An Artificial Intelligence Framework for Slice Deployment and Orchestration in 5G networks

Journal:	*IEEE Transactions on Cognitive Communications and Networking*
Manuscript ID	TCCN-TPS-19-0034.R1
Manuscript Type:	Transactions Paper Submissions
Date Submitted by the Author:	09-Jul-2019
Complete List of Authors:	Dandachi, Ghina; CEA LETI De Domenico, Antonio; CEA LETI Hoang, Dinh; University of Technology Sydney, School of Electrical and Data Engineering Niyato, Dusit; Nanyang Technological University,
Keyword:	Artificial intelligence, Protocols, Architecture, Mobile communication

An Artificial Intelligence Framework for Slice Deployment and Orchestration in 5G networks

Ghina Dandachi[1], Antonio De Domenico[1], Dinh Thai Hoang[2], and Dusit Niyato[3]
[1]CEA-LETI, Grenoble, France
[2]School of Electrical and Data Engineering, University of Technology Sydney, Australia
[3]School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore

Abstract—**Network slicing is a key enabler to successfully support 5G services with specific requirements and priorities. Due to the diversity of these services, slice deployment and orchestration are essential to guarantee service performance in a cost-effective way. In this paper, we propose an Artificial Intelligence framework for cross-slice admission and congestion control that simultaneously considers communication, computing, and storage resources with the aims of maximizing resources utilization and operator revenue. First, we propose a smart feature extraction solution to analyze the characteristics of incoming requests together with the already deployed slices, and then automatically evaluates the request requirements to make appropriate decisions. Second, we design an online algorithm that controls the slice admission based on their priorities, the arrival and departure characteristics, and the available resources. To mitigate system overloading, our framework dynamically adjusts resources allocated to low priority slices, thereby reducing the dropping probability of new slice requests. The proposed algorithm offers outstanding advantages over traditional static approaches by automatically adapting the controller decisions to the system changes. Simulation results show that our framework significantly improves the resource utilization and reduces the slice request dropping probabilities up to 44% as compared to the baseline schemes.**

I. Introduction

The technological revolution for the last decade has been changing the way we communicate by introducing emerging applications and services. Massive machine type communications (mMTC), autonomous vehicles, smart factories, and virtual reality are changing the network requirements in terms of number of connections per user, traffic volume, and end-to-end latency [1]. To efficiently support use cases and applications with heterogeneous requirements, the fifth generation (5G) communication systems will deploy a novel and flexible architecture where the network infrastructure is logically split into different instances, i.e., network slices, each designed for a specific service and running in the cloud environment [2]. A network slice is composed of physical and virtual network functions (NFs), where a virtual NF represents the software implementation of the traditional functions, such as routing or packet scheduling.

In this context, depending on the network load and service requirements, the 5G systems will need to manage network resources smartly according to different radio, transport, and cloud domains. Moreover, it is necessary to take into account that different slice requests may have diverse priorities and constraints, and the network resources have to be managed accordingly. Therefore, 3GPP has been developing network slice management and orchestration solutions based on the European Telecommunications Standards Institute (ETSI) Management and Orchestration and Network Function Virtualization frameworks [3]. In addition, the ETSI Experiential Network Intelligence (ENI) group has been investigating Artificial Intelligence (AI) to achieve autonomous, and thus cost-effective, slice management and orchestration in future communication networks [4]. However, these frameworks define only brief guidelines on architectural aspects (interfaces and requirements) and design principles without providing specific solutions. Accordingly, there is an urgent need for new schemes and frameworks able to provide cross-slice resource orchestration and management. In this paper, we focus on AI-based cross-slice admission and congestion control with the goal of maximizing the operator revenue by improving the network resource utilization.

