delay of a VNF $v$ of type $t$ due to queues and processing, respectively. |
| $R_{t(v)}$ | binary, 1 if VNF $v$ of type $t$ is replicable. |
| $F_x^{v,s}$ | binary, 1 if VNF $v \in \mathbb{V}_s$ from SFC $s$ has been already allocated at server $x \in \mathbb{X}$, 0 otherwise. |

| Vars. | Meaning |
| --- | --- |
| $z_p^s$ | binary, 1 if SFC $s$ uses path $p \in \mathbb{P}_s$. |
| $z_p^{\lambda,s}$ | binary, 1 if traffic demand $\lambda$ from SFC $s$ uses path $p \in \mathbb{P}_s$. |
| $f_x$ | binary, 1 if server $x$ is used, 0 otherwise. |
| $f_x^{v,s}$ | binary, 1 if VNF $v \in \mathbb{V}_s$ from SFC $s$ is allocated at server $x \in \mathbb{X}$, 0 otherwise. |
| $f_{x,\lambda}^{v,s}$ | binary, 1 if VNF $v \in \mathbb{V}_s$ from SFC $s$ is used at server $x \in \mathbb{X}$ by traffic demand $\lambda$, 0 otherwise. |
| $h_p^{v,s}$ | binary, 1 if VNF $v \in \mathbb{V}_s$ from SFC $s$ uses path $p \in \mathbb{P}$ for state synchronization, 0 otherwise. |
| $d_p^{\lambda,s}$ | continuous, service delay of a traffic demand $\lambda$ in path $p$. |
| $u_\ell, u_x$ | continuous, utilization of a link $\ell \in \mathbb{L}$ and server $x \in \mathbb{X}$, respectively. |

### 4.2.2. VNF placement

VNF placement is modeled using the binary variable $f_{x,\lambda}^{v,s}$, which has only value 1, if VNF $v$ from SFC $s$ is allocated at server $x \in \mathbb{X}$ and used by traffic demand $\lambda \in \Lambda_s$. Similar to (4), the next constraint defines that each traffic demand $\lambda \in \Lambda_s$ from SFC $s \in \mathbb{S}$ traverses every VNF $v \in \mathbb{V}_s$ in only one specific server $x \in \mathbb{X}$:

$$\forall s \in \mathbb{S}, \forall v \in \mathbb{V}_s, \forall \lambda \in \Lambda_s : \sum_{x \in \mathbb{X}} f_{x,\lambda}^{v,s} = 1. \tag{6}$$

Then, similarly to (5), the next constraint takes the activated VNFs for each traffic demand from the variable $f_{x,\lambda}^{v,s}$ and activates the VNF for a certain SFC $s$ as follows:

$$\forall s \in \mathbb{S}, \forall v \in \mathbb{V}_s, \forall x \in \mathbb{X}, \forall \lambda \in \Lambda_s :$$
$$f_{x,\lambda}^{v,s} \le f_x^{v,s} \le \sum_{\lambda' \in \Lambda_s} f_{x,\lambda'}^{v,s}, \tag{7}$$

where the left side forces to $f_x^{v,s}$ to be 1 when at least one traffic demand $\lambda \in \Lambda_s$ is using VNF $v \in \mathbb{V}_s$ at server $x \in \mathbb{X}$ and the right side forces to $f_x^{v,s}$ to be 0 when no traffic demand is using that specific VNF $v$ on server $x$. Likewise, we determine if a server is being used or not by constraining the variable $f_x$ as:

$$\forall x \in \mathbb{X} : \frac{1}{|\mathbb{S}||\mathbb{V}_s|} \sum_{s \in \mathbb{S}} \sum_{v \in \mathbb{V}_s} f_x^{v,s} \le f_x \le \sum_{s \in \mathbb{S}} \sum_{v \in \mathbb{V}_s} f_x^{v,s}, \tag{8}$$

where $f_x$ is 1 if at least one VNF from any SFC is allocated at server $x \in \mathbb{X}$, 0 otherwise.

### 4.2.3. Mapping VNFs to paths

The next equation maps the activated VNF to the activated paths defined in the previous constraints. The first one defines how many times a VNF can be replicated:

$$\forall s \in \mathbb{S}, \forall v \in \mathbb{V}_s : \sum_{x \in \mathbb{X}} f_x^{v,s} \le R_{t(v)} \sum_{p \in \mathbb{P}_s} z_p^s + 1 - R_{t(v)}, \tag{9}$$

where $R_{t(v)}$ specifies if a certain VNF $v$ of type $t$ is replicable. When $R_{t(v)}$ is 0, the total number of activated VNFs $v \in \mathbb{V}_s$ from SFC $s \in \mathbb{S}$ is $\sum_{x \in \mathbb{X}} f_x^{v,s} \le 1$. In case the VNF is replicable, then the maximum number of replicas is limited by the total number of activated paths $\sum_{p \in \mathbb{P}_s} z_p^s$ for that specific SFC $s$. The next constraint activates the VNFs on the activated paths:

$$\forall s \in \mathbb{S}, \forall p \in \mathbb{P}_s, \forall \lambda \in \Lambda_s, \forall v \in \mathbb{V}_s : z_p^{\lambda,s} \le \sum_{x \in \mathbb{X}_p} f_{x,\lambda}^{v,s}. \tag{10}$$

If the variable $z_p^{\lambda,s}$ is activated, then every VNF $v \in \mathbb{V}_s$ from SFC $s \in \mathbb{S}$ has to be activated in some server $x \in \mathbb{X}_p$ from the path $p \in \mathbb{P}$ for a specific traffic demand $\lambda$. When $z_p^{\lambda,s}$ is deactivated, then no VNFs can be placed for that specific traffic demand. The last general constraint controls that all VNFs $V_s$ from a specific SFC $s$ are traversed by every traffic demand $\lambda \in \Lambda_s$ in the given order, i.e.:

$$\forall s \in \mathbb{S}, \forall \lambda \in \Lambda_s, \forall p \in \mathbb{P}_s, \forall v \in \mathbb{V}_s, \forall n, m \in \mathbb{N} :$$
$$\left( \sum_{m=1}^n \sum_{y \in \mathbb{X}_m} f_{y,\lambda}^{(v-1),s} \right) - \sum_{x \in \mathbb{X}_n} f_{x,\lambda}^{v,s} \ge z_p^{\lambda,s} - 1 \begin{cases} 1 < v \le |\mathbb{V}_s| \\ n \ne m \end{cases}, \tag{11}$$

where the variable $z_p^{\lambda,s}$ activates the ordering constraint side (left side) when is 1 and deactivates it, otherwise. Then, if path $p \in \mathbb{P}$ is activated, the ordering is checked for every traffic demand $\lambda \in \Lambda_s$ individually by using the variable $f_{x,\lambda}^{v,s}$. Hence, for every traffic demand $\lambda$ of SFC $s$, the $v^{\text{th}}$ VNF is allocated at server $x \in \mathbb{X}_n$ only if the previous $(v-1)^{\text{th}}$ VNF is allocated at any server $y \in \mathbb{X}_m$, where $m$ is the i$^{\text{th}}$ node from 1 until $n$ traversed by path $p$. It should be noted, that the correct sequence of VNFs relies on the correct sequence of subset of servers, i.e. $x \in \mathbb{X}_n$. This assumes that the correct sequence of VNFs inside these subsets is organized by the local routing, which may be located at the node $n$ or at a local switch not modeled in detail.

### 4.3. Traffic and performance constraints

#### 4.3.1. Synchronization traffic

When performing replications of VNFs, the stateful states between the original and replicas has to be maintained in order to be reliable against VNF failures and avoiding the loss of information. For this reason, we consider that when a VNF is replicated, the generated synchronization traffic between replicas and the original has to be also considered. The amount of the state synchronization traffic depends on the state space and its time dynamic, where it is assumed, that each VNF has full knowledge on the state of all its instances used to implement the VNF $v \in \mathbb{V}_s$. Let us assume, that this amount is proportional to the total traffic offered to the SFC weighted by an synchronization ratio $\Gamma_{t(v)}^{\mathrm{syn}}$, which depends on the type of VNF $t$. In summary, the directional traffic from a VNF to its replica is given by $\Gamma_{t(v)}^{\mathrm{syn}} |\Lambda_s|$, and its routing should be optimized within the network.

