Sciancalepore, V., Costa- Pérez, X. y Banchs, A. (2019). RL-NSB: Reinforcement Learning-based 5G Network Slice Broker. *IEEE Transactions on Networking*, 27(4), pp. 1543 - 1557.

DOI: https://doi.org/10.1109/TNET.2019.2924471

RL-NSB: Reinforcement Learning-based 5G Network Slice Broker

Vincenzo Sciancalepore, *Member, IEEE,* Xavier Costa-Perez, *Senior Member, IEEE,*
Albert Banchs, *Senior Member, IEEE*

***Abstract*—Network slicing is considered one of the main pillars of the upcoming 5G networks. Indeed, the ability to slice a mobile network and tailor each slice to the needs of the corresponding tenant is envisioned as a key enabler for the design of future networks. However, this novel paradigm opens up to new challenges such as isolation between network slices, the allocation of resources across them and the admission of resource requests by network slice tenants. In this work, we address this problem by designing the following building blocks for supporting network slicing: i) traffic and user mobility analysis, ii) a learning and forecasting scheme per slice, iii) optimal admission control decisions based on spatial and traffic information, and iv) a reinforcement process to drive the system towards optimal states. In our framework, namely *RL-NSB*, infrastructure providers perform admission control considering the Service Level Agreements (SLA) of the different tenants as well as their traffic usage and user distribution, and enhance the overall process by means of learning and reinforcement techniques that consider heterogeneous mobility and traffic models among diverse slices. Our results show that, by relying on appropriately tuned forecasting schemes, our approach provides very substantial potential gains in terms of system utilization while meeting the tenants' SLAs.**

I. Introduction

MOBILE users and their respective data traffic are expected to grow excessively in the next years touching an eight-fold increased compared to 2015 [1]. This uncountable explosion opens to new remunerative business models while bringing vertical segments into play, such as automotive digital factories (with the novel concept of industry 4.0), smart cities, e-health with regular advanced multimedia services and ultra-high definition video. Indeed, 3GPP has released the first 5G networks guidelines providing support for critical communications, massive machine type communications and vehicular-to-everything [2]. Among the others, the core feature of the fifth generation of mobile networks is the possibility of hosting on the same infrastructure different services with possibly conflicting requirements. This would require a flexible network architecture that builds on i) the network virtualization paradigm allowing for the on-demand introduction of very diverse services in a shared infrastructure, and ii) the novel concept of *network slicing* [3], which allows to deploy different (virtualized) network instances of different services.

V. Sciancalepore, X. Costa-Perez are with NEC Laboratories Europe GmbH ({name.surname}@neclab.eu). A. Banchs is with IMDEA Networks Institute and University Carlos III of Madrid, Spain (albert.banchs@imdea.org).

The work of V. Sciancalepore and X. Costa-Perez was supported by the European Union H-2020 Project 5G-TRANSFORMER under Grant Agreement 761536. The work of A. Banchs was supported in part by the 5GCity project of the Spanish Ministry of Economy and Competitiveness (TEC2016-76795-C6-3-R).

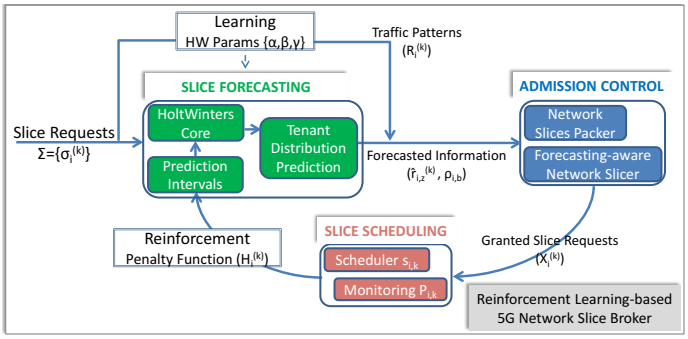


Fig. 1: RL-NSB: a holistic view.

The network slicing paradigm allows infrastructure providers (such as mobile network operators, MNOs) to open their physical network facilities to multiple tenants through the instantiation of logical self-contained networks, orchestrated in different ways depending on the tenants' specific service requirements; such network slices are (temporarily) owned and managed by the respective tenants. In contrast to classical resource provisioning, network slicing resource management deals with aggregated multi-flow resource guarantees per slice instead of single per-flow guarantees which are still expected to be handled by schedulers.

In the above context, network tenants may issue requests for network slices with associated (networking and computational) resources thereby calling for an advanced admission control that preserves the service quality to already-running network slices. To this end, the 5G Network Slice Broker [4] is envisioned as a novel network element that builds on the capacity broker functional block considered by 3GPP for advanced RAN sharing [5]. This element maps incoming Service Level Agreement (SLA) requirements associated to network slice requests into physical resources. The architectural specifications for this new network paradigm are currently under definition and the necessary algorithms yet to be devised.

When considering network slice requests, very conservative criteria may be followed for mission critical services that need ultra-high availability, while more aggressive criteria may be applied in other cases, leveraging multiplexing gains of traffic among slices and thus optimize network utilization and monetization. To this end, the ability to predict the actual *footprint* of a particular network slice is essential to increase the number of slices that might be running on the same infrastructure without harming their performance.

Building on the above idea, in this paper we design three key network slicing building blocks: i) a *forecasting* module that predicts network slices' traffic based on past information regarding user demands and mobility patterns, ii) a network

slicing *admission control* algorithm and iii) a network slicing *scheduler* algorithm in charge of meeting the agreed SLAs and report back deviations to the forecasting module.

The remaining of the paper is organized as follows. In Section II we review the state-of-the-art solutions, before presenting our framework building blocks in Section III. In Section IV we establish the basis of our slice forecasting model, whereas in Section V we formulate the admission control problem as a geometric knapsack, showing that this problem is NP-Hard. In Section VI we study the slice scheduling process and analyze how its feedback is used to adjust the forecasting process. In Section VII we discuss the simulation results and, finally, we conclude the paper in Section VIII.

II. Related Work

The time-series prediction topic has been exhaustively investigated in the past providing a number of practical solutions applied on different fields. In [6], the authors propose to use a multi-smoothing function to predict short- and long-term traffic windows. Similarly, [7] applies the concept of Bayesian neural networks to predict the number of active UEs in an LTE network so as to efficiently assign resources. Conversely, in our solution we apply known forecasting techniques to profile the slice traffic behaviours and user mobility while still performing an efficient slice resource allocation.

A RAN sharing solution applying the proportional fairness criterion is proposed in [8]. To share resources among different operators under diverse radio conditions, [9] introduces the Network Virtualization Substrate (NVS), a two-step process where the infrastructure first allocates resources to the virtual instances of eNBs and then each tenant customizes scheduling within its eNB instance [10]. A network slicing solution considering a gateway-based approach is proposed in [11]. In such solution, a controller provides application-oriented resource abstraction of the underlying RAN, and allocates resources to network slices based on an optimization framework that considers an elasticity margin in resource provisioning. Similarly to the last-mentioned work, we adopt a similar two-step process, allocating slices via a broker entity that performs admission control based on the requested SLAs.

Our approach builds on the concept of a signaling-based network slicing broker solution by implementing a capacity forecasting algorithm [12] that considers guaranteed and best-effort traffic as well as user mobility. A study that explores different options for network sharing based on a centralized broker is provided in [13], considering mobility means, spectrum transfer policies and resource virtualization to optimize the usage of MNO's limited resources. Unlike our proposal, this study introduces new 3GPP interfaces to accommodate the broker functionality. A scheme that integrates the capacity broker with enhancements on the 3GPP architecture is documented in [14]. Such capacity broker forecasts the network capacity when allocating guaranteed and best-effort slices, considering their respective SLAs. Our approach complements this solution by introducing algorithms that dynamically evaluate network slices SLA requests, while maximizing the infrastructure resources utilization.

The work in [15] assigns a different slice to each operator implementing the sharing of network resources among operators by dynamically allocating slice resources. Furthermore, a dynamic slicing scheme that accounts for tenants' priority, baseband resources, fronthaul and backhaul capacities, quality of service (QoS) and interference within the context of heterogeneous cloud radio access network architecture is proposed in [16] as well as in [17].

In contrast to all these approaches, our work focuses on the admission control and traffic forecasting to enable mobile network operators (MNOs) to make the best out of their spare resources.

III. RL-NSB: System Design

This paper builds on the concept of a 5G network slice broker for establishing network slices through a proper signaling, studied in [4], in the context of the 3GPP network sharing management architecture [18]. The concept of broker is envisioned to become part of the future 5G network architecture, introducing the required new interfaces into the architecture[1].

Fig. 1 depicts the Reinforcement Learning-based 5G Network Slice Broker (RL-NSB) building blocks addressed in this paper and their interactions. The design of the different modules is addressed in detail in the following sections.

Slice Forecasting. This module evaluates the tenants' traffic near-future. In addition, knowledge about the tenant users (spatial) distribution might further increase the efficiency of the admission process, as explained in Section IV. When no forecasting solution is applied or during the *training period (for adjusting the forecasting algorithm parameters), original network slice SLA request information is used.*

Admission Control. This module selects the granted network slice requests for the next time window based on two different algorithms, as illustrated in Section V.

Slice Scheduling. A list of granted slice requests is sent to the *Scheduling* module, which allocates network slice physical resources and monitors the served traffic levels and potential SLA violations with a penalty history function. Such a function is used to provide a feedback signal to the forecasting module and to adaptively adjust the system behaviour. The design of this module is addressed in Section VI.

IV. Slice Forecasting

Forecasted traffic patterns provide a useful information to predict base stations load for the different network tenants. This allows to properly dimension network slices and thus maximize the overall system resource utilization across different cells. This involves predicting the resource usage of the different tenants in time as well as in space: by forecasting the load at different cells, the RL-NSB framework can properly assign cell resources to different tenants while leaving unused resources in under-utilized cells for further slice requests. The effectiveness of such an approach highly depends on the accuracy of the forecasting algorithm: the more accurate, the more aggressive the resource provisioning while still keeping low the probability of violating slice SLAs. In the following we address the first aspect, while we refer the reader to Section VI for more details on SLA violations and dynamic forecasting parameters adjustments.

[1]We refer the reader to [19] that provides an overall picture of an implementable mobile architecture shedding the light on potential issues while including the broker entity into the new standard architecture.

A. Tenant traffic analysis: characterization and forecasting

Traffic predictions are performed on an aggregate basis for every admitted tenant. Tenants might ask for a different network slice request tailored to their specific service requirements without specifying the set of physical cells users will be lying in. Indeed, a given vertical tenant might need only a small subset of cells to provide selected services to its own users, e.g., automotive may need only cells covering extra-urban roads. The idea is that the forecasting process categorizes the traffic requests based on the associated service requirements and (geographical) location, thereby performing a prediction separately per cell and per slice. In our analysis, we first assume that traffic requests are uniformly distributed within the whole network. However, in Section IV-B and Section IV-C we further extend this assumption by considering multi-cellular environments where user mobility follows different patterns for different tenants, yielding heterogeneous user distributions across tenants.

We assume the following traffic model. We assume different classes of traffic based on specific SLAs, as shown in Table I. We let the traffic volumes of tenant i for traffic class k (e.g., satisfying particular service requirements) be a realization of a point process, $\zeta_i^{(k)} = \sum_{t=0}^{T} \delta_t \sum_{b\in\mathcal{B}} r_{i,b}^{(k)}(t)$, where δ_t denotes the Dirac measure for sample t. We express traffic requests $r_{i,b}^{(k)}(t)$ per cell b in terms of required resources (note that such resource requirements could be easily translated into different metrics, such as latency or throughput demands) and the aggregate traffic requests as $r_i^{(k)} = \sum_{b\in\mathcal{B}} r_{i,b}^{(k)}$ for each tenant i.