A. Related Work

In the literature, the admission control problem for 5G network slicing has been investigated under two main directions: by using Markov Decision Process (MDP) and game theory. Specifically, Hoang et al. characterized the slice admission control problem as an MDP problem that can be solved by using the value iteration algorithm, which is based on dynamic programming [5]. However, this approach requires a minimum knowledge of service model characteristics (e.g., the slice request arrival and departure rates) and can only find the optimal solution for problems with a limited number of states. Bega et al. [6] introduced the concepts of elastic and inelastic slices, modeled the slice admission control as an MDP, and implemented the Q-learning algorithm to maximize the operator revenue. ~~However, the Q-learning algorithm suffers from the so-called "curse of dimensionality" [7], meaning that its computational requirements grow exponentially with the number of state variables in the problem. Accordingly, this approach cannot be efficiently implemented in realistic scenarios.~~

Caballero et al. [8] proposed a game theoretic approach for the slice admission control problem to enable fair resource partitioning between deployed slices. ~~However, this study only focuses on radio resources and does not consider priorities among different slice requests.~~ Jiang et al. [9] investigated the virtual resource allocation for network slicing by using auction mechanisms in order to maximize the network revenue under different slice priorities. ~~However, this study does not consider admission control and is unable to limit the slice request dropping probability.~~ Leconte et al. [10] proposed an elastic framework for cross-slice resource allocation that achieves a Pareto-efficient solution ensuring fair radio and computing resource allocation. ~~However, the obtained solution considers only instantaneous optimization and does not take into account the priorities between slices.~~ Delgado et al. [11] studied a greedy algorithm for slice resource allocation in wireless sensor networks. Four resource types are considered, i.e., radio, storage, computing and energy, with the energy resources being the most important one. ~~Nevertheless, the work in [11] did not consider priorities among different slices. In addition, game theory-based solutions cannot deal with system dynamics, and thus their outcome strategies are not able to provide long-term optimization. Moreover, none of the above works propose a solution for resource shortage problem in the context of slice deployment and orchestration.~~

Early works on AI for mobile networks highlighted the importance of introducing intelligence and automation in admission control function [12]. Chen et al. [13] studied admission control using fuzzy Q-learning to achieve a lower blockage rate in WCDMA/WLAN heterogeneous networks. Al-Maitah et al. [14] employed a genetic neurofuzzy controller to improve the quality of call admission in mobile networks. Although these methods proposed AI solutions, to the best of our knowledge, there is a lack of AI-based framework that can simultaneously address the admission and congestion control problems for 5G slice management and orchestration.

B. Contributions and Organization

The key contributions of this paper can be summarized as follows:

1) We propose a new architecture for joint cross-slice admission control and congestion control to maximize the operator revenue while considering different slice types and resource requirements. Accordingly, we introduce three new functions: Slice Analytics (SA), Admission Control (AC), and Congestion Control (CC). This novel architecture is fully compatible with the Network Slice Management layer proposed by 3GPP [3]. By adopting it, we jointly improve the usage of communication, computing, and cloud storage resources while guaranteeing the slice priorities.
2) The proposed SA function attempts to assign a new slice request to an existing Network Slice Instance (NSI) based on the common NFs to maximize the resource sharing across slices and accelerate the slice deployment process. In particular, we propose two different algorithms that can implement this functionality: Jaccard similarity-based assignment and spectral clustering-based assignment. The first scheme assigns each new slice request to the NSI that maximizes the Jaccard similarity. Alternatively, the spectral clustering approach tries to iteratively optimize the resource sharing by clustering all slices into new NSIs at each slice arrival. Our simulation results show that the proposed SA function can significantly reduce the resource footprint for each accepted slice.
3) The proposed cross-slice AC aims to optimize the trade-off between resource utilization and dropping probability of slice with high priorities. We then introduce the concept of cross-slice AC based on AI, where the controller is able to deal with system dynamics and its decision enables a long-term optimal solution in terms of the slice resource management. To achieve this goal, we design an AI scheme based on the State-Action-Reward-State-Action (SARSA) scheme [15] with linear function approximation (LFA), which is a model-free reinforcement learning approach that can deal with high dimensional and complex problems.
4) We also introduce a CC function that limits the dropping of slice requests, especially when the system becomes overloaded. The CC can scale down resources allocated to slices with low priorities, so that slices with high priorities can be admitted. This function is tightly coupled with the AC, as it determines the resources available for accepting new slice requests, while the amount of resources that can be scaled down is the results of the AC action. Therefore, we develop a joint implementation of both controllers using SARSA with LFA, which we denote as cross-slice admission and congestion control (CSACC).