In order to know if the same VNF $v \in \mathbb{V}_s$ from SFC $s$ is placed in two different servers $x \in \mathbb{X}$ and $y \in \mathbb{X}$, we define:

$$\forall s \in \mathbb{S}, \forall v \in \mathbb{V}_s, \forall x \in \mathbb{X}, \forall y \in \mathbb{X} :$$
$$g_{x,y}^{v,s} = f_x^{v,s} f_y^{v,s}, \quad \text{for } y \neq x , \tag{12}$$

where the variable $g_{x,y}^{v,s}$ is 1 only when both variables $f_x^{v,s}$ and $f_y^{v,s}$ are also 1, and 0 otherwise. In this way, this variable is used to know if two different servers have the same VNF placed, which means that model is allocating one replica. We use the well-known linearization method when multiplying two binary variables. In case $g_{x,y}^{v,s} = 1$, we need to carry the synchronization traffic from server $x$ to $y$, by selecting only one predefined path between them, i.e.:

$$\forall s \in \mathbb{S}, \forall v \in \mathbb{V}_s, \forall n, m \in \mathbb{N}, \forall x \in \mathbb{X}_n, \forall y \in \mathbb{X}_m :$$
$$g_{x,y}^{v,s} \leq \sum_{p \in \mathbb{P}} h_p^{v,s} \cdot T_p^{n,m} \leq 1, \quad \text{for } n \neq m , \tag{13}$$

$$\forall s \in \mathbb{S}, \forall v \in \mathbb{V}_s, \forall n, m \in \mathbb{N} :$$
$$\sum_{p \in \mathbb{P}} h_p^{v,s} \cdot T_p^{n,m} \leq \sum_{x \in \mathbb{X}_n} \sum_{y \in \mathbb{X}_m} g_{x,y}^{v,s}, \quad \text{for } n \neq m , \tag{14}$$

where the constant $T_p^{n,m} = 1$ indicates, that the path $p \in \mathbb{P}$ exists which connects servers $x \in \mathbb{X}_n$ and $y \in X_m$ using the shortest path between nodes $n$ and $m$. The right term of (13) guarantees that only one path $p \in \mathbb{P}$ is selected by variable $h_p^{v,s}$. Moreover, (14) guarantees that this path is only used if at least one $g_{x,y}^{v,s}$ is 1. Note that $h_p^{v,s}$ is a binary variable used for every VNF $v$ of SFC $s$.

#### 4.3.2. Link and server utilization

The utilization of a link is calculated as follows:

$$\forall \ell \in \mathbb{L} : u_\ell = \frac{1}{C_\ell^{\mathrm{max}}} \sum_{s \in \mathbb{S}} \sum_{p \in \mathbb{P}_s} \sum_{\lambda \in \Lambda_s} \lambda \cdot T_p^\ell \cdot z_p^{\lambda,s} +$$
$$\frac{1}{C_\ell^{\mathrm{max}}} \sum_{p \in \mathbb{P}} T_p^\ell \sum_{s \in \mathbb{S}} \sum_{v \in \mathbb{V}_s} \Gamma_{t(v)}^{\mathrm{syn}} \cdot |\Lambda_s| \cdot h_p^{v,s} \leq 1 , \tag{15}$$

where $\lambda \cdot T_p^\ell$ adds the traffic demands from SFC $s \in \mathbb{S}$ when a path $p \in \mathbb{P}_s$ traverses the link $\ell \in \mathbb{L}$. Then, the variable $z_p^{\lambda,s}$ specifies if the traffic demand $\lambda$ from SFC $s$ is using path $p$. The second term is the sum of the extra traffic generated due to the state synchronization between VNFs $v \in \mathbb{V}_s$ from SFC $s$, which is proportional to its total traffic $|\Lambda_s|$ multiplied by the synchronization traffic ratio $\Gamma_{t(v)}^{\mathrm{syn}}$ of the VNF of type $t$. This traffic is only added, if the variable $h_p^{v,s}$ is 1, which indicates that path $p \in \mathbb{P}$ is used for synchronization by a VNF $v$ from SFC $s$, and the link $\ell \in \mathbb{L}$ belongs to this path. Both summation terms are divided by the maximum link capacity $C_\ell^{\mathrm{max}}$ to restrict the utilization.

The processing load of a server is derived as:

$$\gamma_x = \sum_{s \in \mathbb{S}} \sum_{v \in \mathbb{V}_s} \left( \Gamma_{t(v)}^{\mathrm{pro}} \sum_{\lambda \in \Lambda_s} \lambda \cdot f_{x,\lambda}^{v,s} + \Theta_{t(v)}^s \cdot f_x^{v,s} \right) , \tag{16}$$

where the first term sums the traffic $\lambda \in \Lambda_s$ that is using the VNF $v \in \mathbb{V}_s$ from SFC $s \in \mathbb{S}$ at server $x \in \mathbb{X}$, which is determined by the variable

$f_{x,\lambda}^{v,s}$, and multiplied by the processing load ratio $\Gamma_{t(v)}^{\mathrm{pro}}$ of the VNF of type $t$. The second term adds the overhead generated by the VM where the VNF is running and is only added, when the variable $f_x^{v,s}$ determines that this VNF is placed in server $x$. Then, the utilization follows to be given by:

$$\forall x \in \mathbb{X} : u_x = \frac{\gamma_x}{C_x^{\mathrm{max}}} \leq 1 , \tag{17}$$

where $C_x^{\mathrm{max}}$ is the maximum processing capacity.

#### 4.3.3. Service delay

Since every service has a maximum allowed delay $D_s^{\mathrm{max}}$ specified in the SLA agreement, in case of exceeding it, some penalty costs are applied. In our model, and for simplicity, we take into account the propagation delay due to the traversed links, the processing delay that every VNF requires in the servers and, where applicable, the downtime delays caused by the interruption of the service during the migrations of VNFs.

*Processing delay*: The processing delay $d_{x,v,s}^{\mathrm{pro}}$ of a VNF $v$ in a server $x$ depends, on the one side, on the amount of traffic being processed by a specific VNF, described by $d_{x,v,s}^{\mathrm{proq}}$, and on $d_{x,v,s}^{\mathrm{prox}}$, which is related to the VNF type and the total server load $u_x$, given as:

$$\forall s \in \mathbb{S}, \forall v \in \mathbb{V}_s, \forall x \in \mathbb{X}_p : d_{x,v,s}^{\mathrm{pro}} = d_{x,v,s}^{\mathrm{proq}} + d_{x,v,s}^{\mathrm{prox}} , \tag{18a}$$

$$d_{x,v,s}^{\mathrm{proq}} = D_{t(v)}^{\mathrm{proq}} \frac{\Gamma_{t(v)}^{\mathrm{pro}} \cdot \sum_{\lambda \in \Lambda_s} f_{x,\lambda}^{v,s} \cdot \lambda}{C_{x,t(v)}^{\mathrm{proq,max}}} , \tag{18b}$$

$$d_{x,v,s}^{\mathrm{prox}} = D_{t(v)}^{\mathrm{pro\_x,min}} \cdot f_x^{v,s} + D_{t(v)}^{\mathrm{prox}} \cdot u_x . \tag{18c}$$

In (18b), the numerator of $d_{x,v,s}^{\mathrm{proq}}$ determines the total processing load assigned to the VNF of type $t$, which is controlled by the variables $f_{x,\lambda}^{v,s}$. Thus, if the assigned processing load is equal to $C_{x,\lambda}^{\mathrm{proq,max}}$, the VNF adds the processing delay $D_{t(v)}^{\mathrm{proq}}$. The second delay term, given in (18c), adds the load independent minimum delay associated to the usage of a type of this VNF, and a delay part which increases with the server utilization. As a consequence the processing delay $d_{x,v,s}^{\mathrm{pro}}(\vec{\lambda})$ depends on the server $x$, the used VNF type and linearly increases with increasing traffic. Furthermore, the dependency on all traffic demands is denoted by the vector $\vec{\lambda}$, which is omitted for simplicity in (18).

*Downtime duration*: If a VNF $v$ of SFC $s$ has to be migrated, we assume an interruption of the service with duration $D^{\mathrm{dwt}}$. Thus, the total service downtime will consider the migration of all VNFs in that SFC which yields a constraint as follows:

$$\forall s \in \mathbb{S} : d_s^{\mathrm{dwt}} = D^{\mathrm{dwt}} \sum_{x \in \mathbb{X}} \sum_{v \in \mathbb{V}_s} F_x^{v,s} (1 - f_x^{v,s}) , \tag{19}$$

where the parameter $F_x^{v,s}$ determines if a VNF $v$ was placed on server $x$ during the first placement. Thus, if a VNF migrates to another server $y \neq x$, the variable $f_x^{v,s}$ is equal to zero and the service downtime $D^{\mathrm{dwt}}$ has to be taken into account.

