Foresighted Resource Provisioning for Network Slicing

Quang-Trung Luu, Sylvaine Kerboeuf, Michel Kieffer

Foresighted Resource Provisioning for Network Slicing

Quang-Trung Luu*†, Sylvaine Kerboeuf*, and Michel Kieffer†
*Nokia Bell Labs, 7 Route de Villejust, 91620 Nozay, France
†University of Paris-Saclay - CNRS - CentraleSupélec, Laboratoire des Signaux et Systèmes (L2S),
91192 Gif-sur-Yvette, France, e-mails: {quangtrung.luu, michel.kieffer}@centralesupelec.fr

***Abstract*—Network slicing has emerged as a pivotal concept in 5G systems, allowing mobile operators to build isolated logical networks (slices) on top of shared infrastructure networks. Within a network slice, several Service Function Chains are usually deployed on a best-effort premise. Nevertheless, this approach does not guarantee the availability of enough infrastructure resources to accommodate the uncertain and time-varying slice resource demands.**

This paper investigates two adaptive slice resource provisioning methods accounting for the evolution with time of the slice resource demands. A probabilistic guarantee of meeting the slice resource requirements can be obtained, while being robust against uncertainties. The *myopic* approach accounts for the past demands when provisioning the current demands, while the foresighted approach accounts for both past and future demands. These two methods lead to MILP problems. Their performance is compared with a quasi-static method, where provisioning is agnostic of the past and future demands.

***Index Terms*—Network slicing, foresighted provisioning.**

I. Introduction

With Network Slicing (NS), multiple slices, *i.e.*, *customized*, *isolated*, and *service-dedicated* end-to-end logical networks can be operated simultaneously in a shared infrastructure network [1]. Allocating resources for network slices so as to fulfill the diverse service requirements is very important in this context [2, 3]. Slice resource demands vary with time, due to the dynamic nature of slice resource requests and to the uncertainties related to the number of users, hardly predictable user location [4], and time-varying per-user resource requirements. An efficient slice resource provisioning method has to be robust to these dynamic characteristics.

Adopting the perspective of the Infrastructure Provider (InP), this paper investigates an efficient way to adaptively provision resources for network slices. The aim is to provide a probabilistic guarantee of meeting the slice resource requirements while maximizing the benefits of the InP. This work adapts it to a dynamic context the slice resource provisioning approach introduced in [5]. It provides a provisioning mechanism robust against the dynamic behavior of concurrent slice provisioning requests, each of which has random arrival, activation, and deactivate time.

The topic of dynamic slice deployment has received significant attention, see, *e.g.*, [6–8]. Taking the dynamic nature of slice requests as a starting point, these works aim to design adaptive resource allocation and reconfiguration algorithms for the deployment of network slices or Service Function Chains (SFCs). In [6], a dynamic resource allocation for SFCs is investigated, in which the deployment of newly arrived SFCs and readjustment of in-service SFCs are taken into account. Readjustment of in-service SFCs usually involves the migration of instances of Virtual Network Functions (VNFs) and the update of virtual links to match the resource demand variations. An ILP formulation is used to address the dynamic deployment problem, aiming at minimizing the cost of VNF deployment and migration. In [8], an hybrid slice reconfiguration framework is introduced. The slice reconfiguration can be performed either within small time intervals for individual slices, or within large time intervals to readjust resource allocation of multiple slices. A deep-learning approach is considered in [9] aiming to maximize the long-term revenue of the network provider. The uncertainties related to the slice allocation requests and occupation time are considered. Nevertheless, a slice is regarded as a whole, *i.e.*, not made up of multiple elements (*e.g.*, VNFs), which over-simplifies the problem of slice resource allocation.

This paper is organized as follows. In Section II, we introduce notations and hypotheses. In Section II, the infrastructure and the slice resource demands are detailed. Section III describes the proposed approaches to provision resources for concurrent slices, while being robust to the dynamic nature of slice requests and to the uncertainties related to infrastructure and slice parameters. Numerical results are then provided in Section IV before drawing some conclusions in Section V.

II. Notations and Hypotheses

A typical network slicing system involves several entities: one or many InPs, Mobile Network Operators (MNOs), and SPs, see Figure 1 [10]. An InP manages the wireless and wired infrastructure such as the cell sites, the fronthaul and backhaul networks, and cloud datacenters. An MNO leases resources from InPs to setup and manage the slices. An SP then exploits the slices supplied by an MNO, and provides to its customers the required services running within the slices. Service needs are forwarded by an SP to an MNO within an SLA denoted SM-SLA in what follows.

One considers SM-SLAs composed of: *i*) a probability mass function (pmf) describing the target number of users/devices to be supported by the slice, *ii*) a description of the characteristics

978-1-6654-4005-9/21/$31.00

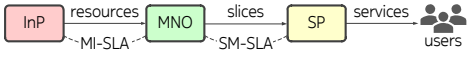


Fig. 1. Network slicing entities and their SLA-based relationships.

of the service and of the way it is employed by a typical user/device, and *iii*) a target service satisfaction probability (SSP). Several time intervals over each of which the service characteristics and constraints are assumed stable are considered in the SM-SLA.

Each slice consists of one or multiple SFCs of different types. An SFC consists of an ordered set of interconnected VNFs describing the processing of data flows related to a given service. To characterize the variability over time and among users, we assume that the MNO considers a probabilistic description of the consumption of slice resources by a typical user. The MNO then forwards to the InP these characteristics as part of an SLA between them (MI-SLA).

Each InP then provisions infrastructure resources needed for the SFCs. In this paper, one considers an infrastructure owned by a single InP. To perform the provisioning, the InP has to identify the infrastructure nodes which will provide resources for future deployment of VNFs and the links able to transmit data between these nodes, while respecting the structure of SFCs and optimizing a given objective (*e.g.*, minimizing the infrastructure and software license costs).

Taking the InP perspective, our aim is to reserve enough infrastructure resources to ensure that the MNO will be able to provide a slice with characteristics as stated in the SM-SLA. The InP serves periodically new slice provisioning requests or updates the provisioning schemes of in-service slices. One considers that time is slotted into slots of constant duration T (typically of few tens of minutes). The slot of index $k \in \mathbb{N}$ lasts over the time interval $[kT, (k+1)\,T[$. One considers that the slice lifetime spans over several time slots of duration T over each of which resources will be provisioned so as to be compliant with the variations of the number of users and of their demands during the slice lifetime. For each time slot of index k, a slice resource provisioning decision is made in advance during the interval $[kT - \varepsilon, kT[$, where $\varepsilon < T$. This time interval is required for the infrastructure *provisioning* and *allocation* and for the slice deployment and activation.

A. Network Model

Consider an infrastructure network managed by a given InP. This network is represented by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of infrastructure nodes and $\mathcal{E}$ is the set of infrastructure links. The resources at each $i \in \mathcal{N}$ are denoted as $a_n(i)$, $n \in \Upsilon = \{\mathrm{c}, \mathrm{m}, \mathrm{w}\}$, representing the total amount of computing $a_{\mathrm{c}}(i)$, memory $a_{\mathrm{m}}(i)$, and wireless $a_{\mathrm{w}}(i)$ resources.

The resource cost associated to a given node i consists of a fixed part $c_{\mathrm{f}}(i)$ for node disposal (paid to the InP for each slice using node i), and per-unit variable parts $c_{\mathrm{c}}(i)$, $c_{\mathrm{m}}(i)$, and $c_{\mathrm{w}}(i)$. Similarly, each infrastructure link $ij \in \mathcal{E}$ connecting node i to j has an available bandwidth $a_{\mathrm{b}}(ij)$, and an associated per-unit bandwidth cost $c_{\mathrm{b}}(ij)$. Distinct VNFs of the same slice may be deployed on a given infrastructure node. When communication between these VNFs is required, an internal (loop-back) infrastructure link $ii \in \mathcal{E}$ can be used at each node $i \in \mathcal{N}$, as in [11].