The rest of the paper is organized as follows. The proposed architecture and the associated system model is presented in Section II. In Section III, we define the SA function and the proposed algorithms. In Section IV, we introduce the mathematical model of the designed CSACC, and then we present the SARSA algorithm with LFA in Section V. Simulation results are provided in Section VI. Finally, the conclusions and future directions are discussed in Section VII.

II. System Model

We consider a system supporting network slicing to satisfy the diverse 5G service requirements ~~of the services related to the 5G ecosystem~~. In this network, a slice management framework (described in Fig. 1) is used to instantiate network slices and orchestrate the ~~network~~ resources across the accepted slices by exploiting the so called *resource elasticity* paradigm [16]. ~~To design an efficient, multi-service, multi-slice, and multi-tenant architecture, we take advantage of the so called *resource elasticity* paradigm.~~ In a nutshell, with resource elasticity, the network is able to ~~the proposed slice management framework~~ gracefully adapts ~~the network~~ its configuration to system changes ~~in an automatic manner~~ through AI, such that ~~at each point in time~~ the available resources match the service demand as closely and efficiently as possible. ~~Although elasticity in mobile networks has traditionally been exploited in the context of communications resources (e.g., when a base station gracefully downgrades the spectral efficiency of a given communication link), here~~ Here, we apply this concept to communication, computational, and storage resources jointly. More specifically, we focus on elastic slices that enable a

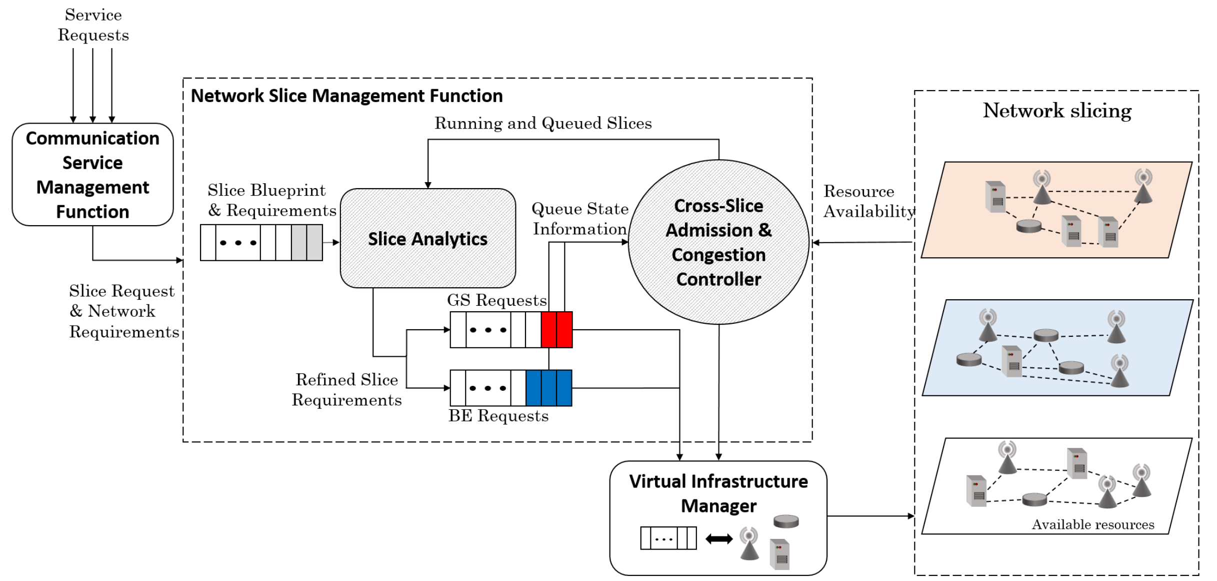

Service Requests
Communication Service Management Function
Slice Request & Network Requirements
Network Slice Management Function
Running and Queued Slices
Slice Blueprint & Requirements
Slice Analytics
Queue State Information
Cross-Slice Admission & Congestion Controller
GS Requests
Refined Slice Requirements
BE Requests
Resource Availability
Network slicing
Virtual Infrastructure Manager
Available resources

Figure 1: The proposed framework for network slice deployment and orchestration.

certain level of flexibility in the resource orchestration: we consider guaranteed quality-of-service (GS) slices that do not require fully dedicated network resources (i.e., resource isolation) and best effort (BE) slices that, in addition, allow the system to temporarily reduce their resources in order to deal with GS slices, which have higher priorities and tighter requirements.