*Total delay*: Because the model allows that different traffic demands per service can be assigned to different paths, we define individual end-to-end delay $\hat{d}_p^{\lambda,s}$ for every traffic demand, as follows:

$$\forall s \in \mathbb{S}, \forall \lambda \in \Lambda_s, \forall p \in \mathbb{P}_s :$$
$$\hat{d}_p^{\lambda,s} = \sum_{\ell \in \mathbb{L}} D_\ell \cdot T_p^\ell + \sum_{x \in \mathbb{X}_p} \sum_{v \in \mathbb{V}_s} d_{x,v,s}^{\mathrm{pro}}(\vec{\lambda}) \cdot f_{x,\lambda}^{v,s} + d_s^{\mathrm{dwt}} . \tag{20}$$

The first term is the propagation delay, where $D_\ell$ is the delay of the link $\ell$, and $T_p^\ell$ specifies if the link $\ell$ is traversed by path $p \in \mathbb{P}_s$. The second term adds the processing delays caused by all VNFs from the SFC placed on the servers $x \in \mathbb{X}_p$, in which the variable $f_{x,\lambda}^{v,s}$ has to ensure that the demand $\lambda$ is processed at a specific server $x$. Finally, the third term is the total downtime duration due to the migrations of that service chain. It should be noted that the second term of (20) includes a nonlinear relation between the binary variable $f_{x,\lambda}^{v,s}$ and the

delay variable $d_{x,v,s}^{\mathrm{pro}}$, which also depends on all decision variables $f_{x,\lambda'}^{v',s'}$. To solve that, we introduce a new delay variable $d_{x,\lambda}^{v,s}$, which is bounded as follows:

$$d_{x,v,s}^{\mathrm{pro}} - D_{t(v)}^{\mathrm{pro,max}}(1 - f_{x,\lambda}^{v,s}) \le d_{x,\lambda}^{v,s} \le D_{t(v)}^{\mathrm{pro,max}} \cdot f_{x,\lambda}^{v,s} . \tag{21}$$

If the VNF is selected at server $x$ by $f_{x,\lambda}^{v,s} = 1$, the variable is lower bounded by the exact delay $d_{x,v,s}^{\mathrm{pro}}$ and upper bounded by the maximum VNF delay $D_{t(v)}^{\mathrm{pro,max}}$. Since $d_{x,v,s}^{\mathrm{pro}} \le d_{x,\lambda}^{v,s} \le D_{t(v)}^{\mathrm{pro,max}}$, the specific delay of a VNF can be restricted. If the VNF is not selected, i.e., $f_{x,\lambda}^{v,s} = 0$, the variable has value $d_{x,\lambda}^{v,s} = 0$, since the constant $D_{t(v)}^{\mathrm{pro,max}}$ makes the left size of (21) to be negative. Hence, the end-to-end delay is mapped to an upper and lower bounded variable $d_p^{\lambda,s}$ given as

$$\forall s \in \mathbb{S}, \forall \lambda \in \Lambda_s, \forall p \in \mathbb{P}_s :$$
$$d_p^{\lambda,s} = \sum_{\ell \in \mathbb{L}} D_\ell \cdot T_p^\ell + \sum_{x \in \mathbb{X}_p} \sum_{v \in \mathbb{V}_s} d_{x,\lambda}^{v,s} + d_s^{\mathrm{dwt}} , \tag{22}$$

in which the bounding feature is used in the optimization scenarios described next.

## 5. Online heuristic approaches

Since the model presented is a MILP optimization problem and these models are known to be NP-hard [34], in this section we propose a greedy algorithm to work as an online solution and, First-Fit and Random-Fit algorithms for comparison purposes.

### 5.1. First-fit and random-fit algorithms

Both *First-Fit* (FF) and *Random-Fit* (RF) algorithms are described in Algorithm 1. While both approaches share most of the code, the *FF_RF* parameter specifies whether the code has to run FF or RF. The process starts with a loop where every demand from every SFC is going to be considered (line 1). The first step is to then retrieve all the paths with enough link resources to assign traffic demand $\lambda$ and that also connect both source and destination nodes (line 2). These paths are saved into $\mathbb{P}_s'$, from where one admissible path $p$, first one for FF or a random one for RF, is selected (line 3). In this point, we make sure here that in this path, there are enough server resources to allocate all the VNFs for SFC $s$. Then, from that path, for every VNF $v$ from SFC $s$ (line 4) we start with the process of selecting servers for allocations. First, we retrieve all servers with enough free capacity to allocate the VNF $v$ and to provide service to demand $\lambda$ (line 5), and then we choose the first available server in FF or a random one in RF (line 6). It is to be noted here, that to satisfy VNF ordering (see Eq. (11)), the procedure *chooseServer* will return a valid server from before/after the previous/next VNF allocated. While for the FF case, we assure in line 3 that there will always be a server where to allocate the next VNF in the chain, in RF case we make sure here (line 6) that after the random server selected there is still place to allocate all the rest of the VNFs from the chain in next servers in the path, or we select another server instead. In line 7, we assign the demand and the VNF to the server (i.e. Eqs. (6) and (7)). After all the VNFs have been placed, the next step is to route traffic demand $\lambda$ to path $p$ (line 9), to finally add the synchronization traffic for the service chain (line 10).

### 5.2. Greedy algorithm

The greedy algorithm main function is described in Algorithm 2. The procedure starts with the natural ordering of SFCs by the total traffic demand value (line 1). This is done in order to first allocate services with lower impact on the utilization resources in order to avoid the creation of bottlenecks in servers and links during the firsts phases of the allocation. Then, it starts iterating over each service (line 2) and over each traffic demand for certain service (line 3). Then, for each traffic demand we first retrieve all paths with enough free link

---

**Algorithm 1:** First-Fit/Random-Fit: *main(FF_RF)*

1: **for** $s \in \mathbb{S}, \lambda \in \Lambda_s$ **do**
2:     $\mathbb{P}_s' \leftarrow$ getAdmissiblePaths$(s, \lambda)$
3:     $p \leftarrow$ choosePath(FF_RF, $\mathbb{P}_s'$)
4:     **for** $v \in \mathbb{V}_s$ **do**
5:         $\mathbb{X}_p' \leftarrow$ getAvailableServers$(s, \lambda, v, p)$
6:         $x \leftarrow$ chooseServer(FF_RF, $\mathbb{X}_p'$)
7:         addVNFToServer$(s, v, \lambda, x)$
8:     **end for**
9:     routeDemandToPath$(s, p, \lambda)$
10:     addSynchronizationTraffic$(s)$
11: **end for**

---

resources in $\mathbb{P}_s'$ (line 4). Then, we choose a path $p$ inside of a loop from all retrieved paths (line 6, details explained later). This is done to ensure that in case a path cannot be used for allocating all VNFs, the algorithm tries with the next one. Once the path is selected, we start with the placement of all VNFs on the selected path. First, all the available servers for a specific VNF $v$ on path $p$ are retrieved in variable $\mathbb{X}_p'$ (line 8), then we choose one server $x$ for that specific VNF in line 9 (this procedure explained later) and place the VNF (line 10). In case the VNF has been already placed by another demand of the same service, the demand is associated to that VNF, instead. Finally after all VNFs are placed, we map the demand over path (line 12). Finally, as in the previous case, the synchronization traffic for that service is added (line 15)

When selecting a path for a specific traffic demand in line 6, the procedure described in Algorithm 3 is executed. This procedure execute the following methods in this specific order: return an already used path for the same demand $\lambda$ during the initial placement (line 1), return any used path for SFC $s$ during the initial placement (line 4), return any used path for SFC $s$ (line 7) or return the path with shortest path delay (line 10). If one method does not return a path, then the next one is executed.