A fixed cost $c_{\mathrm{d}}(i, v)$ accounts for downloading VNF image v from a local registry at node i.Finally, a fixed cost $c_{\mathrm{a}}(v)$ accounts for the provisioning readjustment of the number of VNF instances of type v between consecutive time slots.

B. Slice Provisioning Requests

1) Request Arrivals: Let k_s be the time slot during which the provisioning request for slice s arrives at the MNO. This slice is characterized by the index k_s^{on} of the time slot at the beginning of which it has to be activated, and the index k_s^{off} of the time slot at the end of which it has to be deactivated.

Considering the time required for the provisioning and deployment of the resources of a slice, see Section II, the provisioning operations for time slot k start at $kT - \varepsilon$ for a slice activation at kT. Consequently, requests for slices to be activated at kT should reach the MNO before $kT - \varepsilon$.

Let $\mathcal{S}_k$ be the set of slice indexes whose provisioning request reaches the MNO before $kT - \varepsilon$. Moreover, let $\mathcal{S}_{k|k}^{+} \subset \mathcal{S}_k$ be the set of slices that need to be active during time slot k, one has $\mathcal{S}_{k|k}^{+} \triangleq \left\{ s \in \mathcal{S}_k : k \in \left[k_s^{\mathrm{on}}, k_s^{\mathrm{off}}\right] \right\}$.

2) Slice Resource Demand: A demand for slice resources is defined on the basis of an SM-SLA between an SP and the MNO. As in [12], we consider that a slice is devoted to a single type of service supplied by a given type of SFC. Several SFCs of the same type may have to be deployed so as to satisfy user demands. The topology of each SFC of slice s is represented by a graph $\mathcal{G}_s = (\mathcal{N}_s, \mathcal{E}_s)$ representing the VNFs and their interconnections. Each virtual node $v \in \mathcal{N}_s$ represents a VNF and each virtual link $vw \in \mathcal{E}_s$ represents the connection between virtual nodes v and w.

Based on $\mathcal{G}_s$, one introduces the following resource demand (RD) vectors.

The vector $\boldsymbol{r}_s = \left(r_{s,n}(v), r_{s,\mathrm{b}}(vw)\right)^{\top}_{n \in \mp, (v,vw) \in \mathcal{G}_s}$ of *deterministic* SFC-RD gathers the computing $(r_{s,\mathrm{c}}(v))$, memory $(r_{s,\mathrm{m}}(v))$, wireless $(r_{s,\mathrm{w}}(v))$, and bandwidth $(r_{s,\mathrm{b}}(vw))$ resources of the VNFs and virtual links of an SFC of slice s.

The *random* vector $\mathbf{U}_{s,k} = \left(U_{s,n,k}(v), U_{s,\mathrm{b},k}(vw)\right)^{\top}_{n \in \mp, (v,vw) \in \mathcal{G}_s}$ represents the User-RD (U-RD) of slice s during time slot k. Each user of slice s is assumed to consume a random proportion of the resources of an SFC of that slice. The consumed resources by various users are represented by independently and identically distributed random vectors. $U_{s,n,k}(v)$, $n \in \Upsilon$ and $U_{s,\mathrm{b},k}(vw)$ are the random amount of employed resources of VNF instance v and of virtual link vw.

The *random* vector $\mathbf{R}_{s,k} = \left(R_{s,n,k}(v), R_{s,\mathrm{b},k}(vw)\right)^{\top}_{n \in \Upsilon, (v,vw) \in \mathcal{G}_s}$ represents the Slice-RD (S-RD) of slice s during time slot k. Its components $R_{s,n,k}(v)$, $n \in \Upsilon$, and $R_{s,\mathrm{b},k}(vw)$ aggregate the amount of resources employed by a random number $N_{s,k}$ of independent users of slice s in time slot k.

The characteristics of U-RD and S-RD may change at the beginning of each time slot k and are then assumed constant during the time slot.

One considers, for a typical user and during a given time slot k, that the resource demands of different types for a given node $v \in \mathcal{N}_s$ are correlated. A correlation also exists between the demands for resources of the same type among virtual nodes and between resource demands of different types. The resulting traffic demands between nodes is usually also correlated with the resource demands for a given virtual node, as reported in [13]. Considering the U-RD vector $\mathbf{U}_{s,k}$, one assumes that the elements $U_{s,n,k}(v)$, $\forall n \in \Upsilon$, and $U_{s,\mathrm{b},k}(vw)$ are normally distributed during k. $\mathbf{U}_{s,k}$ thus follows a multivariate normal distribution $f\left(\mathbf{x}; \boldsymbol{\mu}_{s,k}, \boldsymbol{\Gamma}_{s,k}\right)$ with mean

$$\boldsymbol{\mu}_{s,k} = \left[\overline{U}_{s,n,k}(v), \overline{U}_{s,\mathrm{b},k}(vw)\right]_{n\in\Upsilon,(v,vw)\in\mathcal{G}_s} \quad (1)$$

and covariance matrix $\boldsymbol{\Gamma}_{s,k}$ with

$$\mathrm{diag}\left(\boldsymbol{\Gamma}_{s,k}\right) = \left[\widetilde{U}^2_{s,n,k}(v), \widetilde{U}^2_{s,\mathrm{b},k}(vw)\right]_{n\in\Upsilon,(v,vw)\in\mathcal{G}_s} \quad (2)$$

and off-diagonal elements representing the correlation between different types of resource demands. One has thus $U_{s,n,k}(v) \sim \mathcal{N}\left(\overline{U}_{s,n,k}(v), \widetilde{U}^2_{s,n,k}(v)\right)$, with $n \in \Upsilon$ and $U_{s,\mathrm{b},k}(vw) \sim \mathcal{N}\left(\overline{U}_{s,\mathrm{b},k}(vw), \widetilde{U}^2_{s,\mathrm{b},k}(vw)\right)$.

Assume that the number of users $N_{s,k}$ to be supported by slice s is described by the pmf

$$p_\eta = \Pr\left(N_{s,k} = \eta\right). \quad (3)$$

Since the amount of resources of the VNF v and of the virtual link vw consumed by different users is represented by independently and identically distributed copies of $\mathbf{U}_{s,k}$, the joint distribution of the aggregate amount of resources consumed by η independent users is $f\left(\mathbf{x}, \eta\boldsymbol{\mu}_{s,k}, \eta^2\boldsymbol{\Gamma}_{s,k}\right)$. The total amount of resources employed by a random number $N_{s,k}$ of independent users, $\mathbf{R}_{s,k} = \left(R_{s,n,k}(v), R_{s,\mathrm{b},k}(vw)\right)^\top_{n\in\Upsilon,(v,vw)\in\mathcal{G}_s}$, is distributed as

$$g\left(\mathbf{x}, \boldsymbol{\mu}_{s,k}, \boldsymbol{\Gamma}_{s,k}\right) = \sum_{\eta=0}^{\infty} p_\eta f\left(\mathbf{x}, \eta\boldsymbol{\mu}_{s,k}, \eta^2\boldsymbol{\Gamma}_{s,k}\right). \quad (4)$$

III. Slice Resource Provisioning

The slice resource provisioning is represented by a mapping between the infrastructure graph $\mathcal{G}$ and the S-RD graph $\mathcal{G}_s$.

The number of VNF instances of type $v \in \mathcal{N}_s$ that node i will be able to host during time slot k is $\kappa_{s,k}(i,v) \in \mathbb{N}$. The amount of resource of type $n \in \Upsilon$ provisioned by node i for a VNF instance of type v in time slot k is $\kappa_{s,k}(i,v)\, r_{s,n}(v)$. Similarly, let $\kappa_{s,k}(ij,vw)\, r_{s,\mathrm{b}}(vw)$ be the bandwidth provisioned by link ij to support the traffic between virtual nodes of type v and w during time slot k, with $\kappa_{s,k}(ij,vw) \in \mathbb{N}$.