The underlying key-assumption in our model is that traffic requests follow a periodic pattern, which is needed to apply time-series forecasting algorithms. Given such a periodic nature, the traffic forecasting is based on an observed time window T_{OBS}, and is given by the vector $\mathbf{r_i^{(k)}} = (r_i^{(k)}(t-T_{\text{OBS}}), r_i^{(k)}(t-(T_{\text{OBS}}-1)), \cdots, r_i^{(k)}(t))$. Then, given a fixed future time window T_{FUTURE}, the forecasting function f_{HW} provides the forecasted traffic volumes for time period $[t+1, t+T_{\text{FUTURE}}]$, denoted as $\hat{\mathbf{r}}_\mathbf{i}^{(\mathbf{k})} = (\hat{r}_i^{(k)}(t+1), \hat{r}_i^{(k)}(t+2), \cdots, \hat{r}_i^{(k)}(t+T_{\text{FUTURE}}))$. Intuitively, the longer the observed time window T_{OBS}, the more information to rely on, the higher the accuracy of the traffic forecasting within the next time window T_{FUTURE} per tenant slice. In our analysis we fix the observed time window given the periodic nature of the network traffic requests regardless the corresponding service requirements [20].

Following our assumption above, the system exhibits a periodic behavior, which translates into seasons of length W_S that are repeated over time. Within a single season, we assume that process $\zeta_i^{(k)}$ is stationary and ergodic [2]. Such a process is evaluated and predicted through a seasonal exponential smoothing function. To this end, we use the Holt-Winters (HW) forecasting procedure to analyze and predict future traffic requests associated to a particular network slice across all selected cells (as shown in the next section). The forecasting function f_{HW} is defined as:

$$f_{HW}: \mathbb{R}^{T_{\text{OBS}}+1} \rightarrow \mathbb{R}^{T_{\text{FUTURE}}}$$
$$\mathbf{r_i^{(k)}} \rightarrow \hat{\mathbf{r}}_\mathbf{i}^{(\mathbf{k})}.$$

We denote a specific predicted traffic request $\hat{r}_i^{(k)}(t)$ by $\hat{r}_{i,t}^{(k)}$. We rely on the additive version of the HW forecasting problem as the seasonal effect does not depend on the mean traffic level of the observed time window but instead it is added considering values predicted through level and trend effects. Following HW standard procedure and assuming a frequency of the seasonality (W) based on the traffic characteristics, we can predict such requests based on the level l_t, trend b_t and seasonal s_t factors, as follows:

$$\begin{aligned} \hat{r}_{i,t+T_{\text{FUTURE}}}^{(k)} &= l_t + b_t T_{\text{FUTURE}} + s_{t+T_{\text{FUTURE}}-W(\kappa+1)} \quad \text{where} \\ l_t &= \alpha(r_{i,t}^{(k)} - s_{t-W}) + (1-\alpha)(l_{t-1} + b_{t-1}), \\ b_t &= \beta(L_t - l_{t-1}) + (1-\beta) b_{t-1}, \\ s_t &= \gamma(r_{i,t}^{(k)} - l_{t-1} - b_{t-1}) + (1-\gamma) s_{t-W}. \end{aligned} \tag{1}$$

While κ is the integer part of $(T_{\text{FUTURE}}-1)/W$ and the set of optimal HW parameters α, β and γ can be obtained during a training period employing existing techniques [22], we focus on the forecasting errors and how the forecasting inaccuracy may affect our network slicing solution. In other words, inaccurate forecasted traffic values (for instance, lower values) might lead the admission control to accommodate more network slices that cannot fit the overall system capacity resulting in service degradation and, in turn, in SLA violation.

We define the one-step training forecasting error $e_{i,t}^{(k)}$ as follows

$$e_{i,t}^{(k)} = r_{i,t}^{(k)} - \hat{r}_{i,t}^{(k)} = r_{i,t}^{(k)} - (l_{t-1} + b_{t-1} + s_{t-1}), \tag{2}$$

which is computed during the training period of our forecasting algorithm (when predicted values are compared with the observed ones). Given that our process $\zeta_i^{(k)}$ is ergodic and assuming an optimal HW parameter set, for any predicted value at time z we can derive the prediction interval $\left[\hat{ll}_{i,z}^{(k,\chi)}, \hat{hh}_{i,z}^{(k,\chi)}\right]$ wherein future traffic requests lie with a certain probability $\chi_i^{(k)}$ for that particular network slice.

Thus, it holds that

$$Pr\left\{\hat{ll}_{i,z}^{(k,\chi)} \leq \hat{r}_{i,z}^{(k)} \leq \hat{hh}_{i,z}^{(k,\chi)}\right\} = \chi_i^{(k)}, \forall z \in [t+1, t+T_{\text{FUTURE}}] \tag{3}$$

where $\hat{hh}_{i,z}^{(k,\chi)}$ (or $\hat{ll}_{i,z}^{(k,\chi)}$) $= \hat{r}_{i,z}^{(k)} + (-)\Omega_\chi \sqrt{Var(e_{i,z}^{(k)})}$ and

$$Var(e_{i,z}^{(k)}) \approx \left((1 + (z-1)\alpha^2[1 + z\beta + \frac{z(2z-1)}{6}\beta^2]\right)\sigma_e^2.$$

In the above equation, Ω_χ denotes the one-tailed value of a

TABLE I: Network slice traffic requirements [23]

k	$T^{(k)}$	Type and QCI
0	10 ms	GBR - 65
1	50 ms	GBR - 3
2	100 ms	GBR - 1
3	150 ms	GBR - 2
4	300 ms	non-GBR - 6
5	1000 ms	non-GBR - 9

[2] Stationary and ergodicity of point process (c.f. e.g. [21]) imply that $\bar{\mu}_i^{(k)} = \frac{1}{Z}\sum_{z=0}^{Z} X[z] = \frac{1}{T}\sum_{t=0}^{T} r_{i,k}[t]$.

standard normal distribution such that we obtain $\chi_i^{(k)}$ probability and σ_e^2 is the variance of one-step training forecasting error, i.e., $\sigma_e^2 = Var(e_{i,t}^{(k)})$, over the observed time window.

Due to the requirements imposed by traffic SLAs, we focus only on the upper bound of the prediction interval as it provides the "worst-case" of a forecasted traffic level. It can be seen from (3) that a larger prediction time window T_{FUTURE} yields a higher number of predicted values z (spread over fixed time intervals), and this in turn leads to a lower accuracy according to the above equations; the intuition is that the further into the future we need to predict, the higher the uncertainty. Even if accuracy is high, if we set the forecasting error probability $\chi_i^{(k)}$ too low, this can result in severe penalties in case it does not guarantee the desired slice SLAs. Therefore, we adjust the forecasting error probability $\chi_i^{(k)}$ based on the service requirements and to the number of prediction points the forecasting process needs to perform.

Following the above, best-effort traffic requests with no stringent requirements can tolerate a prediction with a longer time pace that results in imprecise values. This makes the upper bound $\hat{hh}_{i,z}^{(k,\chi)}$ very close to the real (future) values $r_{i,z}^{(k,\chi)}$ regardless the error probability $\chi_i^{(k)}$ as the number of z values to predict is limited. Hence, we might select a low forecasting error probability $\chi_i^{(k)}$ for this service type. On the other hand, when guaranteed bit rate traffic is considered, the corresponding SLA must be fulfilled in a shorter time basis, which makes our forecasting process much more complex, requiring significantly more predicted values z. To achieve this, our system models such a type of traffic with a higher forecasting error probability $\chi_i^{(k)}$.

We implement the above mathematically as follows. According to the traffic classes defined in Table I, traffic class $k = 0$ provides a forecasted horizon shorter than the other traffic classes, and hence a larger number of values z must be predicted. To achieve this, we derive an upper bound for the forecasting probability error per tenant for this traffic class. We define the maximum potential gain between the slice request and the forecasted traffic requests as $\hat{d}_i^{(k)} \doteq \max_{z \in \text{T}_{\text{FUTURE}}} \left(R_i^{(k)} - \hat{r}_{i,z}^{(k)} \right)$. We then compute the forecasting error probability as follows

$$\chi_i^{(k=0)} : \Omega_\chi \sqrt{Var(e_{i,z}^{(k=0)})} = \hat{d}_i^{(k=0)}. \tag{4}$$

As soon as the potential gain $\hat{d}_i^{(k=0)}$ becomes very large, we cap the one-tailed value Ω_χ to 3.49, resulting in $\chi_i^{(k=0)} = 99.9\%$. Conversely, for the best-effort traffic ($k = 5$) we compute the forecasting error probability $\chi_i^{(k=|\mathcal{K}|)} = 50\%$, due to its more relaxed service. For the other traffic classes k, intermediate forecasting error probabilities $\chi_i^{(k)}$ are calculated from (4) by deriving $\hat{d}_i^{(k)}$ values from the upper and the lower bound values. In addition to the above, note that forecasting error probability values are dynamically evaluated and adjusted based on the SLA violations experienced during the slice scheduling process, as explained in detail in Section VI-B.

B. User mobility and traffic model periodicity

We next extend our forecasting model to dynamic scenarios where user mobility is considered and the traffic periodicity assumption may no longer hold. The Hold-Winters method used above cannot be applied unless the underlaying system is periodic. To overcome this issue, in the following we devise an approximation to the traffic load which is i) as close as possible to the original traffic load, and yet ii) is periodical.

We consider a multi-cellular environment covering the whole area. In order to design forecasting algorithms that are accurate under realistic settings, we rely on human-based mobility patterns. Specifically, we employ the well-accepted SLAW mobility model [24] for user motions. According to this model, users move among a number of waypoints, which are distributed over the covered area according to self-similarity rules forming a number of clusters. Clusters with more waypoints can be seen as hotspots attracting more users. When performing a flight (a movement from one waypoint to the other within the same trip), users choose a set of clusters which are dynamically and randomly replaced during the flight based on some given probabilities. Then, users start moving between a subset of waypoints residing within the selected clusters according to a least-action trip planning (LATP) with $\alpha_{\text{SLAW}} = 3$. Traffic is randomly generated during the user trip. Assuming that users stop when reaching a waypoint for a pause-time, we can model the value of the flight-time (x_L) and pause-time (x_P) as a random value drawn from a heavy-tailed distribution function defined in terms of Fourier transformations as

$$f_L(x) = f_P(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-iu\,x - |\psi u|^{\alpha_{\text{DISTR}}}} \mathrm{d}u \tag{5}$$

where ψ is the scale factor and α^{DISTR} depends on the distribution considered (pause-time or flight-time).

Considering a uniform user speed distribution, the traffic model of the considered users is dominated by a heavy-tailed distribution. We decouple the variation trends of the traffic model by means of the Fourier transformation as showed in (5). In this way, each component of the traffic variation is isolated so as to provide a periodic behaviour that improves the accuracy of the forecasting process (like in the previous section).Without loss of generality, we can obtain a periodic traffic vector as follows. Let M denote the period and $\mathbf{r_i^{(k)}} = \{r_t\}$ a generic traffic vector. Then, the forecasting process applies a Discrete Fourier Transform (DFT) to retrieve the M-periodic samples $R_w = \sum_{n=0}^{M-1} r_t e^{-iw\frac{2\pi}{N}t}$, where $w = 0, \cdots, M-1$. Note that R_w is a complex number translating the sinusoidal component of r_t. Then, the forecasting process can obtain all single time-series components derived by each of those frequency samples by applying the Inverse Discrete Fourier Transform (IDFT), e.g., $r_n = \frac{1}{N} \sum_{w=0}^{M-1} R_w e^{\frac{2\pi i}{N}wn}$, where $n = 0, \cdots, N-1$, which provides a periodic traffic vector $\mathbf{r_i^{(k)}} = (r_i^{(k)}(n), r_i^{(k)}(n+1), \cdots, r_i^{(k)}(n+M))$. This vector is a good approximation of the real traffic and is periodic, and hence we can use the Holt-Winters method described in the previous section over this vector to obtain a prediction of future traffic load.