In the 3GPP specification [3], the Communication Service Management Function (CSMF) receives requests for new services and transforms the consumer-facing service descriptions into slice-related network requirements such as connection density, end-to-end latency, and coverage. Then, the CSMF sends ~~the network slice requests together with the associated requirements~~ this information to the Network Slice Management Function (NSMF), which derives accordingly the so-called network slice blueprint [3]. The network slice blueprint is a description of the network slice in terms of required NFs, their interconnection and configuration ~~according to the specific service request~~. The main role of the NSMF is to manage and orchestrate the overall network slice life-cycle.

~~More specifically, in~~ In this work, we study how, based on the slice blueprint, the NSMF identifies the resource requirements for each slice and decides whether and when to instantiate the slice request. Then, the NSMF will distribute the available resources across the activated slices and try to maximize the resource utilization, i.e., the number of slices that can be handled at the same time. However, maximizing the resource utilization may prevent the system from accepting new slice requests with high priorities, which provide high revenue for the operator, due to the potential scarcity of available resources. Hence, this procedure needs to be carefully managed to optimize the trade-off between resource utilization and operator revenue.

To optimize such trade-off, we investigate two main functionalities represented by the grey blocks in Fig. 1:

- The SA ~~block~~ receives the slice blueprint and resource requirements related to the accepted and queued slice requests. It refines the amount of resources to be allotted to each network slice by analyzing the similarities in the slice requirements. Additionally, ~~this block~~ it classifies the slice requests into two types, GS and BE slices, according to the associated requirements[1].
- The CSACC ~~is a resource orchestrator that~~ performs jointly cross-slice AC and CC functions. ~~This block~~ It monitors the requests of the queued slices and the available resources, and manages the slice deployment and the resource allocation accordingly.

Finally, in the proposed framework, the role of the virtual infrastructure manager is to execute the instructions received from the CSACC.

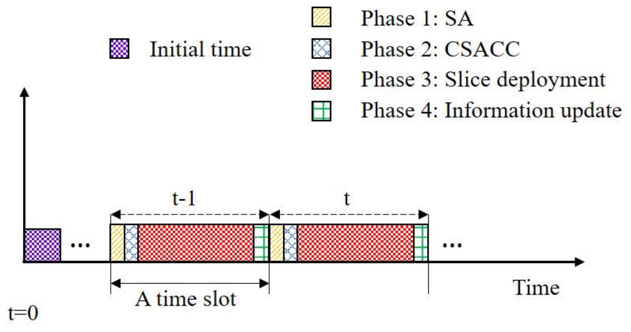


Figure 2: Decision making cycle for network slice deployment and orchestration.

According to the above system operations, we consider a decision-making cycle that can be divided into four phases

[1]This model can be straightforwardly extended to a more general classification that may include classical 5G use cases such as eMBB, URLLC, and mMTC.

as illustrated in Fig. 2. At the beginning of each time slot, the SA phase evaluates the effective resource requests based on the similarities between the queued network slices and the active network slices. Then, in the CSACC phase, based on the current states of the GS and BE queues, and the momentary available resources of the system, the CSACC first decides whether to scale down the resources allocated to the BE slices, and then selects the slice requests to admit. Then, in the third phase, i.e., the slice deployment phase, the virtual infrastructure manager allocates the network resources according to the CSACC decisions, and the accepted slices are instantiated. Finally, in the last phase, i.e., the information updating phase, new requests arrive, and they are processed by the CSMF, and the input of the SA is updated accordingly. The duration of the above discussed cycle should be short enough to monitor the variation in the slice resource requests and enable the NSMF functions to react accordingly. For the slice admission control period, a duration in the order of one second was considered in [17]. The main notations used in this paper are described in Table I.