The node mapping indicator function is $\widetilde{\kappa}_{s,k}(i,v) = 1$ if $\kappa_{s,k}(i,v) > 0$ and $\widetilde{\kappa}_{s,k}(i,v) = 0$ otherwise. Moreover, $\widetilde{\kappa}_k(i) \in \{0,1\}$ indicates whether the infrastructure node i is used for the slice resource provisioning in time slot k.

For a given time slot k, these decision variables are gathered in the vector $\boldsymbol{\kappa}_k = \left\{\boldsymbol{\kappa}_{s,k}\right\}_{s\in\mathcal{S}^+_{k|k}}$, with $\boldsymbol{\kappa}_{s,k} = \left\{\kappa_{s,k}(i,v), \kappa_{s,k}(ij,vw), \widetilde{\kappa}_{s,k}(i,v), \widetilde{\kappa}_k(i)\right\}_{(i,ij)\in\mathcal{G},(v,vw)\in\mathcal{G}_s}$.

A solution of the provisioning problem for slice s for time slot k is thus defined by an assignment of $\boldsymbol{\kappa}_{s,k}$. This assignment has to satisfy some constraints to ensure a proper behavior of the slice and the compliance with the MI-SLA defined in terms of SSP $\underline{p}_s$ accounting for the aggregate user demands, see Section III-A.

A. Provisioning Constraints

This section describes the constraints for the provisioning problem of slices of the active set $\mathcal{S}^+_{k|k}$ for each time slot k.

1) Fundamental Constraints: An assignment of the variables in $\boldsymbol{\kappa}_{s,k}$ must satisfy

$$\sum_{s,v} \kappa_{s,k}(i,v)\, r_{s,n}(v) \leqslant a_n(i), \forall i,n, \quad (5)$$

$$\sum_{s,vw} \kappa_{s,k}(ij,vw)\, r_{s,\mathrm{b}}(vw) \leqslant a_{\mathrm{b}}(ij), \forall ij, \quad (6)$$

$$\sum_{j\in\mathcal{N}} \left[\kappa_{s,k}(ij,vw) - \kappa_{s,k}(ji,vw)\right] = \left(\frac{r_{s,\mathrm{b}}(vw)}{\sum_{vu\in\mathcal{E}_s} r_{s,\mathrm{b}}(vu)}\right)\kappa_{s,k}(i,v) - \left(\frac{r_{s,\mathrm{b}}(vw)}{\sum_{uw\in\mathcal{E}_s} r_{s,\mathrm{b}}(uw)}\right)\kappa_{s,k}(i,w), \forall s,i,vw, \quad (7)$$

$$\kappa_{s,k}(i,v) \leq \widetilde{\kappa}_{s,k}(i,v) < \kappa_{s,k}(i,v) + 1, \forall s,i,v, \quad (8)$$

$$\sum_{s,v} \frac{\widetilde{\kappa}_{s,k}(i,v)}{|\mathcal{N}|\,|\mathcal{N}_s|} \leq \widetilde{\kappa}_k(i) < \sum_{s,v} \frac{\widetilde{\kappa}_{s,k}(i,v)}{|\mathcal{N}|\,|\mathcal{N}_s|} + 1, \forall i, \quad (9)$$

$$p_{s,k}\left(\boldsymbol{\kappa}_{s,k}\right) \geqslant \underline{p}_s, \forall s. \quad (10)$$

Constraints (5) and (6) ensure that the resources provisioned for all slices $s \in \mathcal{S}^+_{k|k}$ do not exceed the available resources in each infrastructure node $i \in \mathcal{N}$ and link $ij \in \mathcal{E}$. A flow conservation constraint similar to that introduced in [12] is described in (7). Constraints (8)-(9) ensure that the relation between $\kappa_{s,k}(i,v)$ (real), $\widetilde{\kappa}_{s,k}(i,v)$ (binary), and $\widetilde{\kappa}_k(i)$ are respected.

Finally, constraint (10) imposes that an assignment $\boldsymbol{\kappa}_{s,k}$ satisfying (5)-(7), also meets the minimum SSP $\underline{p}_s$ for every slice $s \in \mathcal{S}^+_{k|k}$. The SSP is the probability that an assignment is compliant with the constraints imposed for slice s and by the infrastructure, and is evaluated as

$$p_{s,k}\left(\boldsymbol{\kappa}_{s,k}\right) = \Pr\left\{\sum_i \kappa_{s,k}(i,v)\, r_{s,n}(v) \geqslant R_{s,n,k}(v), \forall v,n, \sum_{ij} \kappa_{s,k}(ij,vw)\, r_{s,\mathrm{b}}(vw)\, R_{s,\mathrm{b},k}(vw), \forall vw\right\} \quad (11)$$

The constraint (10) is nonlinear due to the need to evaluate $p_{s,k}\left(\boldsymbol{\kappa}_{s,k}\right)$ using (11). Using the approach introduced in [5], one can replace the SSP constraint (10) by the following linear deterministic constraints, for each slice s,

$$\sum_{i} \kappa_{s,k}(i,v)\, r_{s,n,k}(v) \geqslant \widehat{R}_{s,n,k}(v,\gamma_{s,k})\, d_{s,k}, \forall v, n, \quad (12)$$

$$\sum_{ij} \kappa_{s,k}(ij,vw)\, r_{s,\mathrm{b},k}(vw) \geqslant \widehat{R}_{s,\mathrm{b},k}(vw,\gamma_{s,k})\, d_{s,k}, \forall vw \quad (13)$$

where $d_{s,k}$ is a binary variable indicating whether resources are actually provisioned for slice s at time slot k and

$$\widehat{R}_{s,n,k}(v,\gamma_s) = \overline{R}_{s,n,k}(v) + \gamma_{s,k}\widetilde{R}_{s,n,k}(v), \quad (14)$$

$$\widehat{R}_{s,\mathrm{b},k}(vw,\gamma_s) = \overline{R}_{s,\mathrm{b},k}(vw) + \gamma_{s,k}\widetilde{R}_{s,\mathrm{b},k}(vw), \quad (15)$$

are the target aggregate user demands, depending on some parameter $\gamma_{s,k} > 0$. $\overline{R}_{s,n,k}(v)$ and $\widetilde{R}_{s,n,k}(v)$ are the mean and standard deviation of $R_{s,n,k}(v)$, while $\overline{R}_{s,\mathrm{b},k}(vw)$ and $\widetilde{R}_{s,\mathrm{b},k}(vw)$ are the mean and standard deviation of $R_{s,\mathrm{b},k}(vw)$. Their evaluation is detailed in [5].

B. Provisioning Approaches

At each time slot, the InP serves new slice provisioning requests and possibly updates the provisioning schemes of in-service slices. The total provisioning cost of the InP is mainly due to the provisioning of new slices and to the update of already active slices. Hence, for each slice request, we determine (*i*) the amount of infrastructure resources involved in the provisioning, (*ii*) the number of VNF instances required to handle the S-RD of that slice, and (*iii*) the total number of used instances. The operational cost for each newly processed slice request is the cost of provisioning resources for an entire slice, whereas the operational cost for each in-service slice is related to updates required for that slice. The objective of slice resource provisioning is to maximize the earnings of the InP corresponding to the slice income minus the total provisioning costs. In what follows, three different provisioning approaches are proposed.