C. Tenant spatial distribution

While tenants can only request a slice across the entire network area, the accurate prediction of the distribution of

the tenants' users across space can be leveraged to improve the system efficiency, by considering resource usage of the admitted slices on specific cells when taking admission control decisions. Thus, the spatial domain introduces an additional degree of freedom that can be exploited by our *tenant distribution prediction module* to provide an accurate estimation of the cells' load to the admission control module.

In the following, we extend our traffic predictor proposed above to capture not only the overall load of each tenant but also the distribution of this load over the different cells. To this end, we proceed as follows: i) we develop a Markovian chain to capture the mobility pattern of a user, ii) we assume that the mobility of a tenant is reflected by a weighted combination of such patterns, depending on the mobility patterns of the tenants' users, iii) we employ an unsupervised learning method to learn the weights of each tenant, and finally iv) we obtain the load at each cell by combining the overall load predicted by the Holt-Winters method described in the previous sections with the mobility model developed in this section.

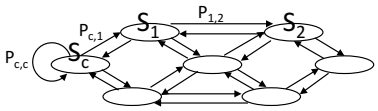
The tenant distribution prediction module is based on the probabilistic latent variable model [25]. This model relies on a Discrete-time Markov Chain $\{X_n \in \mathcal{S}\}$ with the state space $\mathcal{S} = \{S_1, \ldots, S_c, S_B\}$, where each state indicates whether a tenant user is within cell b at time n, and $p_{x,y}$ is the transition probability to move from cell x to cell y. Note that an user might remain within the same cell coverage with probability $p_{b,b}$, as shown in Fig. 2(a). However, the same Markov chain evolution cannot be applied to heterogeneous user behaviours [3]. As consequence, we derive distinct transition probability sets and we update them based on past observations about user associations.

Given that users belonging to the same tenants might visit cells following different mobility behaviours—not known a-priori—we apply the concept of *unsupervised learning* to identify specific mobility patterns showing the tenant request distributions across the network. In some cases, e.g., automotive tenants, only a single mobility pattern may be enough to draw all user motions under the same tenant control.

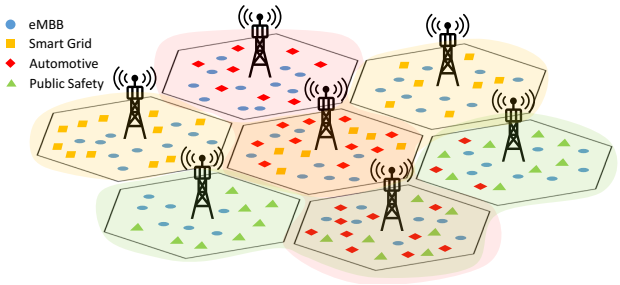
Let $m \in \mathcal{M}$ be the stochastic latent variable denoting the mobility pattern. We can reformulate the transition probability as follows

$$p^m_{x,y} = Pr(X_n = S_y \mid X_{n-1} = S_x, h_i = m), \tag{6}$$

i.e., the probability to stay at time n within cell y if the user under tenant i comes from cell x and follows mobility pattern $h_i = m$. In order to derive such transition probabilities, we use the expectation maximization [4] technique based on previous observations about the users moves for tenant i from cell x to cell y, namely $j^i(x,y)$, obtained through the user context information. We can write the a-posteriori probability



(a) Discrete-time Markov Chain prediction model



(b) Illustration of tenant distributions

Fig. 2: Spatial distribution and mobility prediction.

as follows

$$Pr(h_i = m \mid X_n = S_y, X_{n-1} = S_x) = \frac{Pr(X_n = S_y \mid X_{n-1} = S_x, h_i = m) Pr(h_i = m)}{\sum_{o \in \mathcal{M}} Pr(X_n = S_y \mid X_{n-1} = S_x, h_i = o) Pr(h_i = o)} \tag{7}$$

and the likelihood maximization of the learning problem as follows

$$Pr(X_n = S_y \mid X_{n-1} = S_x, h_i = m) = \frac{\sum_{i \in \mathcal{I}} j^i(x,y) Pr(h_i = m \mid X_n = S_y, X_{n-1} = S_x)}{\sum_{\{s_a, s_b\} \in \mathcal{S}} \sum_{i \in \mathcal{I}} j^i(a,b) Pr(h_i = m \mid X_n = S_b, X_{n-1} = S_a)} \tag{8}$$

$$Pr(h_i = m) = \frac{\sum_{\{s_a, s_b\} \in \mathcal{S}} j^i(a,b) Pr(h_i = m \mid X_n = S_b, X_{n-1} = S_a)}{j^i} \tag{9}$$

where $j^i = \sum_{\{a,b\} \in \mathcal{S}} j^i(a,b)$ denotes all user moves under tenant i within the whole network. By iteratively solving those two sets of equations with specific initial conditions [5], we can easily reach the convergence that provides i) the set of valid mobility patterns $m \in \mathcal{M}$ based on past observations, ii) associations between tenant behaviours h_i and obtained mobility patterns and, iii) the probability to stay in any of the available cells based on such mobility patterns and tenant associations. Finally, given the Markovian property of memoryless, we assign a weight to different mobility patterns m based on how accurately they can be mapped onto previous observed paths $\hat{\mathcal{S}}_i$ chosen by users under the same tenant i as follows

$$w(m \mid \hat{\mathcal{S}}_i) = \frac{\sum_{\{a,b\} \in \hat{\mathcal{S}}_i} p^m_{a,b}}{\sum_{o \in \mathcal{M}} \sum_{\{a,b\} \in \hat{\mathcal{S}}_i} p^o_{a,b}}. \tag{10}$$

[3]We rely on the Markovian property to make our analysis tractable. However, in Section VII we relax this assumption to evaluate our solution with realistic mobility scenarios, e.g., applying the well-known SLAW model.

[4]We refer the reader to [25] for further information on the unsupervised learning methods.

[5]Diverse initial conditions may lead to different optimal solutions. However, as shown in [26] the difference between such optimal solutions can be considered negligible and, in some case, they may even converge to the same identical solution. Therefore, we use a trivial initial condition to trigger the process.

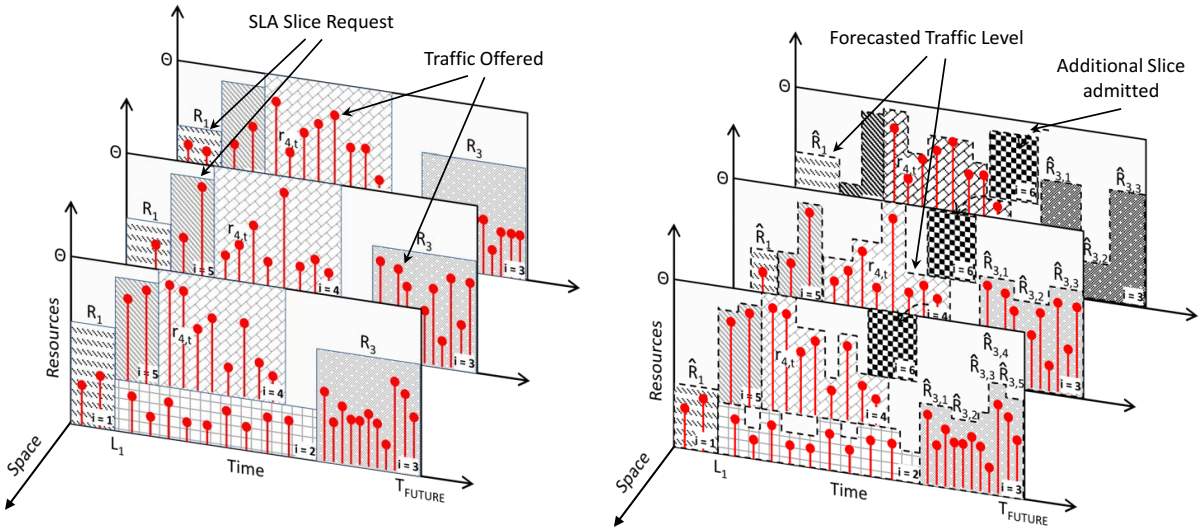


Fig. 3: Admission control problem as multiple geometric knapsack problems on three different cellular areas. On the left-hand side, full requests are shown. On the right-hand side, reshaped tenant requests are shown based on the forecasting module.

The probability to stay at time $n+1$ under coverage of cell b based on the path used by tenant i can be formulated as the following

$$\rho_{i,b} = Pr(X_{n+1}{=}S_b \mid \hat{\mathcal{S}}_i) = \sum_{a\in\mathcal{B}} \sum_{m\in\mathcal{M}} w(m|\ \hat{\mathcal{S}}_i) p^m_{a,b}. \quad (11)$$

The probabilistic latent variable model can be easily applied to unknown mobility models as it only relies on the past observations and future inferences. When the users mobility follows the SLAW model [24], we can easily reduce the complexity of our prediction by priorly setting some transition probability $p^m_{x,y} = 0$. Specifically, the SLAW model imposes that, before moving, each user must select a subset of clusters where it potentially moves to. For the sake of simplicity, we divide our network into multiple clusters wherein each cluster corresponds to a single cell. This automatically prevents users from being moved to specific cells. Additionally, based on the SLAW model, each cluster (or cell) might comprise different way-points that must be visited based on the LATP algorithm as explained in Section IV-B. Finally, the SLAW model may exhibit a pause-time for a single way-point as a random realization of a power-law statistical distribution. Therefore, the number (and the geographical positions) of way-points placed into a single cells and the pause-time within a single way-point directly impact on the probability of remaining within the same cell $p_{b,b}$.

The overall tenant load predicted by the forecasting module is properly combined with the probability of being within specific cells (Eq. (11)) so as to derive the predicted amount of resources requested by tenant i under base station b as $\hat{R}^{(k)}_{i,b,z} = \rho_{i,b}\hat{R}^{(k)}_{i,z}$. This result is used into our admission control algorithm to efficiently instantiate network slices while providing tenant SLA guarantees, as explained in the next section.

V. Admission Control: Design and Validation

We design an admission control scheme to decide on the network slice requests to be granted for the subsequent time window T_{FUTURE} based solely on the current resource availability. It relies on the key-idea of using the forecasting information to accurately reshape the resources consumed by network slice requests thereby fitting additional slice requests into the system. Additionally, users distribution information may further help the RL-NSB framework to spatially install network slices only on affected cells (see Fig. 3).

A. Problem Formulation

In our problem formulation, we first assume a constant amount of resources required for a network slice instantiation. Then, we relax this assumption—considering different forecasted traffic levels and user distributions—and show that this makes the problem more complex, but still tractable for our admission control process. Note that the latter model leads to more efficient solutions.

Let us define a network slice request as $\sigma^{(k)}_i = \{R, L, i, k\}$ [6] where i denotes the tenant, R is the amount of resources required, L is the time duration of the slice and k is the traffic class. Without loss of generality, we simply refer to a tenant request as $R^{(k)}_i(L_i)$. Recalling the main objective of accommodating the network slice requests while maximizing the network resource utilization within a fixed time window T_{FUTURE}, we next derive our model.

Let us assume a rectangular box for each base station with fixed width W and height H representing the resource availability within a fixed time window. In particular, the box width corresponds to T^b_{FUTURE} and box height corresponds to the total amount of resources Θ^b available for base station b. Let us assume a set of items $\mathcal{I}$, where each item $i \in \mathcal{I}$ corresponds to a network slice request having width w_i corresponding to slice duration L_i and height h_i corresponding to the amount of resources R_i. Abusing notation, we write $L^b_i = L_i, \forall b \in \mathcal{B}$ and

[6] Our assumption unveils that tenants may only ask for a certain amount of resources along the whole network deployment. Advanced mechanisms might admit tenants asking for network resources only on particular areas, but this is out of the scope of this paper and may be addressed in future works.