Going back to Algorithm 2, when choosing a server for a specific VNF in line 9, the procedure described in Algorithm 4 is executed. In this point, we first remove servers from the set $\mathbb{X}_p'$ that have already allocated VNFs before/after the current VNF in the path (lines 1 and 2), in order to satisfy with sequence order Eq. (11). Then, we follow up with the selection of a server from the remaining ones. Here, in case it exists, we first retrieve the cloud server $c$ in the path (line 3). Then, we retrieve a server already used for VNF $v$ and demand $\lambda$ during the initial placement (line 4) into server $x$. In line 5, we check the position in the path of that server, where the procedure is specified in line 13. This procedure receives the server $x$, the cloud server $c$ in case it exists and the boolean variable $A$ which specifies whether this is the last attempt in terms of remaining available paths. The objective here is to first check if $x$ is valid (line 14), otherwise it finishes. In case is valid, then we return $x$ if it is the last attempt $A$, if a cloud server exists in the path and if the index of $x$ is lower than the index of $c$ in the array. In case this condition does not apply, we continue with the next condition in line 17 with the difference of checking whether $x$ is after the cloud server in the array. In that case, the cloud server is returned. If none of the previous applies, then $x$ is returned in line 19. This procedure is basically performed to make sure that in all cases there will be a location where to place VNFs which is in the cloud server, but choosing it as the last option. Continuing with line 6, similarly here we try to retrieve a server used during the initial placement for service $s$ regardless for which traffic demand and perform the same procedure like in the previous case (line 7). While the first method tries to reuse the same exact server like in the initial placement in order to avoid a migration, here we try to use a server already used by some other demand during the initial placement for the same service in order to avoid a replication. Similarly, the next case

**Algorithm 2:** Greedy: *main()*

1: $\mathbb{S}' = \text{orderServicesByTotalDemandValue}(\mathbb{S})$
2: **for** $s \in \mathbb{S}'$ **do**
3:   **for** $\lambda \in \Lambda_s$ **do**
4:    $\mathbb{P}'_s \leftarrow \text{getAdmissiblePaths}(s, \lambda)$
5:    **for** $p \in \mathbb{P}'_s$ **do**
6:     $p \leftarrow \text{choosePath}(s, \lambda, \mathbb{P}'_s)$      ▷ go to Alg. 3
7:     **for** $v \in \mathbb{V}_s$ **do**
8:      $\mathbb{X}'_p \leftarrow \text{getAvailableServers}(s, \lambda, p, v)$
9:      $x \leftarrow \text{chooseServer}(s, \lambda, p, v, \mathbb{X}'_p)$   ▷ go to Alg. 4
10:      $\text{mapVNFToServer}(v, x)$
11:     **end for**
12:     $\text{mapDemandToPath}(s, p, \lambda)$
13:    **end for**
14:   **end for**
15:   $\text{addSynchronizationTraffic}(s)$
16: **end for**

---

**Algorithm 3:** Greedy: *choosePath(s, λ, $\mathbb{P}'_s$)*

1: $p \leftarrow \text{getUsedPathDemandInitPlacement}(s, \lambda, \mathbb{P}'_s)$
2: **if** $p$ **then return** $p$
3: **end if**
4: $p \leftarrow \text{getUsedPathInitialPlacement}(s, \mathbb{P}'_s)$
5: **if** $p$ **then return** $p$
6: **end if**
7: $p \leftarrow \text{getUsedPathForSFC}(s, \mathbb{P}'_s)$
8: **if** $p$ **then return** $p$
9: **end if**
10: **return** $\text{getPathWithShortestDelay}(s, \lambda, \mathbb{P}'_s)$

---

**Algorithm 4:** Greedy: *chooseServer(s, λ, p, v, $\mathbb{X}'_p$, A)*

1: $\mathbb{X}'_p \leftarrow \text{removeServersPreviousVNFs}(\mathbb{X}'_p)$
2: $\mathbb{X}'_p \leftarrow \text{removeServersNextVNFs}(\mathbb{X}'_p)$
3: $c \leftarrow \text{getCloudServer}(\mathbb{X}'_p)$
4: $x \leftarrow \text{getUsedServerDemandInitialPlace}(s, v, \lambda, \mathbb{X}'_p)$
5: $\text{checkPosition}(x, c, A)$       ▷ go to line 13
6: $x \leftarrow \text{getUsedServerInitialPlacement}(s, v, \mathbb{X}'_p)$
7: $\text{checkPosition}(x, c, A)$       ▷ go to line 13
8: $x \leftarrow \text{getUsedServerForSFC}(s, v, \mathbb{X}'_p)$
9: $\text{checkPosition}(x, c, A)$       ▷ go to line 13
10: **if** $!A$ **then return** null
11: **else return** $\mathbb{X}'_p[0]$
12: **end if**
13: **procedure** CHECKPOSITION$(x, c, A)$
14:   **if** $x \neq$ null **then**
15:    **if** $A$ & $c$ & $\text{indexOf}(x) < \text{indexOf}(c)$ **then**
16:     **return** x
17:    **else if** $A$ & $c$ & $\text{indexOf}(x) > \text{indexOf}(c)$ **then**
18:     **return** c
19:    **else return** $x$
20:   **end if**
21:   **end if**
22: **end procedure**



(a) N7 network     (b) N45 network

**Fig. 3.** Network topologies used in the performance evaluation.

in line 8 retrieves an already used server by the same service regardless it is from initial placement or allocated during the current placement. Here again we are trying to avoid an unnecessary replication and we check again like before the position of the returned server in line 9. If none of the previous methods returned a valid server, then we return null in line 10 in order to later try with the next available path in case this is not the latest path. If it is the latest path, then we just return the first available server in the set (line 11).

In terms of complexity from bottom to top, for the Algorithm 4 considering $V_L$ as the length of the longest SFC, it is in the order of $\Theta = O(V_L \cdot |\mathbb{X}|)$. The Algorithm 3 is in the order of $\Theta' = O(P_S)$ where $P_S$ is the number of paths per SFC. The Algorithm 2 is calculated based on the complexity of Algorithm 3 and 4, and the complexity of the synchronization traffic (line 15) which is in the order of $\Theta'' = O(V_L \cdot |\mathbb{X}|^2 \cdot |\mathbb{P}|)$. Considering $L_P$ as the length of the longest path, then the complexity of the entire Algorithm 2 is in the order of $O(|\mathbb{S}^2| + |\Lambda| \cdot L_P \cdot P_S \cdot [\Theta' + V_L \cdot \Theta] \cdot \Theta'')$, which can be simplified as $O(|\mathbb{S}^2| + |\Lambda| \cdot L_P \cdot V_L \cdot |\mathbb{X}|^2 \cdot |\mathbb{P}| \cdot [P_S^2 + V_L^2 \cdot |\mathbb{X}| \cdot P_S])$.

## 6. Performance evaluation

We use MILP model implemented with Gurobi Optimizer tool to evaluate a smaller size network N7 (7 nodes, 20 directed links with 500 units of capacity each, see Fig. 3(a)) and heuristics for a larger-size network N45 (45 nodes, 140, directed links with 1000 units of capacity, Fig. 3(b)). In N7, every node is equipped with one server, whereas in N45 there are 8 servers per node. In both networks, we assume that all nodes can establish on-demand connectivity to a third-party cloud server of which the geographic location is determined based on the closest common locations by cloud providers. Thus, for the 7-nodes network, in N7 the geographic locations are based regionally, such as the area of Braunschweig (Germany) for the network and

the area of Frankfurt, for the cloud server, respectively. For N45, we use a modified version of Palmetto network in South Carolina, USA and the cloud server in North Virginia, USA. The propagation delay is correspondingly calculated considering the distance between nodes from their latitude and longitude using the Haversine method using 2/3 of the speed of light. We thereby assume the links used to connect to the third-party cloud have sufficient capacity for any demand, and therefore do not impact the analysis of server utilization.

For each source–destination pair of nodes, 3 paths are pre-computed that do not traverse the cloud node and 1 additional path that does. Also 2 additional paths per node are computed for the synchronization traffic between possible VNFs allocated in the cloud and in the network. The path computation is carried out in this way to make sure the model has enough freedom to allocate all SFCs in the network and at least there is one admissible path per SFC to allocate VNFs in the cloud. We assume that every source–destination pair of nodes (except the cloud node) instantiates independent SFCs with variable length from 1 to 10 VNFs depending on the scenario. The processing load of a certain VNF is calculated from the total amount of processed traffic in the VNF multiplied by a random load ratio ($\Gamma^{\text{pro}}_{t(v)}$) between 1% and 100%. Additionally, an overhead ($\Theta^s_{t(v)}$) is calculated as a random percentage between 1% and 10% of the processing load [35]. The synchronization traffic between VNFs ($\Gamma^{\text{syn}}_{t(v)}$) is calculated as 10% of the processing load of the VNF. The delay parameters per VNF, already explained is