The *quasi-static* approach only considers the active slice set $\mathcal{S}^+_{k|k}$. No prior nor posterior information on the provisioning of the slices is taken into account. The *myopic* approach considers the active slice set $\mathcal{S}^+_{k|k}$. Moreover, to reduce VNF migrations, it also takes as input the provisioning assignment of any slice $s \in \mathcal{S}^+_{k|k}$ that has been evaluated in the previous time slot, *i.e.*, $\left\{\boldsymbol{\kappa}_{s,k} : s \in \mathcal{S}^+_{k|k} \cap \mathcal{S}^+_{k\text{-}1|k-1}\right\}$. Finally, the *foresighted* approach additionally accounts for the slice requests $s \in \mathcal{S}_k$ that have to be activated in future time slots $k+1, k+2, \ldots$

1) Quasi-Static Provisioning: In this approach, the criterion to optimize is

$$E_{\mathrm{st}}(\boldsymbol{\kappa}_k) = \sum_{s \in \mathcal{S}^+_{k|k}} I_s d_{s,k} - C_{\mathrm{st}}(\boldsymbol{\kappa}_k), \quad (16)$$

where I_s is the income of the InP for a slice s whose MI-SLA is satisfied and

$$C_{\mathrm{st}}(\boldsymbol{\kappa}_k) = \sum_i \widetilde{\kappa}_k(i)\, c_{\mathrm{f}}(i) + \sum_{s \in \mathcal{S}^+_{k|k}} C_{\mathrm{st}}(\boldsymbol{\kappa}_{s,k}),$$

is the total provisioning cost, where

$$C_{\mathrm{st}}(\boldsymbol{\kappa}_{s,k}) = \sum_{i,v,n} \kappa_{s,k}(i,v)\, r_n(v)\, c_n(i) + \kappa_{s,k}(ij,vw)\, r_{\mathrm{b}}(vw)\, c_{\mathrm{b}}(ij) + \sum_{i,v} \widetilde{\kappa}_{s,k}(i,v)\, c_{\mathrm{d}}(i). \quad (17)$$

The first term of $C_{\mathrm{st}}(\boldsymbol{\kappa}_k)$ represents the cost of using infrastructure nodes. The first and second terms of (17) indicate the total cost for leasing resources from infrastructure nodes and links, and the last term accounts for the deployment of one or several VNF instances on infrastructure nodes. In (17), the cost of possible readjustments of VNF instances, *i.e.*, changes in $\kappa_{s,k}(i,v)$ in different time slots k, is neglected.

The *quasi-static* resource provisioning problem for all slices $s \in \mathcal{S}^+_{k|k}$ is then formulated as the following ILP problem.

Problem 1: Quasi-Static Slice Resource Provisioning

$$\underset{\boldsymbol{d}_k, \boldsymbol{\kappa}_k}{\text{maximize}} \sum_{s \in \mathcal{S}^+_{k|k}} I_s d_{s,k} - C_{\mathrm{st}}(\boldsymbol{\kappa}_k), \quad (18)$$

subject to $(5-9, 12-13)$.

2) Myopic Provisioning: In the myopic approach, the criterion to optimize is now

$$E(\boldsymbol{\kappa}_k) = \sum_{s \in \mathcal{S}^+_{k|k}} I_s d_{s,k} - C_{\mathrm{my}}(\boldsymbol{\kappa}_k | \boldsymbol{\kappa}_{k\text{-}1}), \quad (19)$$

where

$$C_{\mathrm{my}}(\boldsymbol{\kappa}_k | \boldsymbol{\kappa}_{k\text{-}1}) = \sum_i \widetilde{\kappa}_t(i)\, c_{\mathrm{f}}(i) + \sum_{s \in \mathcal{S}^+_{k|k}} C_{\mathrm{my}}(\boldsymbol{\kappa}_{s,k} | \boldsymbol{\kappa}_{s,k\text{-}1})$$

accounts for the provisioning scheme $\boldsymbol{\kappa}_{k\text{-}1}$ performed during time slot $k-1$. Moreover

$$C_{\mathrm{my}}(\boldsymbol{\kappa}_{s,k} | \boldsymbol{\kappa}_{s,k\text{-}1}) = \sum_{s \in \mathcal{S}^+_{k|k}} \Bigg[\sum_{i,v,n} \kappa_{s,k}(i,v)\, r_n(v)\, c_n(i) + \sum_{ij,vw} \kappa_{s,k}(ij,vw)\, r_{\mathrm{b}}(vw)\, c_{\mathrm{b}}(ij) + \sum_{i,v} \max\left\{\kappa_{s,k}(i,v) - \kappa_{s,k\text{-}1}(i,v), 0\right\} c_{\mathrm{a}}(v) + \sum_{i,v} \left(1 - \widetilde{\kappa}_{s,k\text{-}1}(i,v)\right) \widetilde{\kappa}_{s,k}(i,v)\, c_{\mathrm{d}}(i,v) \Bigg]. \quad (20)$$

The criterion (19) differs from (17) in the last two terms of 20. The fourth term, $\sum_{i,v} \max\left\{\kappa_{s,k}(i,v) - \kappa_{s,k\text{-}1}(i,v), 0\right\} c_{\mathrm{a}}(v)$, accounts the cost of readjusting VNF instances $v \in \mathcal{N}_s$ on infrastructure nodes $i \in \mathcal{N}$. The fifth term, $\sum_{i,v} \left(1 - \widetilde{\kappa}_{s,k\text{-}1}(i,v)\right) \widetilde{\kappa}_{s,k}(i,v)\, c_{\mathrm{d}}(i)$ accounts for the cost of using a new infrastructure node i to provision resources for any instance v, *i.e.*, this cost is not counted if resources are provision for a VNF instance on an infrastructure node that was previously used by the same instance.

$$C_{\text{fo,1}}\left(\boldsymbol{\kappa}_k, \boldsymbol{\kappa}_{k+1}|\boldsymbol{\kappa}_{k\text{-}1}\right) = \sum_{s\in\mathcal{S}^+_{k|k}} C_{\text{my}}\left(\boldsymbol{\kappa}_{s,k}|\boldsymbol{\kappa}_{s,k\text{-}1}\right) + \lambda \sum_{s\in\mathcal{S}^+_{k+1|k}} C_{\text{my}}\left(\boldsymbol{\kappa}_{s,k+1}|\boldsymbol{\kappa}_{s,k}\right)$$
$$= \sum_i c_{\text{f}}\left(i\right)\left[\widetilde{\kappa}_t\left(i\right) + \lambda\widetilde{\kappa}_{k+1}\left(i\right)\right] + \sum_{i,v,n} c_n\left(i\right) r_n\left(v\right)\Big(\sum_{s\in\mathcal{S}^+_{k|k}} \kappa_{s,k}\left(i,v\right) + \sum_{s\in\mathcal{S}^+_{k+1|k}} \lambda\kappa_{s,k+1}\left(i,v\right)\Big) + \sum_{ij,vw} c_{\text{b}}\left(ij\right) r_{\text{b}}\left(vw\right)\Big(\sum_{s\in\mathcal{S}^+_{k|k}} \kappa_{s,k}\left(ij,vw\right) +$$
$$\sum_{s\in\mathcal{S}^+_{k+1|k}} \lambda\kappa_{s,k+1}\left(ij,vw\right)\Big) + \sum_{s\in\mathcal{S}^+_{k|k}}\sum_{i,v} c_{\text{d}}\left(i\right)\left(1-\widetilde{\kappa}_{s,k\text{-}1}\left(i,v\right)\right)\widetilde{\kappa}_{s,k}\left(i,v\right) + \sum_{s\in\mathcal{S}^+_{k+1|k}}\sum_{i,v} \lambda c_{\text{d}}\left(i\right)\widetilde{\kappa}_{s,k+1}\left(i,v\right) - \sum_{s\in\mathcal{S}^+_{k+1|k}\cap\mathcal{S}^+_{k|k}}\sum_{i,v} \lambda c_{\text{d}}\left(i\right)\underbrace{\widetilde{\kappa}_{s,k}\left(i,v\right)\widetilde{\kappa}_{s,k+1}\left(i,v\right)}_{\text{quadratic term}}. \tag{21}$$