$R_i^b = R_i, \forall b \in \mathcal{B}$ as tenant requests are identically distributed along the whole network. In addition, each item provides a profit c_i; we assume that slices pay a price proportional to the resources requested, and hence c_i is proportional to the amount of resources [7]. The objective of our admission control problem is to find a subset of items $\mathcal{I}' \subseteq \mathcal{I}$ that maximizes the total profit $\sum_{i \in \mathcal{I}'} c_i$, i.e., the total amount of granted resources. The following lemma allows to decompose this problem. Formal proofs of the following lemmas are given in the Appendix (provided as Supplementary downloadable material).

Lemma 1. *Let each cell b independently assigned with a fixed amount of resources Θ^b within a given time windows T^b. The admission control problem of the overall network aiming at accommodating the network slice requests while maximizing the network resource utilization, can be obtained as the combination of independent instances of an admission control problem executed for each cell b.*

With the above lemma, we only need to focus on a single admission control problem executed on a single cell. The overall admission control solution can be reformulated as a combination of (simpler) admission control problem instances, as shown in Fig. 3. The following lemma shows that this can be mapped to a well-known NP-hard problem.

Lemma 2. *Let the resource availability of cell b be a box with height Θ^b and width T^b, and let each item $i \in \mathcal{I}$ be the network slice request σ_i with height R_i and width L_i. Then, the admission control problem is mapped onto a Geometric Two-dimensional knapsack problem with the objective of filling up the cell capacity with network slice requests while maximizing the base station resource utilization.*

The above formulation assumes that tenants use all requested resources. However, if we can estimate the actual resource usage through the prediction module proposed in the previous section, we may be able to improve efficiency. Following this, we now assume tenant requests characterized by a set $\hat{R}_{i,z}^{(k)} = \hat{hh}_{i,z}^{(k,\chi)}$ representing the predicted amount of needed resources per time z for a traffic type k (i.e., given a forecasting error probability $\chi_i^{(k)}$) based on the forecasting phase. Note that tenant traffic predictions are derived based on the aggregate traffic requests extended across the entire network for each specific tenant. Thus, to derive the predicted amount of resources requested by tenant i under base station b, we can use Eq. (11) and write $\hat{R}_{i,b,z}^{(k)} = \rho_{i,b}\, \hat{R}_{i,z}^{(k)}$. This results in time-variant resource requests where item shapes are no longer rectangular.

Lemma 3. *Let the resource availability of cell b be a box with height Θ^b and width T^b, and let each item $i \in \mathcal{I}$ be the network slice request σ_i with irregular shapes, identified by different height values $\hat{R}_{i,b,z}^{(k)}$ and width L_i. Then, the admission control problem is mapped onto a Flexibile Geometric Two-dimensional knapsack problem, with the objective of maximizing the cell resources utilization whilst accommodating network slice requests.*

An illustrative example is provided in Fig. 3 for three different cells running independently the admission control problem. As depicted, different amounts of resource values are forecasted for a single network slice request. It may be observed that when the forecasting is accurate, i.e. real traffic (pulse signal) are correctly bounded within new slice values (slice $i = 4$), there is more room to accommodate slices ($i = 6$), as shown in the right-hand side. Information on the users distribution may further help to properly reshape the network slice by considering the spatial domain as an additional degree of freedom. For example, user traffic requests belonging to tenant $i = 5$ are mostly distributed on the first two cells (in the foreground of the picture) and barely present in the last cell (in the background).

Note that in our case the (flexible) geometric two-dimensional knapsack problem is constrained by the orientation law of the considered items. In particular, each item i has a fixed orientation, which can not be changed to fit in the box. Although some state-of-the-art work calls such a problem constrained geometrical knapsack problem, we prefer to omit the "constrained" word as it may refer to additional constraints on the relationship between items stored in the box, which are out of the scope of this work.

Taking into account all the above constraints, we can formulate our overall admission control problem as follows [8]

Problem ADM-CONTROL:

$$\begin{aligned}
\text{maximize} \quad & \sum_{i \in \mathcal{I}} c_i \sum_{b \in \mathcal{B}} x_i^b \\
\text{subject to} \quad & \sum_{i \in \mathcal{I}} w_i^b \cdot x_i^b \leq W^b; \quad \forall b \in \mathcal{B} \\
& \mathcal{S}^b(x_i^b) \cap \mathcal{S}^b(x_k^b) = \emptyset, \quad \forall b \in \mathcal{B}, \forall i \neq k; \\
& \mathcal{S}^b(x_i^b) \subset \mathbb{S}^b, \qquad \forall b \in \mathcal{B}, \forall i \in \mathcal{I}; \\
& x_i^b \in \{0,1\}, \qquad \forall b \in \mathcal{B}, \forall i \in \mathcal{I};
\end{aligned}$$

where $\mathcal{S}^b(x_i^b)$ is the geometrical area of the item i (either rectangular or irregular defined) whereas $\mathbb{S}^b$ is the area of the box defined for each cell b, i.e., $|\mathbb{S}^b| = T^b \cdot \Theta^b$. Note that if the tenant is not admitted into the cell, we assume its geometrical area is zero, i.e., if $x_i^b = 0, \mathcal{S}^b(x_i^b) = \emptyset$. The first constraint refers to the weight of each item. For the sake of simplicity, we consider the weight capacity of our box as infinite $W^b = \infty$ to neglect the item weight. The next two constraints state that items cannot overlap with each other and must be contained within the total space of the box.

When tenants distribution information is available at the tenant distribution prediction module (see Fig. 1), $S^b(x_i^b)$ is accurately derived by means of $\rho_{i,b}$ (in Eq. (11)) and properly used to make more room for additional network slice requests. The solution the above problem provides a set of x_i^b, which is a binary value indicating whether the item i is admitted into the system (and allocated under cell b) or rejected for the next time window T_{FUTURE}. Thus, the solution provides a list of granted slice requests per cell (see Fig. 1).

B. Complexity Analysis

We next analyze the complexity issues of the admission control problem. To this end, we can formulate the following

[7] This assumption could be relaxed to reflect a different economic model within the multi-tenancy framework, which is out of the scope of the paper.

[8] In addition to the constraints included in our problem formulation, the Flexibile Geometric Twodimensional knapsack problem also includes an additional constraint on weight capacities. For the sake of simplicity, we have omitted this constraint in our problem.

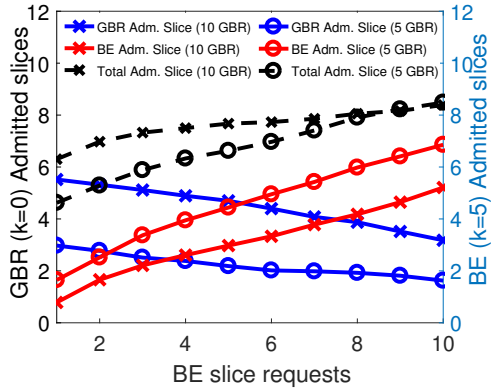


Fig. 4: Admitted slice requests within a time windows while collecting slice requests with GBR traffic requirements ($k = 0$) and best-effort traffic requirements ($k = 5$).

decision problem DEC−ADM−CONTROL: *given an arbitrary value V, n items with a value c_i and a given area a_i enclosed within a two-dimensional shape identified by $\hat{R}^{(k)}_{i,b,z}$ and L_i, and a box with capacity $\mathbb{S}^b$ delimited by Θ^b and T^b, is there a subset $\mathcal{I} \in \{1, 2, \cdots, n\}$ such that items do not overlap and $\sum_{i\in\mathcal{I}} c_i \geq V$?* In the following, we reduce this problem to an NP-hard one.

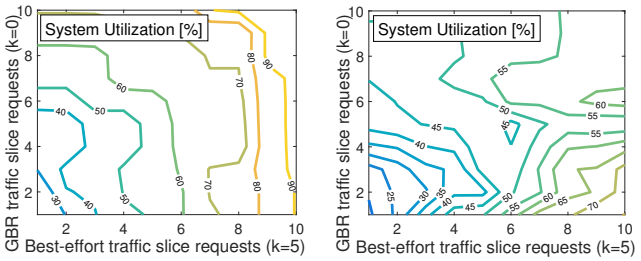
Lemma 4. *Considering all items with full flexible dimensions, we can identify one single weight w_i per item representing the area required. Then, if the utility value $c_i = w_i$ the decision problem DEC−ADM−CONTROL reduces to a "Subset Sum Problem".*

Building on the above, we can show that the admission control problem is NP-hard.

Theorem 1. *The decision problem DEC−ADM−CONTROL is NP-Complete and Problem ADM−CONTROL is NP-Hard for any type of traffic k along the network slice request.*

Sketch of Proof: We use a reduction from the subset sum problem based on Lemma 4. We apply a polynomial reduction to the decision problem DEC−ADM−CONTROL considering only items with full flexible dimensions collapsed into a weight w_i and utility value equal to the weights $c_i = w_i$. This reduces the problem to a Subset-Sum problem, known to be NP-COMPLETE. When considering items with fixed resource provisioning, e.g., items with constrained shape values, it is even more difficult to find a solution to Problem DEC−ADM−CONTROL, which proves the NP-Completeness. Based on that, for all $\epsilon > 0$, approximating the solution for Problem ADM−CONTROL, $|\mathcal{I}| = n$ within $n^{1-\epsilon}$, is NP-Hard. This proves that our Problem ADM−CONTROL is NP-Hard. □

Theorem 1 suggests that no optimal poly-time algorithm solves our admission control problem. Interestingly, we remark that the admission control problem is easier when only best-effort slice requests are processed (still NP-Hard). This could negatively drive the infrastructure provider to have a particular tendency for best-effort, or less-demanding, requirements as depicted in Fig. 4. In particular, assuming only stringent traffic class requirements GBR ($k = 0$) and best-effort (BE) class ($k = 5$), we show the number of admitted slice requests considering different resource demands to the 5G Network Slice Broker. The total number of admitted slices increases with the number of best-effort slice requests showing that best-effort slice requests are preferred due to the higher flexibility. This is further supported by Fig. 5(a), where we show the contour of the total system utilization when different number of slice requests arrive and span equally the entire network. Although a disparity between GBR and BE slice requests appears, the utilization of the system is maximized providing outstanding results (more than 90%) in the best case.



(a) $\eta = 0$; no inter-class priority (b) $\eta = 1$; inter-class priority

Fig. 5: System utilization with different utility functions.

Along these lines, we provide a smart mechanism that ensures no traffic inter-class prioritization. We define the utility value in Problem ADM−CONTROL for each slice request as $c_i = \frac{L_i R_i}{(T^{(k)})^\eta}$, with $\eta \in \{0, 1\}$. For $\eta = 0$, the utility value is exactly the amount of data required within the slice whereas $\eta = 1$ leads to more priority for strict service requirements slices. In Fig. 5(b), we show the contour of the system utilization when $\eta = 1$. While the inter-class fairness is guaranteed (as shown in the top-right part of the picture), the overall utilization degrades exhibiting values around 55% in the best case.

C. Heuristic algorithm design

Given that the admission control problem is NP-hard, in the following we derive a heuristic algorithm. As given by the formulation of the Problem ADM−CONTROL, this algorithm needs to cope with different network slice requests along multiple cells and optimize the total utility function. Network slice requests can be i) regularly shaped, i.e., no forecasted information is considered, but with different flexibility degrees due to the traffic class considered, ii) irregularly shaped exhibiting a different degree of freedom. The first class of network slice requests is handled through a Network Slices Packer algorithm, a revised and improved version of [27]. The second class of network slice problem admits at least the same solution of the first class but, if properly explored, it could provide much more flexibility and resources utilization.