A. Blueprint and Slice Resource Requirements

The NGMN Alliance defines the slice blueprint as a complete description of the structure, configuration, and the work flows to instantiate and control a network slice during its life-cycle [18]. In this work, we characterize a slice blueprint as follows:

- A list of NFs that the slice requires and the description of the interactions among the NFs.
- The parameters that describe the configuration of each NF.

Depending on the type of service, the system will deploy a specific version of each NF, which enables to satisfy the service requirements. For example, in the case of eMBB service, the NF providing 5G PHY layer is characterized by precise configuration parameters as large bandwidth, modulation and coding schemes with high spectral efficiency, etc. The configuration of a NF, in turn, defines the amount of resources that it requires to be deployed in the 5G system. We consider that a 5G network slice can be composed by three sets of NFs: radio $\mathcal{K}^{\mathrm{R}}$, transport $\mathcal{K}^{\mathrm{T}}$, and cloud $\mathcal{K}^{\mathrm{C}}$ (see Fig. 3), each consisting of a finite number of NFs requiring three types of resources: communication (in terms of bandwidth [MHz]), computing (in [GFLOPS/s]), and cloud storage (in [GB]). For example, typical NFs in the radio, transport, and cloud sets are the physical layer, routing, and caching functions, respectively. We use $\mathcal{K} = \mathcal{K}^{\mathrm{R}} \cup \mathcal{K}^{\mathrm{T}} \cup \mathcal{K}^{\mathrm{C}} = \{NF_1, \dots, NF_K\}$ to denote the set of all $K = (K_R + K_T + K_C)$ NFs that can be deployed in the 5G system. Let $N \geq 0$ be the number of slice requests at the NSMF. We suppose that the slice blueprint of the n^{th} slice request includes a set of NFs, denoted by $\mathcal{D}_n \subseteq \mathcal{K}$, and a set of configuration parameters for each $NF_k \in \mathcal{D}_n$, indicated by $\mathcal{L}_{n,k}$. Then, to indicate which NFs compose a request $\mathcal{D}_n$, we use the vector $\boldsymbol{\lambda_n} \in \{0,1\}^K$, whose k^{th} entry is defined as:

$$\lambda_{n,k} = \begin{cases} 1 & \text{if } NF_k \in \mathcal{D}_n, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

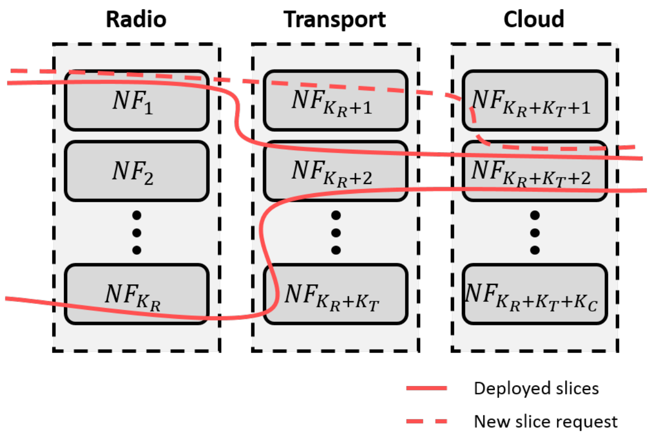


Figure 3: Mapping of network slices to the required network functions.

For each NF $NF_k \in \mathcal{D}_n$, we represent the associated configuration parameters as a set of J binary vectors[2] as follows:

$$\mathcal{L}_{n,k} = \{\boldsymbol{l}_{n,k,1}, \boldsymbol{l}_{n,k,2}, \dots, \boldsymbol{l}_{n,k,J}\}. \quad (2)$$

Then, we map the configuration parameters with the amount of communication ($d_{n,k,r}$), computing ($d_{n,k,c}$), and cloud storage ($d_{n,k,m}$) resources required by the NF NF_k of the n^{th} slice request using a model composed of a static part and a dynamic part:

$$\begin{aligned} d_{n,k,r} &= \rho_k + f_{k,r}(\mathcal{L}_{n,k}), \\ d_{n,k,c} &= \chi_k + f_{k,c}(\mathcal{L}_{n,k}), \\ d_{n,k,m} &= \mu_k + f_{k,m}(\mathcal{L}_{n,k}), \end{aligned} \quad (3)$$