The fourth term of (20) makes the myopic provisioning cost nonlinear. To linearize this term, one introduces $z_{k\text{-}1,k}\left(i,v\right) \in \{0,1\}$ and replaces the term $\max\left\{\kappa_{s,k}\left(i,v\right)-\kappa_{s,k\text{-}1}\left(i,v\right),0\right\}$ in (20) by an additional variable $x_{k\text{-}1,k}\left(i,v\right)$. Imposing $0\leqslant\kappa_{s,k}\left(i,v\right)\leqslant\kappa_{\text{U}}$, $\forall i\in\mathcal{N}$, $\forall v\in\mathcal{N}_s$, and for all time slots k, leads to $-\kappa_{\text{U}}\leqslant\kappa_{s,k}\left(i,v\right)-\kappa_{s,k\text{-}1}\left(i,v\right)\leqslant\kappa_{\text{U}}$. The following constraints should be satisfied to make the substitution of the max constraint valid

$$x_{k\text{-}1,k}\left(i,v\right) \leqslant \kappa_{\text{U}} z_{k\text{-}1,k}\left(i,v\right), \tag{22}$$
$$x_{k\text{-}1,k}\left(i,v\right) \leqslant \kappa_{s,k}\left(i,v\right) - \kappa_{s,k\text{-}1}\left(i,v\right) - \left(-\kappa_{\text{U}}\right)\left(1-z_{k\text{-}1,k}\left(i,v\right)\right) \tag{23}$$
$$x_{k\text{-}1,k}\left(i,v\right) \geqslant 0, \tag{24}$$
$$x_{k\text{-}1,k}\left(i,v\right) \geqslant \kappa_{s,k}\left(i,v\right) - \kappa_{s,k\text{-}1}\left(i,v\right). \tag{25}$$

The upper bound κ_{U} of $\kappa_{s,k}\left(i,v\right)$ is the maximum number of VNF instances that the S-RD at node v requires, and can be calculated by $\kappa_{\text{U}} = \frac{\widehat{R}_{s,n,k}(v,\gamma_s)}{r_{s,n}(v)}$, with a sufficiently large γ_s.

The *myopic* resource provisioning problem for all slices $s\in\mathcal{S}^+_{k|k}$ can be formulated as the following ILP problem.

Problem 2: ILP Myopic Provisioning

$$\underset{\boldsymbol{d}_k,\boldsymbol{\kappa}_k}{\text{maximize}} \sum_{s\in\mathcal{S}^+_{k|k}} I_s d_{s,k} - C_{\text{my}}\left(\boldsymbol{\kappa}_k|\boldsymbol{\kappa}_{k\text{-}1}\right), \tag{26}$$

subject to (5–9, 12–13, 22–25).

3) Foresighted Provisioning: As mentioned at the beginning of Section III-B, in the foresighted approach, for a given time slot k, the provisioning solution for the previous time slot $\boldsymbol{\kappa}_{k\text{-}1}$ and posterior information (predicted slice demand variations and the new slice demand for the next time slots) are taken into account.

Consider first the one-step ahead foresighted provisioning approach in which, for time slot k, one accounts for the already received provisioning requests for slices that needs to be activated in the future time slot $k+1$. One denotes the set for those slice requests as $\mathcal{S}^+_{k+1|k}$, $\mathcal{S}^+_{k+1|k} \triangleq \left\{s\in\mathcal{S}_k : k\text{+}1\in\left[k_s^{\text{on}},k_s^{\text{off}}\right]\right\}$. Note that $\mathcal{S}^+_{k+1|k}\subset\mathcal{S}^+_{k+1|k+1} \triangleq \left\{s\in\mathcal{S}_{k+1} : k\text{+}1\in\left[k_s^{\text{on}},k_s^{\text{off}}\right]\right\}$.

The one step ahead foresighted objective function

$$C_{\text{fo,1}}\left(\boldsymbol{\kappa}_k, \boldsymbol{\kappa}_{k+1}|\boldsymbol{\kappa}_{k\text{-}1}\right) = \sum_i \widetilde{\kappa}_t\left(i\right) c_{\text{f}}\left(i\right) + \sum_{s\in\mathcal{S}^+_{k|k}} C_{\text{my}}\left(\boldsymbol{\kappa}_{s,k}|\boldsymbol{\kappa}_{s,k\text{-}1}\right) + \lambda \sum_{s\in\mathcal{S}^+_{k+1|k}} C_{\text{my}}\left(\boldsymbol{\kappa}_{s,k+1}|\boldsymbol{\kappa}_{s,k}\right). \tag{27}$$

involves two myopic cost functions (20), one accounting for the provisioning of slices in $\mathcal{S}^+_{k|k}$ for the considered time slot k and a second (discounted by the factor λ) for the provisioning of the slices $\mathcal{S}^+_{k+1|k}$ active in the following time slot $k+1$. Similarly, the N steps ahead foresighted cost function, $C_{\text{fo},N}$, is given by

$$C_{\text{fo},N}\left(\boldsymbol{\kappa}_k, \cdots, \boldsymbol{\kappa}_{k+N}|\boldsymbol{\kappa}_{k\text{-}1}\right) = \sum_i \widetilde{\kappa}_t\left(i\right) c_{\text{f}}\left(i\right) + \sum_{s\in\mathcal{S}^+_{k|k}} C_{\text{my}}\left(\boldsymbol{\kappa}_{s,k}|\boldsymbol{\kappa}_{s,k\text{-}1}\right) + \sum_{i=1}^{N}\sum_{s\in\mathcal{S}^+_{k+i|k}} \lambda_i C_{\text{my}}\left(\boldsymbol{\kappa}_{s,k+i}|\boldsymbol{\kappa}_{s,k+(i\text{-}1)}\right), \tag{28}$$

where λ_i is the discount factor associated with the myopic cost function $C_{\text{my}}\left(\boldsymbol{\kappa}_{s,k+i}|\boldsymbol{\kappa}_{s,k+(i\text{-}1)}\right)$. The cardinality of the set of variables of $C_{\text{fo},N}$ is approximately N times that of $C_{\text{fo,1}}$, possibly leading to an intractable problem. Therefore, in this paper, one only considers the one-step ahead foresighted approach.

The expression of $C_{\text{fo,1}}$ in (27) could be rewritten as in (21). It is observed that the last term of $C_{\text{fo,1}}$ has a *quadratic* form, hence an optimization problem using $C_{\text{fo,1}}$ as objective will be a Mixed Integer Quadratic Problem (MIQP), which is substantially more difficult than MILP [14]. In [15], a technique is introduced to linearize product of binary variables. For two binary variables $\widetilde{\kappa}_{s,k}\left(i,v\right)$ and $\widetilde{\kappa}_{s,k+1}\left(i,v\right)$, this technique replaces $\widetilde{\kappa}_{s,k}\left(i,v\right)\widetilde{\kappa}_{s,k+1}\left(i,v\right)$ by an additional variable $y_{s,k,k+1}\left(i,v\right)\in[0,1]$ and introduces the following Fortet constraints

$$y_{s,k,k+1}\left(i,v\right) \leqslant \widetilde{\kappa}_{s,k}\left(i,v\right), \forall i,v, \tag{29}$$
$$y_{s,k,k+1}\left(i,v\right) \leqslant \widetilde{\kappa}_{s,k+1}\left(i,v\right), \forall i,v, \tag{30}$$
$$y_{s,k,k+1}\left(i,v\right) \geqslant \widetilde{\kappa}_{s,k}\left(i,v\right) + \widetilde{\kappa}_{s,k+1}\left(i,v\right) - 1, \forall i,v, \tag{31}$$

for all $s\in\mathcal{S}^+_{k+1|k}\cap\mathcal{S}^+_{k|k}$ (slices for which resources need to be provisioned in both time slot k and $k+1$). Replacing $\widetilde{\kappa}_{s,k}\left(i,v\right)\widetilde{\kappa}_{s,k+1}\left(i,v\right)$ in the last term of (21) by $y_{s,k,k+1}\left(i,v\right)$, the cost $C_{\text{fo,1}}\left(\boldsymbol{\kappa}_k, \boldsymbol{\kappa}_{k+1}|\boldsymbol{\kappa}_{k\text{-}1}\right)$ is transformed to $C_{\text{fo,1}}\left(\boldsymbol{\kappa}_k, \boldsymbol{\kappa}_{k+1}, \boldsymbol{y}_{k,k+1}|\boldsymbol{\kappa}_{k\text{-}1}\right)$, where $\boldsymbol{y}_{k,k+1} = \left\{y_{s,k,k+1}\right\}_{s\in\mathcal{S}^+_{k+1|k}\cap\mathcal{S}^+_{k|k}}$.