(a) RMSE                    (b) Training time



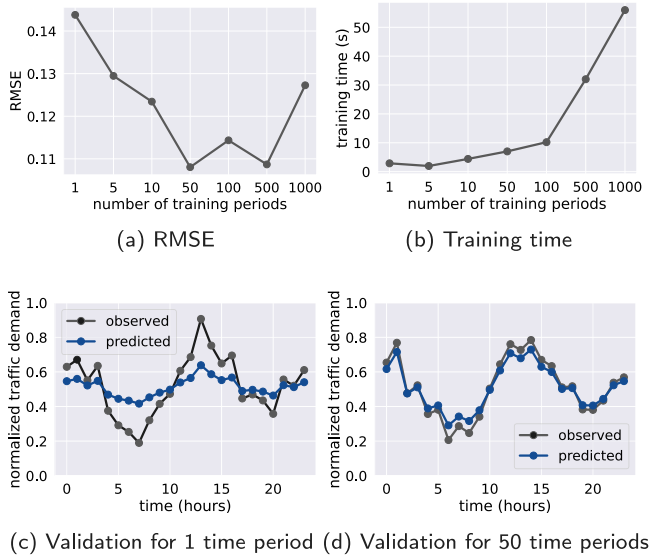(c) Validation for 1 time period (d) Validation for 50 time periods
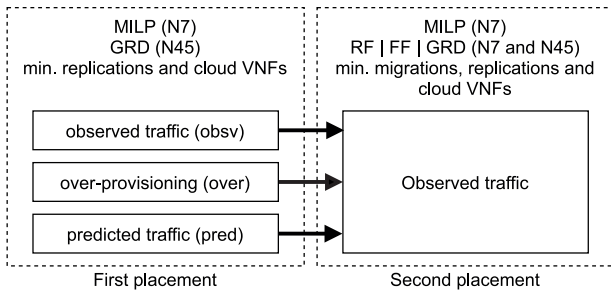
**Fig. 4.** Traffic prediction model results.



**Fig. 5.** Optimization scenarios.

Section 4.3.3, are specified using typical values as follows: $D_{t(v)}^{\text{proq}} = 3$ ms, $D_{t(v)}^{\text{prox}} = 5$ ms, $D_{t(v)}^{\text{pro\_x,min}} = 2$ ms and $D_{t(v)}^{\text{pro,max}} = 10$ ms. In the networks studied, for all SFCs the service delay is constrained to $D_s^{\text{max}} = 400$ ms. The round trip time is, for both networks, always shorter than 5 ms which leads to a service downtime of duration $D^{\text{dwt}} = 27.5$ ms when performing a migration, in the worst case scenario [9].

Two types of results are produced: (i) one setting all SFCs with a certain length while all servers have the same capacity and (ii) one setting all servers to the certain capacity while all SFCs have a random length. The reason for that is to independently see the effects that SFCs lengths and server capacities have on the network. In case (i), the server capacities are set to 1000 for N7 and 2000 units for N45, and all SFC lengths are chosen in increments from 1 to 10. In case (ii), the server capacities vary from 250 to 3000 units and every SFC is of random length between 1 and 10.

### 6.1. Optimization scenarios

We assume that every source–destination pair of nodes generates between 1 and 3 traffic flows, with traffic demand per flow set to a random value between 1 and 100 traffic units. For each traffic demand, 24 values are generated in one time period following a lognormal distribution with time-varying mean and variance, as explained in Section 3.3. For the time series forecasting, one LSTM network is created and trained per each traffic flow for a certain number of time periods, and then evaluated for 1 time period.

To determine the optimum number of required training time periods, the model has been tested using from 1 to 1000 time periods

for training. The resulting RMSE is shown in 4(a) where it shows that above 50 periods of time, the performance is not improving anymore and it starts decreasing for 1000 training time periods. The reason why to train with larger dataset is typically to solve overfitting, but in our specific the RMSE does not improve due to high variance on our input data even when using more data or applying the dropout technique. The training time, however, continues to increase with the number of training time periods as expected, see Fig. 4(b). Taking 1 time period as the worst case and 50 as the best case, Figs. 4 and 4(d) show the predicted and observed normalized traffic demand values over time during the evaluation period, respectively. Here, we can see how the number of training time periods impacts the accuracy of the model.

To illustrate the issues of computation time, we show the results obtained by using the CPU of a machine with an Intel Core i7-6700 and 32 GB of RAM. The total computation time considering all traffic demands for takes ≈7 min in N7, when training for 1 time period and ≈12 min when training for 50 time periods. For N45, it takes in total ≈13 h for 50 training time periods. While the specific total computational time can be improved by using GPUs or by training models in parallel, it should be noted that the network size needs to be considered when using predictions.

From the generated traffic demand values produced for the evaluation period, three optimization scenarios are derived based on which values are considered during the first placement: (i) observed values (obsv), (ii) 80% of the maximum individual traffic demand values, which corresponds to overprovisioning (over) and (iii) predicted values (pred). After the first placement, the second placement is carried out considering the location of the VNFs during the first placement, as explained in Eq. (3) and considering the new traffic demand values after a Δt time shift from the set of traffic demand values (see Fig. 4(d)). In our case, the first time step for the first placement is taken randomly from the first 18 time values and Δt is set to 6 time periods. Hence, for the first scenario obsv, only the current observed values at time t are considered for the placement of VNFs. In the second scenario over, the observed values are ignored, and instead, the VNFs are placed assuming the traffic is always at the 80% of the maximum traffic demand value. The third scenario places VNFs considering the predicted traffic values after Δt.

Fig. 5 illustrates the optimization process. The second placement uses the first placement as input, and it optimizes the placement again by considering the real monitored and observed traffic demand values. The first placement is carried out using either the MILP model in N7, or the greedy algorithm (GRD) in N45. In all cases, the objective is to allocate VNF while minimizing the number of replications and the number of virtual functions placed in the cloud. In the first placement, the are no migrations from any previous step to consider. In the second placement, the MILP model in N7, and all heuristics for both networks, all by considering the same objectives which is to minimize the number of migrations, replications and cloud VNFs. Regarding computational time, for every optimization using the MILP model, the times vary in the order of seconds/minutes for the N7 network. When using heuristics, the computational time is always in the order of milliseconds, for the N7 network, or seconds for the N45 network. Finally, for the reminder of the paper we show the results obtained from the second placement, while using the three scenarios during the first placement, as described.

### 6.2. Objective function

Since the objective function (Eq. (2)) is a joint optimization from three different weighted terms, we first show the results when minimizing all terms, so all three weights $W_m$, $W_r$ and $W_c$ are equal to 1. Fig. 6 shows the objective value for the three scenarios obsv, over and pred when varying the SFC lengths and when varying the server capacities in N7. It should be noted that some zero values for certain SFC lengths or server capacities are omitted in the plots
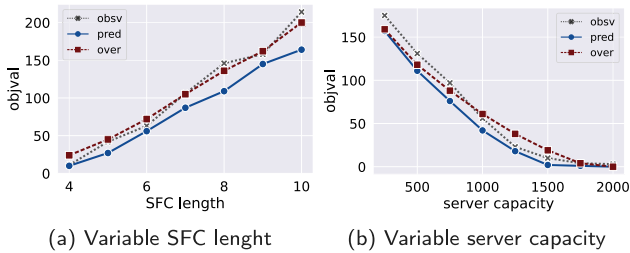
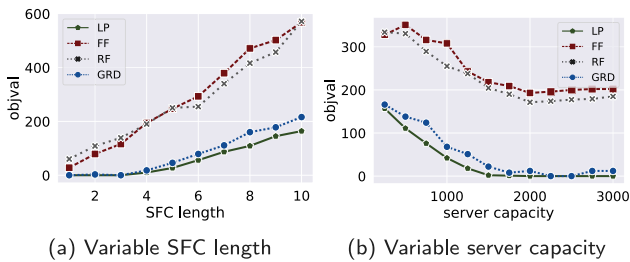**Fig. 6.** Objective function value for `obsv`, `over` and `pred` scenarios in the N7 network using the `MILP` model.



**Fig. 7.** Objective function values of RF, FF, GRD and MILP for the `pred` scenario in the N7 network.



**Fig. 8.** Objective function value for `obsv`, `over` and `pred` scenarios in the N45 network using the `GRD` algorithm.



**Fig. 9.** Objective function values for RF, FF and GRD for the `pred` scenario in the N45 network.

due to clarity. We can observe that `pred` overperforms the other two cases. Between `over` and `obsv`, when the servers are overloaded the `over` case performs slightly better than `obsv` as expected, due to the overprovisioning factor.

Before analyzing the three scenarios in large network N45, let us first compare how heuristics compare to MILP model in N7. Fig. 7 shows again the objective values for `pred` scenario, but now comparing the MILP model with the heuristic algorithms RF, FF and GRD. Here we can see that both RF and FF are far from the optimal solution, being RF slightly better than FF in most cases.