where ρ_k, χ_k, and μ_k are the minimum amount of required resources for activating a given NF, $NF_k \in \mathcal{D}_n$, and $f_{k,r}(\mathcal{L}_{n,k})$, $f_{k,c}(\mathcal{L}_{n,k})$, and $f_{k,m}(\mathcal{L}_{n,k})$ are the additional amount of required resources that can be defined as a function of the configuration parameters $\mathcal{L}_{n,k}$ characterized for the NF_k. In summary, the total communication, computing, and cloud storage resource demand of the n^{th} slice request can be computed as follows:

$$\boldsymbol{T}_n = (d_{n,r}, d_{n,c}, d_{n,m}), \quad (4)$$

where $d_{n,j} = \sum_{NF_k \in \mathcal{D}_n} d_{n,k,j}, j \in \{r, c, m\}$.

B. State Space of the Proposed Slice Deployment and Orchestration Framework

Here we describe the state space of the proposed framework for the network slice deployment and orchestration. ~~In the following sections, we describe the proposed functions to optimize the network resource utilization, while considering the priorities of the different service requests.~~ The state space of the system, denoted by $\mathcal{S}$, includes the sets that describe the states of the GS queue, the BE queue, the available network resources, the number of deployed GS slices, the number of

[2]For the ease of presentation, we consider that all NFs have the same number of configuration parameters.

Table I: Main Notations and System Parameters.

Notation	Parameter
$\mathcal{S} \triangleq \mathcal{S}_g \times \mathcal{S}_b \times \mathcal{S}_p \times \mathcal{U}_g \times \mathcal{U}_b \times \mathcal{X}_p$	State space
$\mathbf{s}(t) \in \mathcal{S}$	System state
$s_g(t) \in \mathcal{S}_g, s_b(t) \in \mathcal{S}_b$	Slice requests in GS and BE queues
$\boldsymbol{s_p}(t) = (r, c, m) \in \mathcal{S}_p$	Communication [MHz], computing [GFLOPS/s], and storage [GB] resources
$u_g(t) \in \mathcal{U}_g, u_b(t) \in \mathcal{U}_b$	Deployed GS and BE slices
$\boldsymbol{x_p}(t) = (x_r, x_c, x_m) \in \mathcal{X}_p$	Allocated resources for deployed BE slices
$\boldsymbol{a}(t) = (a_g, a_b) \in \mathcal{A}$	Admission and Congestion Control action and action space
$a_g(t), a_b(t)$	Number of accepted GS and BE requests
$\Lambda_{n,n'}$	Jaccard similarity between slices n and n'
$\mathcal{B}$	Deployed Network Slice Instances
$d_{n,k,j}$	Resources of type j required by the k^{th} NF in the n^{th} request
$d_{n,j}$, $d'_{n,j}$	Resources of type j required by n^{th} slice before, after slice analytics
w_g, w_b	Rewards for each accepted GS and BE slices
$n_d(t), l_d$	Instantaneous dropped GS slices and dropping loss
$n_b(t)$, $n_g(t)$	New BE and GS requests
$R(\mathbf{s}(t), \mathbf{a}(t))$	Instantaneous reward
$Q(\mathbf{s}, \mathbf{a})$	Q-value of a state-action pair
α, γ	Learning rate and discount factor
$\boldsymbol{\phi}(\mathbf{s}, \mathbf{a})$	Feature vector
θ_i	Weight of the i^{th} feature

deployed BE slices, and the resources allocated to the BE slices. Accordingly, $\mathcal{S}$ is defined as follows:

$$\mathcal{S} \triangleq \mathcal{S}_g \times \mathcal{S}_b \times \mathcal{S}_p \times \mathcal{U}_g \times \mathcal{U}_b \times \mathcal{X}_p. \quad (5)$$