Network Slices Packer. We assume rectangular shapes for network slice requests with different traffic requirements. When traffic class $k = 0$, the regular shape of the network slice is hardly defined and no flexibility is allowed for allocating the traffic requests. Conversely, when less-demanding slice requests $k > 0$ are considered, the slice might be reshaped, delaying the slice traffic, to efficiently fit into the network.

The algorithm pseudocode is given in Algorithm 1. We rely on the assumption that each tenant is not allowed to ask more than the half of the resource availability of the infrastructure provider, i.e., $R_i \leq \frac{\Theta}{2}$. This implies that at least 2 network

Algorithm 1 Network Slices Packer: Algorithm to admit network slice requests $\sigma_i^{(k)}$ within the system capacity Θ for the next time window T_{FUTURE}.

Input: $\Sigma = \{\sigma_i^{(k)}\}, \Theta, T_{\text{FUTURE}}$
Initialization: $\mathcal{C} \leftarrow \emptyset, \mathcal{F}_1 \leftarrow \emptyset, \mathcal{F}_2 \leftarrow \emptyset, \mathcal{E} \leftarrow \emptyset$
Procedure
1: **for all** $C_l \leftarrow \binom{\Sigma}{2}$ **do**
2: **if** C_l fits into $\mathbb{S}$ **then**
3: $\mathcal{C} \leftarrow \mathcal{C} \cup C_l$
4: **end if**
5: **end for**
6: **for all** $C_l \in \mathcal{C}$ **do**
7: $\{v(C_l \cup B_l), s(C_l \cup B_l)\} \leftarrow$ **Solve** Problem BIN-KP
8: **end for**
9: $l* = \arg\max_{l \in \mathcal{C}} \{v(C_l \cup B_l)\}$
10: **if** $v(C_{l*}) \geq \frac{v(C_{l*} \cup B_{l*})}{2}$ **then**
11: **return** C_{l*}
12: **else**
13: $\mathcal{F}_1 \leftarrow C_{l*}$
14: $\mathcal{F}_2 \leftarrow B_{l*}$
15: **if** $s(\mathcal{F}_1) \geq \frac{\Theta}{2}$ **then**
16: **return** B_{l*}
17: **else**
18: **Sort** $\mathcal{F}_2$ in non-increasing order of their profits and traffic class k
19: **while** $s(\mathcal{F}_1) < \frac{\Theta}{2}$ **do**
20: e = pop($\mathcal{F}_2$)
21: $\mathcal{F}_1 \leftarrow \{\mathcal{F}_1 \cup e\}$
22: **end while**
23: **if** $v(\mathcal{F}_2) \geq \frac{v(C_{l*} \cup B_{l*})}{2}$ **then**
24: **return** $\mathcal{F}_2$
25: **else**
26: $\mathcal{E} \leftarrow \arg\max\{v(\mathcal{F}_1 \setminus e); v(\mathcal{F}_2)\}$
27: **return** $\mathcal{E}$
28: **end if**
29: **end if**
30: **end if**

slices can be accommodated. Thus, following standard existing algorithms for geometric knapsack problems, we divide the set of elements into two sub-sets by considering among all possible pairs of network slice requests, only those fitting the available system capacity (line 2). For each of these 2-slice sets (C_l), we formulate the following 0-1 knapsack problem

Problem BIN-KP:

$$\begin{aligned} \text{maximize} \quad & \sum_{j \in C_l} v(\sigma_j^{(k)}) + \sum_{i \notin C_l} \omega_i v(\sigma_i^{(k)}) \\ \text{subject to} \quad & \sum_{j \in C_l} s(\sigma_j^{(k)}) + \sum_{i \notin C_l} \omega_i s(\sigma_i^{(k)}) \leq \Theta; \\ & \omega_i \in \{0,1\}, \quad \forall i \in \Sigma; \end{aligned}$$

where $v(\sigma_i^{(k)})$ denotes the profit[9] of slice $\sigma_i^{(k)}$ (in our case $R_i \cdot L_i$) whereas $s(\sigma_i^{(k)})$ is the geometrical area of slice $\sigma_i^{(k)}$. The aim of **Problem** BIN-KP is to maximize the total profit by keeping the 2-slice set (C_l) as fixed while adding additional slice requests as long as all selected items fit within the total system capacity (Θ). The solution provides a set $B_l = \{\omega_i\}$ of binary values specifying whether network slice $\sigma_i^{(k)}$ has been included into B_l. Based on the FPTAS [28], we retrieve the best solution, i.e., a set of network slice requests ($C_{l*} \cup B_{l*}$) among all binary knapsack problems BIN-KP (line 9).

If the total profit $v(\cdot)$ assigned to the 2-slice set requests C_{l*} is greater than the half of the best profit retrieved after running all knapsack problems, we keep C_{l*} as the best feasible set (line 10-11). Otherwise, we split the best set into two subsets $\mathcal{F}_1$ and $\mathcal{F}_2$ (line 13-14), where C_{l*} is assigned to $\mathcal{F}_1$ while the remaining items to $\mathcal{F}_2$. In this case, $\mathcal{F}_1$ has a total profit less than the half of profit of the best solution whereas $\mathcal{F}_2$ shows a total profit greater than the half of the best solution (recall that the sum of the profits of both subsets gives exactly the profit of the best solution). If the total space covered by the items within $\mathcal{F}_1$ is greater than the half of the total system capacity area (line 15), the second subset $\mathcal{F}_2$ will consequently cover less then the half of the available system capacity [10]. Therefore, the subset $\mathcal{F}_2 = B_{l*}$ could be easily (in polynomial time) packed into the system capacity (line 16). Otherwise, if the space of $\mathcal{F}_1$ is not enough to cover the half of the total system capacity Θ, we move the item with the greatest profit and the highest traffic class k (more flexible) from $\mathcal{F}_2$ to $\mathcal{F}_1$ one by one until the space of $\mathcal{F}_1$ is greater than the half of the system capacity (lines 19-22). Then, if the total profit of $\mathcal{F}_2$ is greater than the half of the best one (line 23), the algorithm ends and we keep $\mathcal{F}_2$ as the optimal set. Otherwise, we choose the set providing the best total profit after comparing $\mathcal{F}_2$ without the latest added element, against $\mathcal{F}_1$.

Theorem 2. *The Algorithm 1 provides a performance ratio of at most* $\frac{5}{2} + \epsilon$.

Sketch of Proof: The binary knapsack problem BIN-KP admits an FPTAS, as shown in [28]. For any small $\epsilon > 0$, there exists an algorithm that provides in polynomial time the best solution $C_{l*} \cup B_{l*}$ among $\binom{|\Sigma|}{2}$ knapsack problems with a total profit of $v_* = v(C_{l*} \cup B_{l*})$, where $v_* = \frac{OPT}{1+\epsilon/2}$ and OPT is the profit of the optimal solution. When we split the best set of items into two subsets $\mathcal{F}_1$ and $\mathcal{F}_2$ while moving items one by one from $\mathcal{F}_2$ to $\mathcal{F}_1$ in order to reach half of the total system capacity Θ, the first subset $\mathcal{F}_1$ contains more than 2 items, i.e., $|\mathcal{F}_1| \geq |C_l| + 1$, where $|C_l| = 2$ by definition. Considering the last added item u in $\mathcal{F}_1$, its profit is $v(\sigma_u) \leq \frac{v(\mathcal{F}_1)}{3}$. If $v(\mathcal{F}_2) < \frac{2v_*}{5}$ then $v(\mathcal{F}_1) > \frac{3v_*}{5}$. Therefore, $v(\mathcal{F}_1 \setminus u) \geq \frac{2v(\mathcal{F}_1)}{3} > \frac{2v_*}{5}$. In this case the performance ratio is at most $5/2 + \epsilon$ (line 27 of Algorithm 1). In all other cases (line 11, line 16 and line 24), the solution provides a profit of at least $\frac{v_*}{2}$ that results in a performance ratio of at most $2 + \epsilon$. □

The first 5 rows of our algorithm are solved within $O(n^2)$ computational time, revealing the number of knapsack problems BIN-KP to be solved. Given that the knapsack problem solution is achieved within a $O(n \log n)$, as the solution is optimal with a moderate number of items, the complexity of the Network Slices Packer is dominated by $O(n^3 \log n)$.

Forecasting-aware Network Slicer. When the forecasted information is available, i.e., network slice requests accurately reshaped, the admission control might fit more network slice requests while still guaranteeing the committed traffic SLAs. To this aim, we need to propose a new algorithm that reflects the concept of simulated annealing [29]. The additional complexity is due to the feasibility check of a given set of items into the system capacity: packing items in a different order might influence the solution optimality in the next attempts.

[9] With abuse of notation, we use throughout the paper $v(\mathcal{U})$ to identify the overall profit of all elements in $\mathcal{U}$, i.e., $v(\mathcal{U}) = \sum_{u \in \mathcal{U}} v(\sigma_u)$.

[10] Based on the Steinberg's theorem, if the sum of the item areas are less than the half of the box, they can be packed. See A. Steinberg, "A strip-packing algorithm with absolute performance bound 2", *SIAM Journal on Computing*.

The details of the algorithm are described in the following. We adopt a coding scheme called *sequence pair* [30] to represent candidate solutions of Problem ADM-CONTROL. The solution is represented by a pair of permutations of $|\mathcal{I}|$ items $\{\pi_+, \pi_-\}$. The first π_+ permutation indicates the spatial relation between items on the horizontal axis. In other words, the order of the items included in π_+ unveils how they are spatially allocated into the system capacity, e.g., if i is specified before j in π_+, it should be allocated on the left side of j. Similarly, π_- provides guidelines on how to vertically accommodate items into the system capacity.

The algorithm iteratively works on such permutations. The simulated annealing scheme could easily change the permutations by checking at every step kk whether the new locations of selected items into the system capacity are i) feasible (i.e., within the system capacity bounds and not overlapping each others) and ii) whether they provide a greater objective function value, i.e., $\Delta F = F_{kk+1}(x) - F_{kk}(x) > 0$. The objective function $F_{kk}(x)$ at step kk is defined as the overall profit of accommodated items into the system capacity based on the sequence pair $\{\pi_+, \pi_-\}$.

However, solutions with lower objectives might also be accepted according to an admission probability $Pr_a(\Delta F) = \frac{\Delta F}{Tr}$, where Tr is defined as a temperature value obtained by the logarithmic cooling function $Tr_k = \frac{Tr_0}{ln(1+kk)}$ whereas Tr_0 is the initial temperature. The temperature implicitly defines the scope of our search while looking for a better solution. At the beginning of the algorithm execution, the temperature value is high resulting in a broader area to find a better solution. As the number of steps grows, the temperature decreases and our algorithm focuses on more selective areas. This mechanism prevents our scheme from selecting sub-optimal value thereby choosing the global optimum.

The Forecasting-aware Network Slicer algorithm starts by sorting in non-increasing order the slice requests according to their profits (c_i) and traffic class (k). At each step kk, the algorithm decides to shuffle permutations π_+, π_-, add a new item into both sets or remove an existing one. The algorithm stops when the temperature Tr reaches a zero-value without finding better solutions in the next steps. While this algorithm asymptotically finds the global optimal solution, the running time might be not affordable. In Section VII, we provide an empirical complexity analysis with suggestions to improve it.

VI. Scheduling network slice traffic

Once network slices have been admitted, a proper scheduling is needed to meet the slice SLAs. In the following we present a novel network slice scheduler running on each base station that pursues the following two goals: i) serving the tenant traffic of the granted network slices, and ii) providing the required feedback to the forecasting process, yielding a closed-loop solution that drives the system to optimal performance (see Fig. 1).