The one-step ahead foresighted slice provisioning problem is then described as in Problem 3.

IV. Evaluation

In this section, one evaluates via simulations the performance of the three provisioning algorithms described in

Problem 3: MILP One-Step Ahead Foresighted Provisioning

$$\underset{\boldsymbol{d}_k, \boldsymbol{\kappa}_k, \boldsymbol{y}_{k,k+1}}{\text{maximize}} \quad \sum_{s \in \mathcal{S}_k^+} I_s d_{s,k} - C_{\text{fo},1}\left(\boldsymbol{\kappa}_k, \boldsymbol{\kappa}_{k+1}, \boldsymbol{y}_{k,k+1} | \boldsymbol{\kappa}_{k\text{-}1}\right), \quad (32)$$

$$\text{subject to} \quad (5\text{–}9, 12\text{–}13, 22\text{–}25) \text{ for } \boldsymbol{\kappa}_{s,k}, \forall s \in \mathcal{S}_k^+, \quad (33)$$

$$(5\text{–}9, 12\text{–}13, 22\text{–}25) \text{ for } \boldsymbol{\kappa}_{s,k+1}, \forall s \in \mathcal{S}_{k+1|k}^+, \quad (34)$$

$$(29)\text{–}(31) \text{ for } \boldsymbol{y}_{s,k,k+1}, \forall s \in \mathcal{S}_{k+1|k}^+ \cap \mathcal{S}_k^+. \quad (35)$$

Section III-B, namely `static`, `myopic`, and `foresight`, by solving respectively Problem 1, 2, and 3. The simulation setup is described in Section IV-A. All simulations are performed with the CPLEX MILP solver interfaced with MATLAB.

A. Simulation Conditions

1) Infrastructure Topology: The infrastructure network is generated from a binary fat tree topology, as in [16, 17]. A typical binary fat-tree topology is depicted in Figure 2. The leaf nodes represent the Remote Radio Heads (RRHs). The other nodes represent the edge, regional, and central datacenters. Infrastructure nodes and links provide a given amount of computing, storage, and possibly wireless resources $(a_{\text{c}}, a_{\text{m}}, a_{\text{w}})$, expressed in number of CPUs, Gbytes, and Gbps, depending on the layer they are located. The per-unit resource cost ($c_n\,(i)$, $\forall n \in \Upsilon$ and $c_{\text{b}}\,(ij)$), the node disposal cost $c_{\text{f}}\,(i)$, and the fixed cost $c_{\text{d}}\,(i)$ are set respectively to 1, 50, and 20, $\forall\,(i, ij) \in \mathcal{G}$.

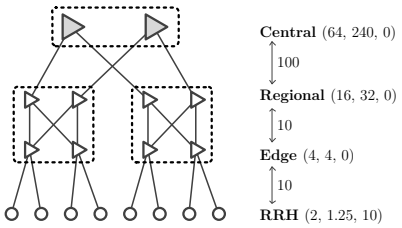
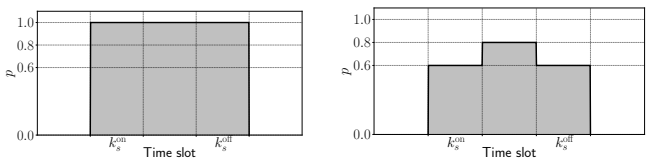


Fig. 2. Description of a binary fat-tree infrastructure network; Nodes provide a given amount of computing a_{c}, memory a_{m}, and wireless a_{w} resources expressed in number of used CPUs, Gbytes, and Gbps; Links are able to transmit data at a rate a_{b} expressed in Gbps.

In each time slot k, we assume that a random part of the infrastructure resources are partly consumed by best-effort background services, see [5]. The performance of the proposed provisioning variants are evaluated considering different network conditions in terms of resource consumption by background services.

Under *unsaturated* (normal) network conditions, the resource consumption of background services is set to 20% of the available resources of each infrastructure node $i \in \mathcal{N}$ and link $ij \in \mathcal{E}$. Under *saturated* network conditions, the resource consumption of background services is set to 50%.

2) Slice Resource Demand (S-RD): The number of users of a slice s is assumed to follow a binomial distribution of parameter $p_{s,k}$. Two patterns are used to represent the temporal evolution of $p_{s,k}$. The first, illustrated in Figure 3a corresponds to a constant demand $p_{s,k} = 1$ during the whole lifetime of the slice. The second, shown in Figure 3b, describes a slice whose demand evolves from one time slot to the next.



(a) (b)

Fig. 3. Probability pattern of service usage: (a) constant over a time interval and (b) piece-wise constant.

Three types of slices are considered.

Slices of type 1 aim to provide an HD video streaming service at average rate of 4 Mbps for VIP users, *e.g.*, in a stadium. The number of users follows a binomial distribution $\mathcal{B}\,(300, 0.9)$. The function architecture of slices of type 1 is composed of 3 VNFs: a virtual Video Optimization Controller (vVOC), a virtual Gateway (vGW), and a virtual Base Band Unit (vBBU). The required SSP for type 1-slices is $\underline{p}_s^{\text{sp}} = 0.99$.

Slices of type 2 are dedicated to provide an SD video streaming service at average rate of 2 Mbps. The number of users follows a binomial distribution $\mathcal{B}\,(1000, 0.8)$. The function architecture of slices of type 2 is similar to that of type 1. The required SSP for type 2-slices is $\underline{p}_s^{\text{sp}} = 0.95$.

Slices of type 3 aim to provide a video surveillance and traffic monitoring service at average rate of 1 Mbps for 100 cameras, *e.g.*, installed along a highway. The third slice type consists of five virtual functions: a vBBU, a vGW, a virtual Traffic Monitor (vTM), a vVOC, and a virtual Intrusion Detection Prevention System (vIDPS). The required SSP for type 3-slices is $\underline{p}_s^{\text{sp}} = 0.9$.

The slice type is chosen uniformly at random. For slices of type 1 and 2, the demand pattern is also chosen uniformly at random. The number of provisioning request arrivals in each time slot obeys a Poisson distribution $\text{Pois}\,(\mu)$ of parameter $\mu = 1$. The arrival time of each slice request is uniformly distributed within each time slot. The activation delay (*i.e.*, $k_s^{\text{on}} - k_s$) follows the uniform distribution $\mathcal{U}\,(1, 4)$ and the lifetime follows the uniform distribution $\mathcal{U}\,(2, 6)$. The discount factor λ in the `foresight` approach is set to 0.6.

Details of each resource type as well as the associated U-RD, SFC-RD, and S-RD parameters are given in Table I. Numerical values in Table I have been adapted from [18].

B. Results

This section illustrates the performance of the three resource provisioning variants (`static`, `myopic`, `foresight`), with the following metrics: (*i*) utilization of infrastructure nodes and links, (*ii*) provisioning cost, (*iii*) total earnings of the InP, (*iv*) acceptance rate, given by $\sum_{s \in \mathcal{S}_{k|k}^+} \frac{d_{s,k}}{\left|\mathcal{S}_{k|k}^+\right|}$, which is the

TABLE I
PARAMETERS OF U-RD, SFC-RD, AND S-RD.