When using the greedy algorithm for the N45 network, we compare again the three scenarios `obsv`, `over` and `pred` in Fig. 8. Here, we can see a more clear difference between the three cases, being again the `pred` scenario the one with a clear advantage compared to the other two. This case also better illustrates how `over` case overperforms `obsv` case mostly when the servers are overloaded confirming what we could slightly see with the N7 network. From Fig. 9 we can compare RF, FF and GRD, in this case for the N45 network. Different from N7, here we can see how FF outperforms RF in all cases. We see here a trend on FF working better the more free the network and servers are, but in any case the achieved values are comparable to the GRD algorithm which performs always better.

### 6.3. Migrations, replications and cloud VNFs

In order to better see how the model behaves individually when minimizing only one of the terms, we set a certain weight (i.e. $W_m$, $W_r$ or $W_c$) equal to 1, and the others close to 0 in such a way that the sum of all secondary terms is within interval $[0, 1)$. By doing that, we limit the freedom of the model while, at the same time, we ensure there is no impact on the main term which value is always going to be a positive integer. In this regard, Fig. 10(a) shows the results in terms of number of replications (`rep`) and number of cloud VNFs (`cld`), when minimizing the number of migrations for the three scenarios `obsv`, `over` and `pred` and different SFC lengths in N7. By looking at `over-rep` and `over-cld`, we see how overprovisioning does not allocate replicas and places more functions in the cloud compared to other cases. In comparison, the `obsv` case allocates less functions in the cloud at expenses of deploying a considerable number of replicas. The `pred`
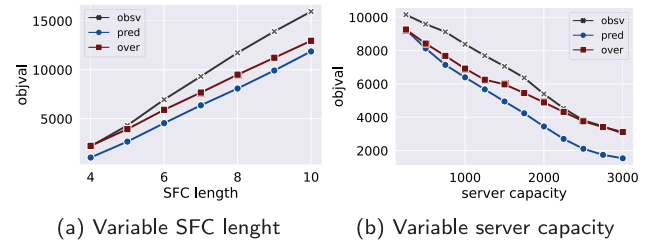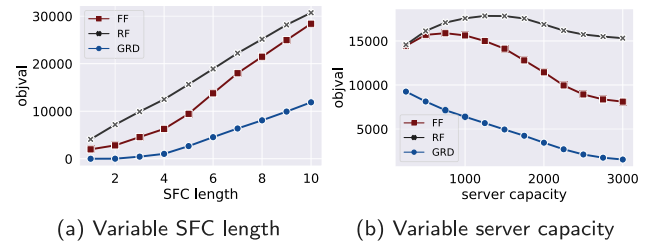
case can be seen as a trade-off solution, as it allocates considerably less VNFs in the cloud compared to `over`, independently from the SFC length, and less than `obsv` mostly when the servers are overloaded with long SFCs. In terms of replicas, the `pred` requires much less resources in almost all cases compared to `obsv` case. When minimizing the number of replications, see Fig. 10(b), the difference between `pred` and `obsv` in terms of allocations in the cloud is much smaller, but still reduces the number of migrations independently from the SFC length. Here the `over` case behaves quite similar to `pred` in number of migrations, but instead requires to allocate more cloud VNFs. When minimizing the number of functions in the cloud, see Fig. 10(c), we see how `pred` requires much less migrations compared to the other two cases, but no remarkable difference regarding replications.

To individually see the number of migrations, replications and cloud VNFs with no influence from the weights (i.e. all terms the same weight), we now study N45 network. Fig. 11(a) shows how `obsv` case requires much more migrations compared to the other cases except when the servers are either too overloaded or too underloaded where the values become closer to `over` case. On the other hand, `pred` case requires the same number of migrations than `over` when the servers are overloaded and improves when there is enough free available resources. In Fig. 11(b), regarding the number of replications we see that there is no much difference between `pred` and `obsv`, but `over` case is the one requiring significantly less replications, except for the cases where the servers are either too overloaded or too underloaded. This effect can be explained by the fact that when there are no available resources in the servers, the model cannot perform replications, and when there are more than enough available resources, the model avoids replications that are not essential. When looking at Fig. 11(c), we see that there is almost no difference between `obsv` and `pred`, but the `over` case allocates considerably more cloud VNFs than the other two cases.

### 6.4. Resource utilization and service delay

To show the difference between the three scenarios, Fig. 12(a), Figs. 12(b) and 12(c) show the average link utilization, server utilization and service delay, respectively, versus a varying server capacity for N7. For both link and server utilization, the link capacity connecting to the cloud and the cloud servers are not considered. Here, in most
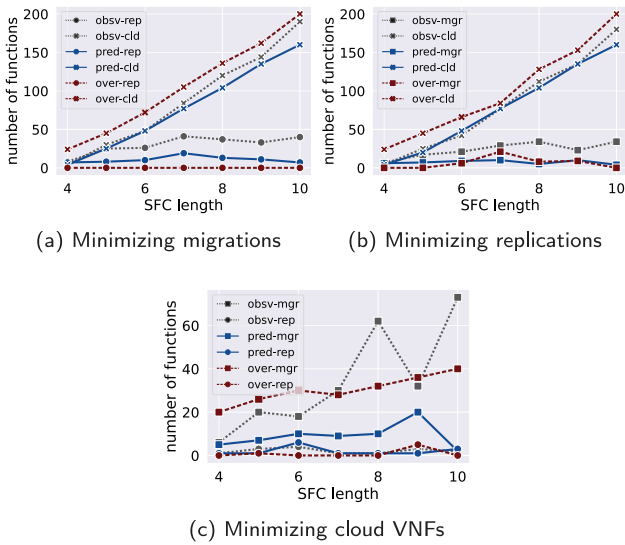
(a) Minimizing migrations



(b) Minimizing replications



(c) Minimizing cloud VNFs

**Fig. 10.** Number of migrations, replications and cloud VNFs for different SFC lengths in the N7 network using `MILP` model.



(a) Number of migrations
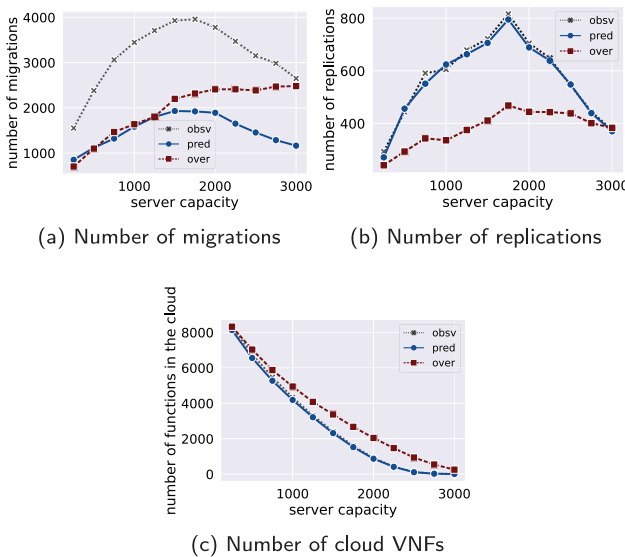


(b) Number of replications



(c) Number of cloud VNFs

**Fig. 11.** Number of migrations, replications and cloud VNFs for different server capacity in N45 network using `GRD` algorithm.



(a) Average link utilization



(b) Average server utilization



(c) Average service delay

**Fig. 12.** Resource utilization and service delays for different server capacities in the N7 network.



(a) Average link utilization



(b) Average server utilization



(c) Average service delay
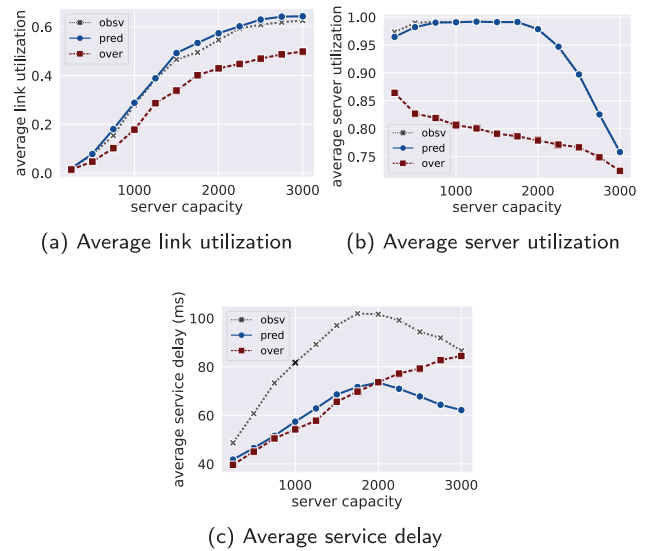
**Fig. 13.** Resource utilization and service delays for different server capacities in the N45 network.

cases when the network is not overloaded, the `over` case has slightly lower link utilization compared to the other cases, since this case allocates more cloud VNFs, so the edge network is less utilized and less replicas are used, so less synchronization traffic is added to the network. Between `pred` and `obsv` cases, the first one has slightly lower link utilization in some specific cases. This difference is inexistent when looking at the server utilization, and here only `over` case has lower utilization for the same reason as before. When comparing the three cases for the average service delay, we notice how `over` has the lowest delay, even though it allocates generally more cloud VNFs as we have seen before, so the propagation delay is larger. However, this case performs less migrations compared to the other cases, and therefore, there is less penalty due to service interruptions. When comparing `pred` with `obsv`, we see how `pred` has less service delay, so less migrations are required.