A. Multi-class slice scheduler

We start by designing a scheduling algorithm that accounts for the different slice SLAs. We denote a traffic request from tenant i under base station b for traffic class k by $r^{(k)}_{i,b,z}$. We consider 6 traffic classes as described in Table I. Each traffic class is characterized by a time window $T^{(k)}$ identifying the time duration $[z; z+T^{(k)}]$ for which a class should see the rate guaranteed, where $z \in \mathbb{Z}$ denotes the individual time intervals. This is shorter for high-demanding traffic requirements, larger for best-effort class. The scheduler ensures that the amount of committed resources is served within the given time window.

The key objective of our novel network slice traffic scheduler is to minimize consumed resources while guaranteeing the traffic SLAs within a network slice. When forecasted traffic and tenant distribution information is available, the scheduler expects slice traffic levels below the predicted traffic bounds i.e., $r^{(k)}_{i,b,z} \leq \hat{R}^{(k)}_{i,b,z}, \forall z \in L_i$. If forecasted traffic bounds are under-estimated and the traffic demands exceed the expected values, the allocated resources may grow to a value as large as the value agreed during the slice request admission, i.e., $R^{(k)}_i = R^{(k)}_{i,b}, \forall b \in \mathcal{B}$. In this case, slice allocations may overlap and traffic class requirements might not be satisfied incurring in slice SLA violations.

We formulate the scheduler problem as a general minimization problem addressing all traffic class SLAs within a time window T_{SCHED} much larger than any time window $T^{(k)}$, i.e., $T_{\text{SCHED}} \gg T^{(k)}, \forall k$. We let $s^{(k)}_{i,b,j}$ denote the scheduled traffic corresponding to the resources served at time j under base station b for each network slice, within the set of admitted slices $x^{b,(k)}_i$ available from the admission phase. The problem is formulated as follows

Problem SLICER-SCHEDULING [11]:

$$\text{minimize} \quad \sum_{j \in T_{\text{SCHED}}} \left(s^{(k)}_{i,j} + \zeta \, P^{(k)}_{i,j} \right)$$

$$\text{subject to} \quad \left(\sum_{j=zk+\bar{t}}^{zk+\bar{t}+T^{(k)}} s^{(k)}_{i,j} \right) \geq r^{(k)}_{i,z} x^{(k)}_i, \; \forall z \in \left[0, \left\lceil \tfrac{L_i}{T^{(k)}} \right\rceil - 1\right];$$

$$\sum_{i \in \mathcal{N}} s^{(k)}_{i,j} \leq \Theta + P^{(k)}_{i,j}, \quad \forall j \in \mathcal{L};$$

$$s^{(k)}_{i,j} \in \mathbb{R}_+, \qquad \forall i \in \mathcal{N}, j \in \mathcal{L}, k \in \mathcal{K};$$

where Θ is the capacity of the cell, given by the total amount of resource blocks, whereas $P^{(k)}_{i,j}$ is the penalty incurred for not having satisfied a particular tenant slice traffic SLA, i.e., a SLA violation. ζ is constant factor set to a large value so as to guarantee that decreasing the penalty $P^{(k)}_{i,j}$ has always the highest priority.

We have design a simple heuristic to solve **Problem** SLICER-SCHEDULING based on the earliest deadline first (EDF) policy, as shown in Algorithm 2. The algorithm is run every time slot j. We select only slice traffic requests $r^{(k)}_i$ of admitted network slice requests σ_i, i.e., where $x_i = 1$ (line 3). We calculate a utility function μ_i for each network slice request by giving more priority to those close to the deadline $T^{(k)}_i$ (line 5). Thus, we start scheduling slice traffic requests within the available system capacity Θ (line 16) based on such utility. We keep assigning resources to network slices until the system capacity is saturated (line 12). Slice traffic requests $r^{(k)}_i$ not fully satisfied are kept for the next time slot $j+1$. If they are not served within the deadline $T^{(k)}_i$, a penalty value $P^{(k)}_{i,j}$ is set up (line 7).

[11] To reduce clutter, hereafter we drop the cell subscript b as the same problem formulation can be independently applied to multiple cells.

Algorithm 2 Multi-class Slice Scheduler: Algorithm to schedule the network slice traffic at time j without violating traffic SLAs within the time window T_{SCHED}.

Input: $\mathcal{X} = \{x_i^{(k)}\}, \Sigma = \{\sigma_i^{(k)}\}, r_{i,j}^{(k)}, \Theta$
Initialization: $C = 0, s_{i,j}^{(k)} = 0, P_{i,j}^{(k)} = 0; \forall i \in \mathcal{I}$
Procedure
$r_i^{(k)} = r_i^{(k)} + r_{i,j}^{(k)}$
for all $\sigma_i(r_{i,z}^{(k)}, T_i^{(k)})$ **do**
  if $x_i^{(k)} == 1$ **then**
    if $(T_i^{(k)} - j) \geq 0$ **then**
      $\mathcal{A} \leftarrow \mu_i = \frac{\left(r_{i,z}^{(k)} - \sum_{t=0}^{j} s_{i,t}^{(k)}\right)}{(T_i^{(k)} - j)\Theta}$
    else
      $P_{i,j}^{(k)} = r_i^{(k)} - \sum_{j=0}^{t} s_{i,j}^{(k)}$
      $r_i^{(k)} = 0$
    end if
  end if
end for
$C = \Theta$
while $C > 0$ **do**
  Sort $\mathcal{A}$ in non-increasing order of utility μ_i
  $\mathcal{H} \leftarrow \mathcal{A}$, **Pull** only element(s) with highest utility μ_i
  $s_{i,j}^{(k)} = \min\{r_i^{(k)}, \lceil \frac{C}{|\mathcal{H}|} \rceil\}, \forall i \in \mathcal{H}$
  $C = C - \sum_{i \in \mathcal{I}} s_{i,j}^{(k)}$
  $r_i^{(k)} = r_i^{(k)} - s_{i,j}^{(k)}, \forall i \in \mathcal{I}$
end while
Output: $s_{i,j}^{(k)}, P_{i,j}^{(k)}, r_i^{(k)}; \forall i \in \mathcal{I}$

The network slice scheduler keeps track of SLA violations by monitoring the penalty values $P_{i,j}^{(k)}$ to promptly trigger dynamic forecasting parameters adjustments (as explained in the next sub-section).

B. Online Reinforcement Learning

Forecasting process failures may lead admission control to overbook available network resources, yielding SLA violations. A monitoring procedure is designed to keep track of the number of such violations and provide feedbacks to the forecasting phase about the penalty value $P_{i,j}^{(k)}$ in Problem SLICER-SCHEDULING. From Eq. (4), we can derive the forecasting error probability for a generic traffic class k as follows

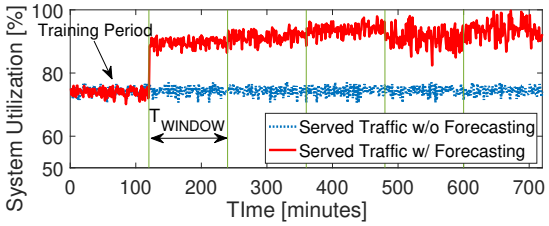
$$\chi_i^{(k)} \; : \; h_i^{(k)} \Omega_\chi \sqrt{Var(e_{i,z}^{(k)})} = \hat{d}_i^{(k)}, \tag{12}$$

where $h_i^{(k)}$ is the penalty history function defined as [12]

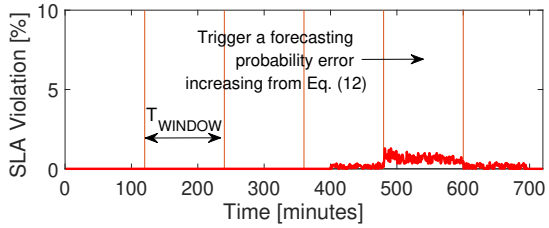
$$h_i^{(k)} = e^{\frac{nm}{W_S + nm}}, \tag{13}$$

with nm defined as the number of times the penalty is null, $P_{i,j}^{(k)} = 0, \forall j$, and W_S as the length of the season considered in the forecasting process in Section IV. The penalty history function represents the control policy and drives the system from a setting where a higher forecasting error probability may be experienced to a more conservative setting, where no SLA violation occurs. This is efficiently done by our algorithm: in case of forecasting failures, a larger forecasting error probability ($\chi_i^{(k)}$) is derived pushing the system towards a more conservative setting, with smaller gains.

[12]Note that different penalty history functions might be applied in order to properly model the feedback loop reactiveness without affecting the complexity of the proposed framework.



(a) System utilization



(b) SLA Violations

Fig. 6: System performance comparison with and without forecasting preprocessing.

VII. Performance Evaluation

We carried out an exhaustive simulation campaign to validate our RL-NSB framework. Our system is evaluated through an ad-hoc simulator developed in MATLAB®with a dual Intel(R) Xeon CPU 2.40GHz 4-cores and 16GB RAM. A summary of the simulation parameters used is provided in Table II. The system includes $|\mathcal{B}| = 7$ base stations ([31]) and $|\mathcal{I}| = 10$ tenants ([3]). The average number of users associated with a tenant is $E[|\mathcal{U}_i|] = 100$, which are distributed uniformly. When they move, a SLAW model is applied [24]. Each tenant slice request may require an amount of resources ranging from 5% to 25% of the total system capacity, while their duration ranges between 1000 and 3600 seconds. $P_{i,k}$ defines the probability that a slice request reaches the network within a time window. At the beginning of each T_{FUTURE} the admission control procedure is invoked. Based on the forecasting information, network slice requests are granted for the next time window and the associated slice traffic is served.

A. System Utilization and SLA violations

A dynamic analysis of our system is provided here. Since no other work in the literature has proposed a solution for addressing network slice request accommodation, we benchmark our proposal against a legacy solution wherein no forecasted information is available during the admission control phase. The results are shown in Fig. 6(a) for a long simulation period of 720 minutes in terms of system utilization $Ul = \Theta - \sum_{i,k} s_{i,j}^{(k)}, \forall j \in T_{\text{FUTURE}}$, based on Problem SLICER-SCHEDULING. After a prior training period, the forecasting process provides useful information to the

TABLE II: System parameters ([31])

Parameters	Values	Parameters	Values
$\|\mathcal{I}\|$	10	$\|\mathcal{B}\|$	7 (21 sectors)
$\|\mathcal{K}\|$	6	$E[\|\mathcal{U}_i\|]$	100
Θ	200 RBs	ISD	250m
T_{OBS}	3600 s	η	0
T_{FUTURE}	7200 s	$P_{i,k}$	$\frac{1}{\|\mathcal{I}\|\|\mathcal{K}\|}$
L_i	$\{1000; 3600\}$ s	R_i	$\{\Theta * 0.05; \Theta * 0.25\}$
α_{SLAW}	3	Av. Speed	1.5m/s
ψ_{SLAW}	2.5	α_{DISTR}	1.5

admission control block. Based on such information, network slice requests are properly reshaped and more traffic requests are efficiently accommodated into the network capacity. The gain after the second time window is about 20%. While no SLA violation occurs, the forecasting process moves from a conservative behaviour to a more aggressive by reducing the safe margin, i.e., the forecasting error probability from Eq. (12), which visibly brings more gain in terms of system utilization. However, due to the randomness of the traffic requests issued to the system, forecasted information might underestimate the real traffic level resulting in a SLA violation, as shown in Fig. 6(b). This triggers the penalty history function $h_i^{(k)}$ which increases the forecasting error probability in the next time window, thereby keeping the SLA violation under control. Interestingly, our solution boosts the system utilization up to 100% while incurring a small SLA violation per tenant request (about 1.8% in a very short period).