Type 1: HD video streaming at 4 Mbps.

Node	$\overline{U}_{s,\mathrm{c}}$	$\overline{U}_{s,\mathrm{m}}$	$\overline{U}_{s,\mathrm{w}}$	$(r_\mathrm{c}, r_\mathrm{m}, r_\mathrm{w})$
vVOC	5.4e-3	1.5e-2	—	$(0.29, 0.81, 0)$
vGW	9.0e-4	5.0e-4	—	$(0.05, 0.03, 0)$
vBBU	8.0e-4	5.0e-4	4e-3	$(0.04, 0.03, 0.2)$

Links: $\overline{U}_{s,\mathrm{b}}(vw) =$ 4e-3 and $r_{s,\mathrm{b}} = 0.22$, $\forall vw$

Type 2: SD video streaming at 2 Mbps.

Node	$\overline{U}_{s,\mathrm{c}}$	$\overline{U}_{s,\mathrm{m}}$	$\overline{U}_{s,\mathrm{w}}$	$(r_\mathrm{c}, r_\mathrm{m}, r_\mathrm{w})$
vVOC	1.1e-3	7.5e-3	—	$(0.17, 1.20, 0)$
vGW	1.8e-4	2.5e-4	—	$(0.03, 0.04, 0)$
vBBU	0.8e-4	2.5e-4	2e-3	$(0.01, 0.04, 0.3)$

Links: $\overline{U}_{s,\mathrm{b}}(vw) =$ 2e-3 and $r_{s,\mathrm{b}} = 0.32$, $\forall vw$

Type 3: Video surveillance and traffic monitoring at 1 Mbps.

Node	$\overline{U}_{s,\mathrm{c}}$	$\overline{U}_{s,\mathrm{m}}$	$\overline{U}_{s,\mathrm{w}}$	$(r_\mathrm{c}, r_\mathrm{m}, r_\mathrm{w})$
vBBU	2.0e-4	1.3e-4	1e-3	$(0.4, 0.25, 2)$ e-2
vGW	9.0e-4	1.3e-4	—	$(0.018, 0.003, 0)$
vTM	1.1e-3	1.3e-4	—	$(0.266, 0.003, 0)$
vVOC	5.4e-3	3.8e-3	—	$(0.108, 0.080, 0)$
vIDPS	1.1e-2	1.3e-4	—	$(0.214, 0.003, 0)$

Links: $\overline{U}_{s,\mathrm{b}}(vw) =$ 1e-3 and $r_{s,\mathrm{b}} = 0.02$, $\forall vw$

percentage of slices that have successfully been provisioned, and (v) the number of VNF instances needed to be redeployed, *i.e.*, the number of nodes $v \in \mathcal{N}_s, \forall s \in \mathcal{S}^+_{k|k}$ such that $\exists i \in \mathcal{N} : \{\widetilde{\kappa}_{s,k\text{-}1}(i,v), \widetilde{\kappa}_{s,k}(i,v)\} = \{1,0\}$.

For all provisioning variants, the earnings and provisioning cost are evaluated following the myopic expressions (19, 20).

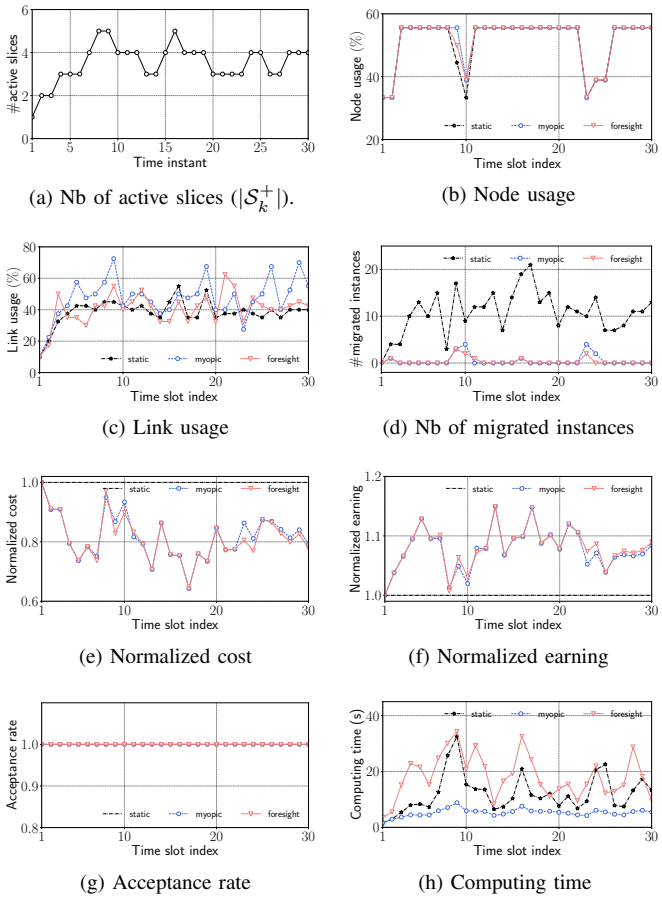
In this section, the performance of the three variants under unsaturated and saturated network condition are evaluated. Except the distribution of the background services, both scenarios use the same simulation setup. All provisioning variants are examined over 30 time slots. The provisioning costs and earnings of the `myopic` and `foresight` variant are normalized with those of the `static` variant.

Figure 4a shows the number of active slices, *i.e.*, $\left|\mathcal{S}^+_{k|k}\right|$, for each time slot.

a) Unsaturated Scenario: The performance of the three variants under the unsaturated network condition are shown in Figure 4. The average node usage, link usage, provisioning cost, earnings, computing time per time slot, and the total number of redeployed VNF instances of the three variants are shown in Table IIa.

Under the unsaturated network condition, all provisioning approaches are capable of provisioning all slices in all time slots, as depicted in Figure 4g. In Figure 4b, one can see that all variants perform similarly with respect to the utilization of infrastructure nodes. Nevertheless, the `static` variant requires a much higher number of VNF instances to be redeployed, see Figure 4d), thus yielding a higher provisioning cost compared to the `myopic` and `foresight` schemes (see Figure 4e), leading to lower earnings for the InP, as shown in Figure 4f.

In general, the `foresight` scheme performs the best among the three variants, in almost all metrics, see Table IIa. This is due to the fact that, by exploiting both the previous provisioning solution and the predicted future resource demand, the `foresight` scheme requires less redeployment of VNF instances than the other schemes (see Figure 4d), thus allowing to achieve the provisioning cost and the highest earnings, as shown in Figures 4e and 4f.



(a) Nb of active slices ($|\mathcal{S}^+_k|$). (b) Node usage

(c) Link usage (d) Nb of migrated instances

(e) Normalized cost (f) Normalized earning

(g) Acceptance rate (h) Computing time

Fig. 4. Performance comparison of three variants in unsaturated scenario, in terms of (a) utilization of infrastructure nodes, (b) utilization of infrastructure links, (c) number of redeployed VNF instances, (d) normalized provisioning cost, (e) normalized earning, (f) acceptance rate.

Nevertheless, as expected, the `foresight` provisioning method is more time-consuming than the other methods (see Figure 4h), since its number of variables, $\{\boldsymbol{\kappa}_k, \boldsymbol{\kappa}_{k+1}, \boldsymbol{y}_{k,k+1}\}$, is more than twice that of the `static` and `myopic` method, which only considers the variables in $\boldsymbol{\kappa}_k$. Using the prior provisioning results, *i.e.*, $\boldsymbol{\kappa}_{k\text{-}1}$, as initialization, allows the `myopic` method to be more efficient in terms of computing time than the `static` method.

b) Saturated Scenario: A similar results can be observed when considering saturated network conditions, as shown in Figure 5 and Table IIb. The `myopic` and `foresight` methods achieve the same acceptance rate as the `static` method, even under limited resource conditions.