Fig. 13 shows again the same results, but this time for the N45 network. Here, we can better see the difference in the lower utilization of links of the `over` case compared with the other two. This is again
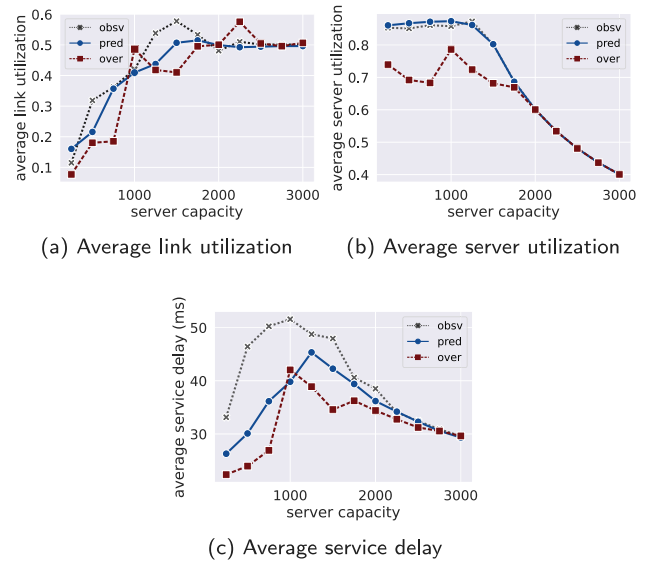
due to the fact that overprovisioning results into a higher usage of the cloud, so the network is less utilized. This is also confirmed when looking at the average server utilization where `pred` and `obsv` cases make full usage of all server resources at the edge before using the cloud, contrary to the `over` case. The most interesting case is with regard the service delay, where we can see how the `pred` case is able to outperform `over` when the servers are not overloaded since the number of migrations are much lower as we could see from Fig. 11(a).

### 6.5. Discussion and remarks

From all three scenarios analyzed, we observe that in all cases, predicting the traffic demands helps to reduce the overall number of migrations, replications and usage of the cloud. More specifically, the overprovisioning case requires in general less replications compared to the other two cases, but requires as many migrations as with the prediction case, when the network is overloaded and considerably more when the network is underloaded. Because overprovisioning does not consider the fluctuations of traffic, it can, in the best case, match the real traffic and it can, in the worst case, provision excessive resources in

advance, which results in using more the cloud compared to the other two cases. Placing VNFs only considering the observed traffic results in using a similar total amount of resources as with prediction, since there is no much difference in the number of replications and usage of the cloud, but it requires significantly more migrations to be able to accommodate future demands. In summary, we can say that when using traffic prediction, the number of migrations can be reduced up to 45% when there is enough available resources to allocate replicas, compared to other cases studied. This is true at expenses of using replications and cloud placements, as much as in the observed traffic case. When comparing it to the overprovisioning case, that statement remains true, but also the usage of the cloud is reduced by allocating almost up to double number of replications. However, for traffic prediction to successfully help on this problem, it requires certain amount of training time periods per independent traffic demand in the network, which can result in high computational resources and computational time for larger networks.

## 7. Conclusions

We studied the problem of optimal placement of VNFs from an ISP point of view, when minimizing migrations and replications. We proposed a traffic forecasting model using LSTM networks and used it to place VNFs accordingly to the predicted traffic demands. We proposed an offline MILP model as well as an online greedy algorithm for the placement optimization problem. We compared three scenarios by either considering: (i) the current observed traffic demands only, (ii) overprovisioning of the 80% of every specific maximum traffic demand value had in the past, or (iii) the predicted traffic values based on history. We showed that with traffic prediction, the number of migrations can be reduced up to 45% when there is enough available resources to allocate replicas. This also results in less usage of the third-party clouds as compared to capacity overprovisioning. While overprovisioning can be valid a solution when unexpected traffic peaks appear resulting in higher usage of the cloud temporarily, traffic prediction can minimize the need for the same by anticipating a proper placement and replication inside the network. The usage of LSTM networks, however, requires non-negligible training time and computational resources which is also something that needs to be taken into consideration.

### Declaration of competing interest

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

### Acknowledgments

## References

[1] A. Laghrissi, T. Taleb, A survey on the placement of virtual resources and virtual network functions, IEEE Commun. Surv. Tutor. 21 (2) (2019) 1409–1434, http://dx.doi.org/10.1109/COMST.2018.2884835.

[2] M.M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, B. Akbari, Joint energy efficient and qos-aware path allocation and VNF placement for service function chaining, IEEE Trans. Netw. Serv. Manag. PP (c) (2017) 1, http://dx.doi.org/10.1109/TNSM.2018.2873225.

[3] A. Basta, A. Blenk, K. Hoffmann, H.J. Morper, M. Hoffmann, W. Kellerer, Towards a cost optimal design for a 5G mobile core network based on SDN and NFV, IEEE Trans. Netw. Serv. Manag. 4537 (c) (2017) 1–14, http://dx.doi.org/10.1109/TNSM.2017.2732505.

[4] L. Qu, C. Assi, K. Shaban, M.J. Khabbaz, A reliability-aware network service chain provisioning with delay guarantees in NFV-enabled enterprise datacenter networks, IEEE Trans. Netw. Serv. Manag. 14 (3) (2017) 554–568, http://dx.doi.org/10.1109/TNSM.2017.2723090.

[5] M. Golkarifard, C.F. Chiasserini, F. Malandrino, A. Movaghar, Dynamic VNF placement, resource allocation and traffic routing in 5G, Comput. Netw. 188 (2021) 107830, http://dx.doi.org/10.1016/j.comnet.2021.107830.

[6] J. Xia, D. Pang, Z. Cai, M. Xu, G. Hu, Reasonably migrating virtual machine in NFV-featured networks, in: IEEE International Conference on Computer and Information Technology, CIT, 2016, http://dx.doi.org/10.1109/CIT.2016.96.

[7] J. Xia, Z. Cai, M. Xu, Optimized virtual network functions migration for NFV, in: 2016 IEEE 22nd International Conference on Parallel and Distributed Systems, ICPADS, IEEE, 2016, http://dx.doi.org/10.1109/ICPADS.2016.0053.

[8] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, A. Akella, Opennf: Enabling innovation in network function control, SIGCOMM Comput. Commun. Rev. 44 (4) (2014) 163–174, http://dx.doi.org/10.1145/2740070.2626313.

[9] T. Taleb, A. Ksentini, P.A. Frangoudis, Follow-me cloud: When cloud services follow mobile users, IEEE Trans. Cloud Comput. 7 (2) (2019) 369–382, http://dx.doi.org/10.1109/TCC.2016.2525987.

[10] R. Cziva, C. Anagnostopoulos, D.P. Pezaros, Dynamic, latency-optimal vNF placement at the network edge, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, 2018-April, IEEE, 2018, pp. 693–701, http://dx.doi.org/10.1109/INFOCOM.2018.8486021.

[11] V. Eramo, E. Miucci, M. Ammar, F.G. Lavacca, An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures, IEEE/ACM Trans. Netw. (2017) http://dx.doi.org/10.1109/TNET.2017.2668470.

[12] K.S. Michael Till Beck, Juan Felipe Botero, M.T. Beck, J.F. Botero, K.S. Michael Till Beck, Juan Felipe Botero, Resilient allocation of service function chains, in: IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN, IEEE, 2016, http://dx.doi.org/10.1109/NFV-SDN.2016.7919487.

[13] A. Engelmann, A. Jukan, A reliability study of parallelized VNF chaining, in: 2018 IEEE International Conference on Communications, ICC, 2018-May, IEEE, 2018, pp. 2–7, http://dx.doi.org/10.1109/ICC.2018.8422595.

[14] W. Ding, H. Yu, S. Luo, Enhancing the reliability of services in NFV with the cost-efficient redundancy scheme, in: 2017 IEEE International Conference on Communications, vol. 1, ICC, IEEE, 2017, http://dx.doi.org/10.1109/ICC.2017.7996840.