B. Training and Prediction performance

In [12, Fig. 6], we deeply evaluated the effectiveness of the forecasted information as the relative gain $G_F = \left(\frac{\bar{U}l_F}{\bar{U}l_L} - 1\right)\%$, where $\bar{U}l_F$ and $\bar{U}l_L$ are the average utilization value of the forecasting and legacy solution, respectively.

Hereafter, we evaluate the impact of the training period on the forecasting process and, in turn, on the overall system performance in terms of efficiency and revenues.

On the one hand, we show the relative gain G_F when different observation window sizes are considered in Fig. 7. As explained in Section IV, the observation window plays a key-role as it unveils the trend, level and season features of the (traffic) time-series for any admitted slice. The larger the observation window, the more available past information, the higher the accuracy of the forecasting process. However, a very long observation period may prevent the system from accurately following traffic variations, especially when dealing with periodic traffic (within fixed time-seasons). This behaviour is clearly showed in Fig. 7, where both curves tend to saturation after certain observation window-values. Note that, when the observation window is set to $T_{\text{OBS}} = 0$, there is no available information for the forecasting process leading to a relative gain equal to 0. In addition, a longer observation window implies higher complexity and requires more memory to store information, as depicted on the right y-axis in Fig. 7.

On the other hand, the future time window T_{FUTURE} exhibits its relevance in Fig. 8 as it directly impacts on the decision window in the slice admission control. While a large future time window may result in inaccurate forecasted values (as explained in Section IV), a short one may require the admission control to run very frequently resulting in higher time complexity. Therefore, both terms must be properly adjusted, as explained in [32], based on the tenant traffic features, user mobility properties and network deployments.

C. User mobility

When users of different tenants are geographically spread on different areas, the proposed solution can learn and leverage on such information to improve the efficiency of admission control mechanism. Indeed, if different slices use network

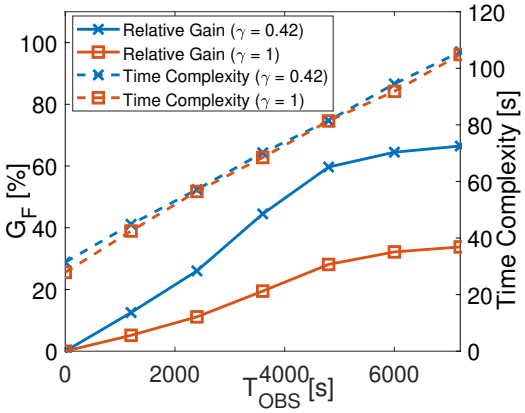


Fig. 7: Evaluation of different observation window-values T_{OBS} when the number of tenants is $I = 10$ and $T_{\text{FUTURE}} = 7200$s.

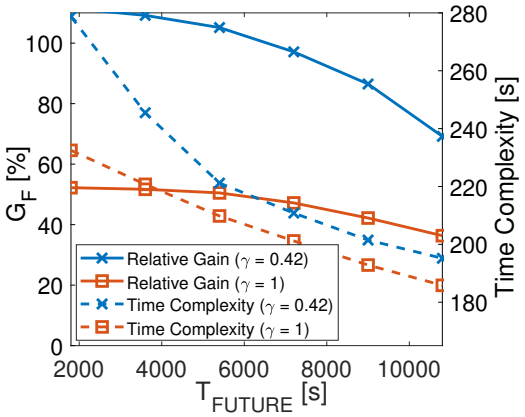


Fig. 8: Evaluation of different future window-values T_{FUTURE} when the number of tenants is $I = 20$ and $T_{\text{OBS}} = 3600$s.

resources of different cells, they can safely be admitted without risking saturation in any of these cells. The improvement on the overall efficiency of the network translates, in turn, into more admitted network slices and increased revenues.

Due to the nature of vertical segments (for e.g. automotive tenant that only gets users within specific geographical areas) we assume that each tenant is modelled with a different set of clusters and different structure of way-points within such clusters, as previously shown in Fig. 2. Please note that the spread of tenant users over the cellular network significantly affects the overall system performance. The smaller the number of selectable clusters, the more the accuracy of our tenant distribution prediction module. This in turn translates into a higher multiplexing gain as shown in Section V. We model the tenant spread with a spread factor γ.

We first improve the synthetic scenario shown in [12, Fig. 6] by limiting the spread of the tenant users to $\gamma = 42\%$. In other words, each tenant users are distributed a-priori only on the 42% of the cells in the network. Our solution can iteratively learn such information and allocate the network slice along those affected cells c. Obtained results are shown in Fig. 9. Considering the maximum system capacity $\Theta = 200RBs$, the relative gain is above 100% when the number of tenants asking for network slices is large.

We also evaluate the relative gain while varying the spread

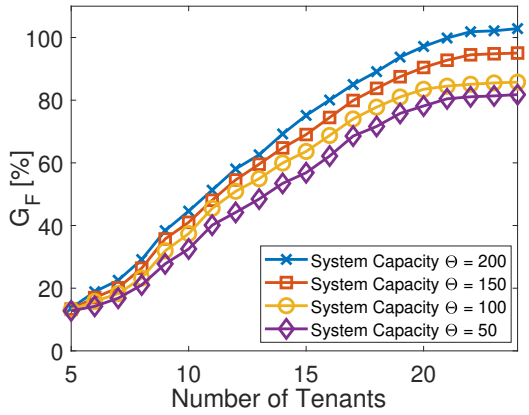


Fig. 9: System utilization gain for different network scenarios when $\gamma = 42\%$.

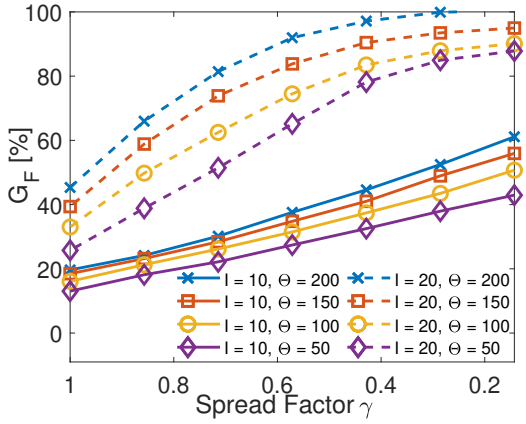


Fig. 10: System utilization gain for different heterogeneity levels γ.

factor γ for different system capacity values and two scenarios with 10 and 20 tenants, as depicted in Fig. 10. Notably, when few tenants are requesting network slices ($I = 10$), the relative gain curve exhibits an increasing trend as the tenant users are located within smaller areas. The reason is due to the spatial multiplexing gain obtained for those tenants using only limited portions of the network: our mechanism can learn and predict the user distributions and efficiently reduce the spatial extension of the network slice admitted into the system. When the number of tenants is large ($I = 20$), due to the high variability of the network slice requests, the system can better select those maximizing the system efficiency and reach even higher values of relative gains G_F. However, we note that after certain spread factor values, the curve shows a saturation behavior. All in all, our simulation campaign proves that specific tenants behaviors—when accurately captured—can significantly benefit the admission control process and the system efficiency maximization.

D. Algorithm Complexity

Finally, we provide an empirically study of the computational cost for the two admission control algorithms proposed. For a fair comparison, we apply the algorithm to the same instances of the problem and average the results over several instances (100). Note that in this study only regular network slice shapes are considered.

In Table III, we show the results for different tenants present in the system. The results are expressed in terms of number of slices admitted into the system capacity (on the left part of the column) and time elapsed for getting a response to an admission request (on the right part of the column). The average number of network slice requests within a single instance of the problem ranges from 30 (with 10 tenants) to 90 (with 30 tenants). Interestingly, the Forecasting-aware Network Slicer algorithm outperforms the Network Slices Packer, but it also experiences a longer computational time. We impose a time limit $T_Z = 600$s to avoid the process starvation. Given the long-term execution (every 30 minutes) of admission control algorithms, this time bound is still acceptable for the overall system implementation. Results for that fixed time window are shown in brackets while optimal ones when the algorithm successfully ends are highlighted in bold text. Indeed, the Forecasting-aware Network Slicer algorithm shows reasonable results in an affordable computing time.

TABLE III: Empirical complexity analysis

No. of tenants	10		20		30	
No. of slice req.	30 - 60		60 - 120		90 - 180	
Algorithms	**slices [no.]**	**time [sec]**	**slices [no.]**	**time [sec]**	**slices [no.]**	**time [sec]**
Network Slices Packer	9.584	78.1	11.337	215.7	13.226	497
Forecasting-aw. Network Slicer	9.607	129.5	**13.061** (11.897)	**714** (600)	**15.81** (13.307)	**3514.4** (600)

Conversely, when irregular shape patterns are considered, the time complexity of the Forecasting-aware Network slicer further increases. This may become a significant drawback when the number of network slice requests is greater then 50. We overcome this problem in the following way. We first apply the network slices packer algorithm for the case with regular shapes. This provides an initial state for the Forecasting-aware Network slicer, which then starts exploring the neighbouring solutions to check whether they fit into the system capacity. In this way, we are able to reduce the computational time of the Forecasting-aware Network slicer by 20%, making it suitable for realistic deployments.

VIII. Conclusions

In this paper, we have presented our proposed network slicing traffic framework for a Reinforcement Learning-based 5G Network Slice Broker (RL-NSB) building on i) traffic forecasting, ii) slice admission control and iii) slice traffic scheduling. The *forecasting* solution builds on Holt-Winters theory to predict future traffic levels per network slice and analyzes users' behavior to infer future mobility patterns. Predictions are used to improve the admission control decision process with the goal of maximizing the overall system utilization. The *admission control* solution maps the problem of admitting slice requests onto a geometric knapsack problem. As this problem is NP-hard, we have proposed two computationally-tractable algorithms that address, respectively, regular and irregular network slice requests. The *network slice scheduling* solution keeps track of SLA violations for the different slices and feeds this information back to the forecasting engine, which adapts its behaviour to correct the deviations observed.

Our main findings can be summarized as follows: i) Holt-Winters theory is effective to forecast network slicing traffic,

ii) elastic network slice requests might increase the maximum achievable system utilization due to the loose service guarantees, iii) the information on users spatial distribution can be used to drive admission control decisions and thus increase the network efficiency, iv) the benefits resulting from a proper forecasting increase as the number of network slice requests and system capacity grows and v) very significant system utilization gains can be achieved while keeping very low SLA violation risk levels.

REFERENCES

[1] GSMA, "Smart 5G Networks: enabled by network slicing and tailored to customers' needs," https://www.gsma.com/futurenetworks/wp-content/uploads/2017/09/5G-Network-Slicing-Report.pdf, September 2017.

[2] Third Generation Partnership Project (3GPP), "System Architecture for the 5G System," 3GPP TS 23.501, v. 16.0.2, Apr. 2019.

[3] NGMN Alliance, "NGMN 5G White Paper," Tech. Rep. Version 1, Feb. 2015.

[4] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From Network Sharing to Multi-tenancy: The 5G Network Slice Broker," *IEEE Communications Magazine*, vol. 54, no. 7, Jul. 2016.

[5] Third Generation Partnership Project (3GPP), "Study on Radio Access Network (RAN) Sharing enhancements," 3GPP TR 22.852, v. 13.1.0, Sep. 2014.

[6] A. R. Raikwar, R. R. Sadawarte, R. G. More, R. S. Gunjal, P. N. Mahalle, and P. N. Railkar, "Long-term and short-term traffic forecasting using holt-winters method: A comparability approach with comparable data in multiple seasons," *Int. J. Synth. Emot.*, vol. 8, no. 2, pp. 38–50, 2017.