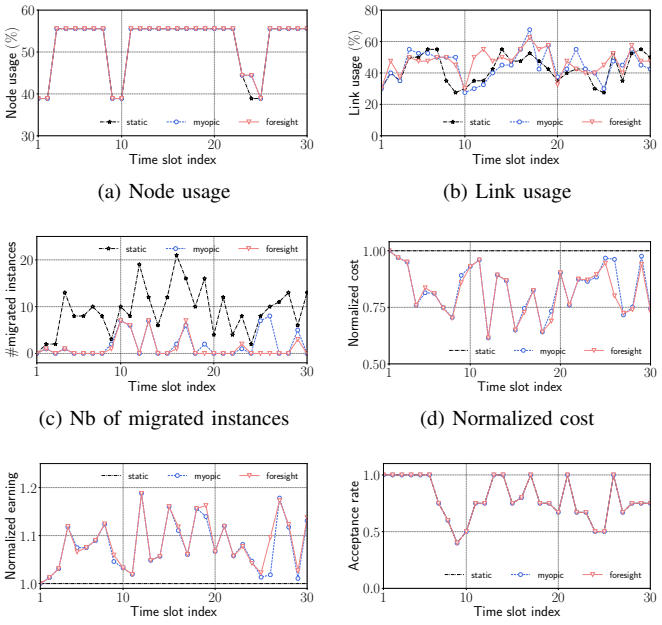
TABLE II
SUMMARIZED RESULTS OF 3 VARIANTS

(a) **Unsaturated network condition**

Criteria	`static`	`myopic`	`foresight`
Average node usage	51.1%	51.7%	51.5%
Average link usage	38.4%	47.2%	40.4%
Average cost	1016.7	824.3	817.0
Average earning	2483.3	2675.7	2683.0
Average time	12.1s	5.3s	18.2s
Tot. redeployed instances	325	15	10

(b) **Saturated network condition**

Criteria	`static`	`myopic`	`foresight`
Average node usage	51.9%	52%	52%
Average link usage	42.5%	44.7%	46.9%
Average cost	846.4	692.9	683.6
Average earning	1820.3	1973.8	1983.1
Average time	7.9s	5.1s	11.5s
Tot. redeployed instances	275	55	36



(a) Node usage (b) Link usage

(c) Nb of migrated instances (d) Normalized cost

(e) Normalized earning (f) Acceptance rate

Fig. 5. Performance comparison of three variants in saturated scenario, in terms of (a) utilization of infrastructure nodes, (b) utilization of infrastructure links, (c) number of redeployed VNF instances, (d) normalized provisioning cost, (e) normalized earning, (f) acceptance rate.

V. CONCLUSIONS

In this paper, different resource provisioning approaches for network slicing are investigated. Adopting the perspective of the InP and using MILP formulations, we propose two provisioning variants, namely *myopic* and *foresighted*, which objective to maximize the amount of slices for which resources can be successfully provisioned, and consequently the earnings of the InP.

A *quasi-static* provisioning method is used as a baseline for the proposed methods. Numerical results demonstrate the advantages of these methods, which are (*i*) the ability to adapt to the dynamic nature of slice provisioning requests; and (*ii*) robustness to the uncertainties raised by the fluctuation of resource demands. In future work, we aim to design efficient heuristics by using alternative techniques such as *eigendecompostion* [19] or *column generation* [20] to scale up to more realistic setups, with numerous slice provisioning requests and large-sized network.

REFERENCES

[1] R. Su, D. Zhang, R. Venkatesan, Z. Gong, C. Li, F. Ding, F. Jiang, and Z. Zhu, "Resource Allocation for Network Slicing in 5G Telecommunication Networks: A Survey of Principles and Models," *IEEE Netw.*, vol. 33, no. 6, pp. 172–179, 2019.

[2] X. Li, M. Samaka, H. A. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain, "Network Slicing for 5G: Challenges and Opportunities," *IEEE Internet Computing*, 2018.

[3] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G Network Slicing Using SDN and NFV: A Survey of Taxonomy, Architectures and Future Challenges," *Computer Networks*, vol. 167, 2020.

[4] M. Richart, J. Baliosian, J. Serrat, and J. L. Gorricho, "Resource Slicing in Virtual Wireless Networks: A Survey," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 462–476, 2016.

[5] Q.-T. Luu, S. Kerboeuf, and M. Kieffer, "Uncertainty-Aware Resource Provisioning for Network Slicing," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 79–93, 2021.

[6] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, "On Dynamic Service Function Chain Deployment and Readjustment," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 543–553, 2017.

[7] A. Fendt, C. Mannweiler, L. C. Schmelz, and B. Bauer, "An Efficient Model for Mobile Network Slice Embedding under Resource Uncertainty," in *Proc. ISWCS*, 2019, pp. 602–606.

[8] G. Wang, G. Feng, T. Q. S. Quek, S. Qin, R. Wen, and W. Tan, "Reconfiguration in Network Slicing-Optimizing the Profit and Performance," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 2, pp. 591–605, 2019.

[9] N. V. Huynh, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz, "Real-Time Network Slicing with Uncertain Demand : A Deep Learning Approach," in *Proc. IEEE ICC*, 2019.

[10] C. Liang and F. R. Yu, "Wireless Network Virtualization: A Survey, Some Research Issues and Challenges," *IEEE Commun. Surveys Tuts.*, pp. 1–24, 2014.

[11] J. Wang, K. L. Wright, and K. Gopalan, "XenLoop: A Transparent High Performance Inter-VM Network Loopback," *Cluster Comput.*, vol. 12, no. 2 SPEC. ISS., pp. 141–152, 2009.

[12] Q.-T. Luu, S. Kerboeuf, A. Mouradian, and M. Kieffer, "A Coverage-Aware Resource Provisioning Method for Network Slicing," *IEEE/ACM Trans. Netw.*, vol. 28, no. 6, pp. 2393–2406, 2020.

[13] C. Jiang, G. Han, J. Lin, G. Jia, W. Shi, and J. Wan, "Characteristics of Co-Allocated Online Services and Batch Jobs in Internet Data Centers: A Case Study from Alibaba Cloud," *IEEE Access*, vol. 7, pp. 22 495–22 508, 2019.

[14] E. Eiben, R. Ganian, and S. Ordyniak, "Solving Integer Quadratic Programming via Explicit and Structural Restrictions," in *Proc. AAAI*, 2019, pp. 1477–1484.

[15] R. Fortet, "Applications de l'Algebre de Boole en Recherche Opérationelle," *Revue Française de Recherche Opérationelle*, vol. 4, no. 14, pp. 17–26, 1960.

[16] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed, and T. Ahmed, "Scheduling Wireless Virtual Networks Functions," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 2, pp. 240–252, 2016.

[17] N. Bouten, R. Mijumbi, J. Serrat, J. Famaey, S. Latre, and F. De Turck, "Semantically Enhanced Mapping Algorithm for Affinity-Constrained Service Function Chain Requests," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 2, pp. 317–331, 2017.

[18] M. Savi, M. Tornatore, and G. Verticale, "Impact of Processing-Resource Sharing on the Placement of Chained Virtual Network Functions," in *Proc. IEEE NFV-SDN*, 2016, pp. 191–197.

[19] M. Mechtri, C. Ghribi, and D. Zeghlache, "A Scalable Algorithm for the Placement of Service Function Chains," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 533–546, 2016.

[20] G. Nemhauser, "Column Generation for Linear and Integer Programming," *Optimization Stories*, vol. I, pp. 65–73, 2012.