[15] Q. Yuan, X. Ji, H. Tang, W. You, Toward latency-optimal placement and autoscaling of monitoring functions in MEC, IEEE Access 8 (2020) 41649–41658, http://dx.doi.org/10.1109/ACCESS.2020.2976858.

[16] F. Carpio, S. Dhahri, A. Jukan, VNF Placement with replication for load balancing in NFV networks, in: IEEE International Conference on Communications, IEEE, 2017, http://dx.doi.org/10.1109/ICC.2017.7996515.

[17] F. Carpio, W. Bziuk, A. Jukan, Replication of virtual network functions: Optimizing link utilization and resource costs, in: 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO, Croatian Society MIPRO, 2017, http://dx.doi.org/10.23919/MIPRO.2017.7973481.

[18] M. Huang, W. Liang, Y. Ma, S. Guo, Throughput maximization of delay-sensitive request admissions via virtualized network function placements and migrations, ICC, IEEE International Conference on Communications 2018-May (c) (2018) http://dx.doi.org/10.1109/ICC.2018.8422337.

[19] F. Carpio, A. Jukan, R. Pries, Balancing the migration of virtual network functions with replications in data centers, in: NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, IEEE, 2018, http://dx.doi.org/10.1109/NOMS.2018.8406275.

[20] H.A. Alharbi, T.E. Elgorashi, A.Q. Lawey, J.M. Elmirghani, The impact of inter-virtual machine traffic on energy efficient virtual machines placement, in: 2019 IEEE Sustainability Through ICT Summit, StICT, IEEE, 2019, http://dx.doi.org/10.1109/STICT.2019.8789381.

[21] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, R. Boutaba, A connectionist approach to dynamic resource management for virtualised network functions, in: 2016 12th International Conference on Network and Service Management, CNSM, IEEE, 2016, http://dx.doi.org/10.1109/CNSM.2016.7818394.

[22] S. Rahman, T. Ahmed, M. Huynh, M. Tornatore, B. Mukherjee, Auto-scaling VNFs using machine learning to improve QoS and reduce cost, in: 2018 IEEE International Conference on Communications, ICC, IEEE, 2018, http://dx.doi.org/10.1109/ICC.2018.8422788.

[23] Y. Yao, S. Guo, P. Li, G. Liu, Y. Zeng, Forecasting assisted VNF scaling in NFV-enabled networks, Comput. Netw. 168 (2020) 107040, http://dx.doi.org/10.1016/j.comnet.2019.107040.

[24] Q. Sun, P. Lu, W. Lu, Z. Zhu, Forecast-assisted NFV service chain deployment based on affiliation-aware vNF placement, in: 2016 IEEE Global Communications Conference, GLOBECOM, IEEE, 2016, http://dx.doi.org/10.1109/GLOCOM.2016.7841846.

[25] H. Tang, D. Zhou, D. Chen, Dynamic network function instance scaling based on traffic forecasting and VNF placement in operator data centers, IEEE Trans. Parallel Distrib. Syst. 30 (3) (2019) 530–543, http://dx.doi.org/10.1109/TPDS.2018.2867587.

[26] K. Qu, W. Zhuang, X. Shen, X. Li, J. Rao, Dynamic resource scaling for VNF over nonstationary traffic: A learning approach, IEEE Trans. Cogn. Commun. Netw. (2020) 1, http://dx.doi.org/10.1109/TCCN.2020.3018157.

[27] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, R. Boutaba, Topology-aware prediction of virtual network function resource requirements, IEEE Trans. Netw. Serv. Manag. 14 (1) (2017) 106–120, http://dx.doi.org/10.1109/TNSM.2017.2666781, URL: https://ieeexplore.ieee.org/document/7849149.

[28] R. Shi, J. Zhang, W. Chu, Q. Bao, X. Jin, C. Gong, Q. Zhu, C. Yu, S. Rosenberg, MDP And machine learning-based cost-optimization of dynamic resource allocation for network function virtualization, in: Proceedings - 2015 IEEE International Conference on Services Computing, SCC 2015, IEEE, 2015, pp. 65–73, http://dx.doi.org/10.1109/SCC.2015.19, URL: https://ieeexplore.ieee.org/document/7207337.

[29] H.-G. Kim, D.-Y. Lee, S.-Y. Jeong, H. Choi, J.-H. Yoo, J.W.-K. Hong, Machine learning-based method for prediction of virtual network function resource demands, in: 2019 Conference on Network Softwarization, NetSoft, IEEE, 2019, http://dx.doi.org/10.1109/NETSOFT.2019.8806687, URL: https://ieeexplore.ieee.org/document/8806687.

[30] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, P. Bertin, Improving traffic forecasting for 5G core network scalability: A machine learning approach, IEEE Netw. 32 (6) (2018) 42–49, http://dx.doi.org/10.1109/MNET.2018.1800104.

[31] T. Subramanya, R. Riggio, Machine learning-driven scaling and placement of virtual network functions at the network edges, in: 2019 IEEE Conference on Network Softwarization, NetSoft, IEEE, 2019, http://dx.doi.org/10.1109/NETSOFT.2019.8806631.

[32] J. Pei, P. Hong, M. Pan, J. Liu, J. Zhou, Optimal VNF placement via deep reinforcement learning in SDN/NFV-Enabled networks, IEEE J. Sel. Areas Commun. 38 (2) (2020) 263–278, http://dx.doi.org/10.1109/JSAC.2019.2959181.

[33] S. Uhlig, B. Quoitin, J. Lepropre, S. Balon, Providing public intradomain traffic matrices to the research community, SIGCOMM Comput. Commun. Rev. 36 (1) (2006) 83–86, http://dx.doi.org/10.1145/1111322.1111341.

[34] A. Bulut, T. Ralphs, On the complexity of inverse mixed integer linear optimization, 2015, URL: https://coral.ise.lehigh.edu/~ted/files/papers/InverseMILP15.pdf, 1–18.

[35] P.V.V. Reddy, L. Rajamani, Virtualization overhead findings of four hypervisors in the CloudStack with SIGAR, in: 2014 4th World Congress on Information and Communication Technologies,, WICT 2014, IEEE, 2014, pp. 140–145, http://dx.doi.org/10.1109/WICT.2014.7077318.

**Francisco Carpio** received the M.Sc. degree in Telecommunications Engineering from universitat Politècnica de Catalunya (UPC), Spain, in 2014. He is currently pursuing the Ph.D. degree with the Institute of Computer and Network Engineering at the Technische Universität Braunschweig, Germany. His research interests include network function virtualization, edge computing networks and machine learning applied to service management.



**Wolfgang Bziuk** received the Dipl.Ing. and Dr.Ing. degrees in electrical engineering from Technische Universität Braunschweig, Germany. He has been a Senior Lecturer with the Institute of Computer and Network Engineering, Technische Universität Braunschweig, since 1986. He was involved in the area of network planning tools, traffic engineering, performance evaluation, and simulation of networks, with applications to connection admission control. He was also involved in the areas of mathematical methods for mobility modeling for wireless networks as well as performance evaluation methods for optical networks. His current research interests include the mathematical modeling of security and reliability in NFV-based network architectures. He is a member of the Section 5.2.1 System Architecture and Traffic Engineering of the German Information Technology Society.
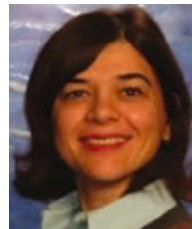


**Admela Jukan** received the Dr.Tech. degree (summa cum laude) in electrical and computer engineering from the Technische Universität Wien, the M.Sc. degree in information technologies from the Politecnico di Milano, Italy, and the Dipl.Ing. degree from the Fakultet Elektrotehnike i Racunarstva (FER), Zagreb, Croatia. She is Chair Professor of communication networks with the Technische Universität Carolo-Wilhelmina zu Braunschweig, Braunschweig, Germany. Dr. Jukan was a recipient of an Award of Excellence for the BMBF/CELTIC project 100Gb Ethernet and the IBM Innovation Award, in 2009. She has chaired and co-chaired several international conferences, including the IEEE/ACM IWqoS, IEEE ANTS, IFIP ONDM, IEEE ICC, and IEEE GLOBECOM. She was an elected Chair of the IEEE Optical Network Technical Committee (ONTC), from 2014 to 2015. She serves as a Senior Editor for the IEEE Journal of Selected Areas in Communications. She is a Co-Editor-in-Chief of the Elsevier Journal on Optical Switching and Networking (OSN).