[7] O. Narmanlioglu, E. Zeydan, M. Kandemir, and T. Kranda, "Prediction of active UE number with bayesian neural networks for self-organizing lte networks," in *2017 8th International Conference on the Network of the Future (NOF)*, Nov. 2017, pp. 73–78.

[8] R. Mahindra, M. A. Khojastepour, H. Zhang, and S. Rangarajan, "Radio access network sharing in cellular networks," in *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, Oct. 2013.

[9] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan, "NVS: A substrate for virtualizing wireless resources in cellular networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1333–1346, Oct. 2012.

[10] X. Costa-Perez, J. Swetina, T. Guo, R. Mahindra, and S. Rangarajan, "Radio access network virtualization for future mobile carrier networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 27–35, Jul. 2013.

[11] J. He and W. Song, "AppRAN: Application-oriented radio access network sharing in mobile networks," in *IEEE International Conference on Communications (ICC)*, June 2015, pp. 3788–3794.

[12] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile traffic forecasting for maximizing 5G network slicing resource utilization," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM'17)*.

[13] J. S. Panchal, R. D. Yates, and M. M. Buddhikot, "Mobile network resource sharing options: Performance comparisons," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4470–4482, Sep. 2013.

[14] G. Tseliou, K. Samdanis, F. Adelantado, X. Costa, and C. Verikoukis, "A capacity broker architecture and framework for multi-tenant support in LTE-A networks," in *Proceedings of IEEE International Conference on Communications (ICC)*, May 2016.

[15] P. Caballero, A. Banchs, G. de Veciana, and X. Costa-Pérez, "Multi-tenant radio access network slicing: Statistical multiplexing of spatial loads," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 3044–3058, Oct. 2017.

[16] Y. L. Lee, J. Loo, T. C. Chuah, and L. C. Wang, "Dynamic network slicing for multitenant heterogeneous cloud radio access networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2146–2161, Apr. 2018.

[17] B. Han, V. Sciancalepore, D. Feng, X. Costa-Perez, and H. D. Schotten, "A utility-driven multi-queue admission control solution for network slicing," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM'19)*.

[18] Third Generation Partnership Project (3GPP), "Telecommunication management; Network Sharing; Concepts and requirements," 3GPP TS 32.130, v. 15.0.0, Jun. 2018.

[19] J. Salvat, L. Zanzi, A. Garcia-Saavedra, V. Sciancalepore, and X. Costa-Perez, "Overbooking network slices through yield-driven end-to-end orchestration," in *Proceedings of International Conference on emerging Networking EXperiments and Technologies (CONEXT'18)*.

[20] G. R. Ash, *Traffic Engineering and QoS Optimization of Integrated Voice & Data Networks*. Morgan Kaufmann Publishers Inc., 2007.

[21] B. Błaszczyszyn, M. Jovanovicy, and M. K. Karray, "How user throughput depends on the traffic demand in large cellular networks," in *in International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2014, pp. 611–619.

[22] A. B. Koehler, R. D. Snyder, and J. Ord, "Forecasting models and prediction intervals for the multiplicative holt–winters method," *International Journal of Forecasting*, vol. 17, no. 2, pp. 269–286, Jun. 2001.

[23] Third Generation Partnership Project (3GPP), "Technical Specification Group Services and System Aspects; Policy and charging control architecture," 3GPP TS 23.203, v. 16.0.0, Mar. 2019.

[24] K. Lee, S. Hong, S. J. Kim, I. Rhee, and S. Chong, "SLAW: Self-similar least-action human walk," *IEEE/ACM Transactions on Networking*, vol. 20, no. 2, pp. 515–529, Apr. 2012.

[25] T. Hofmann, "Unsupervised learning by probabilistic latent semantic analysis," *Mach. Learn.*, vol. 42, pp. 177–196, Jan. 2001.

[26] C. Biernacki, G. Celeux, and G. Govaert, "Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate gaussian mixture models," *Comput. Stat. Data Anal.*, vol. 41, pp. 561–575, Jan. 2003.

[27] K. Jansen and G. Zhang, "Maximizing the total profit of rectangles packed into a rectangle," *Algorithmica*, pp. 323–342, Mar. 2007.

[28] H. Kellerer and U. Pferschy, "A new fully polynomial time approximation scheme for the knapsack problem," *Journal of Combinatorial Optimization*, vol. 3, no. 1, pp. 59–71, Jul. 1999.

[29] A. Liu, J. Wang, G. Han, S. Wang, and J. Wen, "Improved simulated annealing algorithm solving for 0/1 knapsack problem," in *Proceedings of the 6th International Conference on Intelligent Systems Design and Applications*, Oct. 2006, pp. 1159–1164.

[30] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1518–1524, Dec. 1996.

[31] ITU-R, "Guidelines for evaluation of radio interface technologies for IMT-Advanced," Report ITU-R M.2135-1, Dec. 2009.

[32] S. Gelper, R. Fried, and C. Croux, "Robust forecasting with exponential and holt–winters smoothing," *Journal of Forecasting*, vol. 29, no. 3, pp. 285–300.

[33] E. M. Arkin, S. Khuller, and J. S. B. Mitchell, "Geometric knapsack problems," *Algorithmica*, vol. 10, no. 5, pp. 399–427, Nov. 1993.

Vincenzo Sciancalepore (S'11–M'15) is a Senior Researcher and RAN specialist at NEC Laboratories Europe, Germany. He is currently focusing his activity in the area of network virtualization and network slicing challenges. He is currently involved in the IEEE Emerging Technologies Committee leading the initiatives on SDN and NFV. He was also the recipient of the national award for the best Ph.D. thesis in the area of communication technologies in 2015.

Xavier Costa-Perez (M'06–SM'18) is Head of 5G Networks R&D and Deputy General Manager of the Security & Networking Research Division at NEC Laboratories Europe. His team contributes to products roadmap evolution as well as to European Commission R&D collaborative projects and received several awards for successful technology transfers. He received both his M.Sc. and Ph.D. degrees in Telecommunications from the Polytechnic University of Catalonia (UPC) in Barcelona and was the recipient of a national award for his Ph.D. thesis.

Albert Banchs (M'04–SM'12) received his M.Sc. and Ph.D. degrees from the Polytechnic University of Catalonia (UPC-BarcelonaTech) in 1997 and 2002, respectively. He is currently a Full Professor with the University Carlos III of Madrid (UC3M), and has a double affiliation as Deputy Director of the IMDEA Networks institute. Before joining UC3M, he was at ICSI Berkeley in 1997, at Telefonica I+D in 1998, and at NEC Europe Ltd. from 1998 to 2003. His research interests include the performance evaluation and algorithm design in wireless and wired networks.

Supplementary downloadable material

APPENDIX

Lemma 1. *Let each cell b independently assigned with a fixed amount of resources Θ^b within a given time windows T^b. The admission control problem of the overall network aiming at accommodating the network slice requests while maximizing the network resource utilization, can be obtained as the combination of independent instances of an admission control problem executed for each cell b.*

Proof: We reasonably assume that cell coverages do not overlap so as each user is connected to a single cell. Therefore, considering all cells deployed and users placed in our network, we can generalize the admission control problem of the entire network deployment as the following

Problem GENERAL−ADM−CONTROL:

$$\begin{aligned} \text{maximize} \quad & \sum_{i\in\mathcal{I}} c_i x_i \\ \text{subject to} \quad & \sum_{i\in\mathcal{I}} w_i \cdot x_i \leq W; \quad \forall b \in \mathcal{B} \\ & x_i \in \{0,1\}, \qquad \forall b \in \mathcal{B}, \forall i \in \mathcal{I}. \end{aligned}$$

By solving Problem GENERAL−ADM−CONTROL, an optimal set of tenants (x_i) admitted into our network is provided. Clearly, when the geographical scope of our network is reduced to a single cell b, the suggested solution provides only the set of tenants (x_i^b) admitted into that specific cell. This turns into multiple optimization problems that are independently solved to provide distinct solutions of admitted tenants. ■

Lemma 2. *Let the resource availability of cell b be a box with height Θ^b and width T^b, and let each item $i \in \mathcal{I}$ be the network slice request σ_i with height R_i and width L_i. Then, the admission control problem is mapped into a Geometric Two-dimensional knapsack problem with the objective of filling up the cell capacity with network slice requests while maximizing the base station resource utilization.*

Proof: We can prove the above lemma by recalling the formulation of the geometric knapsack problem [33]. Specifically, the classical geometric knapsack problem is expressed as follows

$$\begin{aligned} \text{maximize} \quad & \sum_{k=1}^{n} V_k N(X_k) \\ \text{subject to} \quad & \sum_{k=1}^{n} W_k N(X_k) \leq W_{\text{max}}; \\ & N(X_k) \in \mathcal{Z}^+ \\ & \mathcal{S}(X_{k_p}) \cap \mathcal{S}(X_{j_r}), \qquad \forall j_r \neq k_p; \\ & \mathcal{S}(X_{k_p}) \subset \mathcal{S}_{\text{total}}; \end{aligned}$$

where $\mathcal{S}(\cdot)$ denotes the space (in n-dimensions) of the k_pth item, $\mathcal{S}_{\text{total}}$ is the space bounding the knapsack and $N(X_k)$ represents the integer number of times each item is allocated. Assuming that each item can be placed only once, i.e., $N(X_k) \leq 1$ and considering $n = 2$ dimensions, i.e., time and capacity, Problem ADM−CONTROL can be easily mapped onto a Geometric Two-dimensional knapsack problem. This concludes the proof. ■

Lemma 3. *Let the resource availability of cell b be a box with height Θ^b and width T^b, and let each item $i \in \mathcal{I}$ be the network slice request σ_i with irregular shapes, identified by different height values $\hat{R}_{i,b,z}^{(k)}$ and width L_i. Then, the admission control problem is mapped into a Flexibile Geometric Two-dimensional knapsack problem, with the objective of maximizing the cell resources utilization whilst accommodating network slice requests.*

Proof: The proof of this lemma is straightforward. Basically, we assume that a Flexibile Geometric Two-dimensional knapsack problem is a Geometric Two-dimensional knapsack problem where items do not have fix dimensions but they can be considered as fluid. This allows to relax the space constraint ($\mathcal{S}(X_{k_p}) \cap \mathcal{S}(X_{j_r})$) of the geometric knapsack problem by letting items to dynamically change their shapes (while preserving the assigned area) to fit within the given knapsack bounds. ■

Lemma 4. *Considering all items with full flexible dimensions, we can identify one single weight w_i per item representing the area required. Then, if the utility value $c_i = w_i$ the decision problem DEC−ADM−CONTROL reduces to a "Subset Sum Problem".*

Proof: We recall the "Subset Sum Problem" as the following. Given a set of n items and a knapsack, with w_j as the weight of item j and c as the capacity of the knapsack, select a subset of the items whose total weight is closest to, without exceeding, c, i.e.,

$$\begin{aligned} \text{maximize} \quad & z = \sum_{k=1}^{n} w_k x_k \\ \text{subject to} \quad & \sum_{k=1}^{n} w_k x_k \leq c, \\ & x_k \in \{0,1\}, \qquad \forall k \in \mathcal{N} = \{1, \cdots, n\}. \end{aligned}$$

Let us consider Problem DEC−ADM−CONTROL as the following: given an arbitrary value V, n items with a value c_i and a given area a_i enclosed within a two-dimensional shape identified by $\hat{R}_{i,b,z}^{(k)}$ and L_i, and a box with capacity $\mathbb{S}^b$ delimited by Θ^b and T^b, is there a subset $\mathcal{I} \in \{1, 2, \cdots, n\}$ such that items do not overlap and $\sum_{i\in\mathcal{I}} c_i \geq V$?

Assuming $c_i = w_i$, Problem DEC−ADM−CONTROL can be mapped onto a "Subset Sum Problem". ■