At each time slot of the proposed cycle for slice deployment and orchestration (see Fig. 2), the numbers of slice requests in the GS and BE queues are denoted respectively by $s_g \in \mathcal{S}_g = \{0, 1, \dots, Q_g\}$ and $s_b \in \mathcal{S}_b = \{0, 1, \dots, Q_b\}$, where Q_g and Q_b are the maximum length of the GS and BE queues, respectively. As for the network resource state, let $\boldsymbol{s_p} = (r, c, m) \in \mathcal{S}_p$, where $r \in \{0, 1, \dots, R\}$, $c \in \{0, 1, \dots, C\}$, and $m \in \{0, 1, \dots, M\}$, denote the available communication, computing, and cloud storage network resources, respectively. We also indicate the number of deployed GS slices, deployed BE slices, and the network resources associated to the running BE slices respectively as $u_g \in \mathcal{U}_g = \{0, 1, \dots, U_g\}$, $u_b \in \mathcal{U}_b = \{0, 1, \dots, U_b\}$ and $\boldsymbol{x_p} = (x_r, x_c, x_m) \in \mathcal{X}_p \subseteq \mathcal{S}_p$, respectively, where U_g and U_b are the maximum numbers of GS and BE slices, which can be simultaneously accepted by the system. Note that, in our optimization framework, we scale down only the resources allocated to BE slices, in order to increase the number of accepted slices. Therefore, we do not need to define a set related to the resources allocated to the GS slices.

Then, we use $n_g \in \mathcal{N}_g = \{0, 1, \dots, N_g\}$ and $n_b \in \mathcal{N}_b = \{0, 1, \dots, N_b\}$ to denote the number of new GS and BE slice requests at each time slot, where N_g and N_b are the maximum numbers of arriving requests for the GS and BE services, respectively. Let p_n^g and p_n^b be the probabilities of arriving n_g and n_b slice requests at a given time slot; accordingly, we have:

$$\sum_{n=0}^{N_g} p_n^g = 1 \qquad \text{and} \qquad \sum_{n=0}^{N_b} p_n^b = 1. \quad (6)$$

It is important to note that, a slice request is dropped when it arrives in a queue that is full[3]. Finally, we denote f^g and f^b as the departure probabilities of an accepted GS slice and an accepted BE slice at a given time slot, respectively. In addition, when a given slice leaves the system, all its allocated resources are immediately released.

III. Slice Analytics Procedure

As mentioned in Section II, one of the role of the NSMF is to elaborate and update the slice blueprint, which clearly identifies the NFs required by each network slice in the 5G system (see Fig. 3). In the proposed framework, we classify each required NF either as a dedicated NF or as a shared NF with respect to the other NFs required by the other (queued or actively deployed) network slices. Accordingly, the resource usage of the 5G system can be optimized if multiple NFs shared across different slices can (even partially) share common network resources at the same time. For instance, two slices deployed in the same geographical area can share the same base stations if the available capacity is large enough to satisfy the overall demands. Moreover, during a sport event, a single operator can use network slicing to accommodate the wireless services required by different broadcasters. In this case, a large part of the communication resources can be shared by these slices to broadcast the common content (e.g., the video) to different users, while specific content (e.g., the speaker's voice) can use dedicated resources. From the operator's point of view, this approach enables to increase the number of accepted slice requests (generating higher revenue), decrease the slice deployment cost, and limit the service creation time.

Using the 3GPP terminology, we denote the set of NF instances and the associated network resources deployed to

[3]The queues capacity is limited to avoid long delay for the slice requests waiting in the queues.

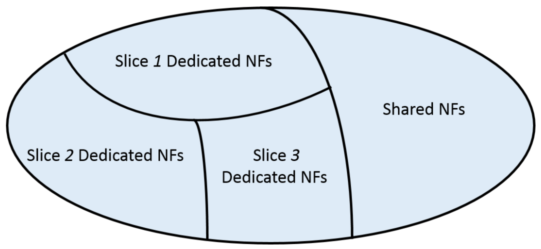


Figure 4: An illustration of shared and dedicated NFs in a network slice instance. A slice is composed of dedicated NFs and shared NFs with other slices.

satisfy one or multiple slice requests as a Network Slice Instance (NSI). An NSI is composed of NFs shared between two or more slices, as well as dedicated NFs, as shown in Fig. 4. To efficiently map a network slice request to an NSI, we propose a procedure, namely SA, that analyzes the network slice blueprints, in order to evaluate the *similarities* among network slices and identify the NSI that can serve the new slice request with a minimum amount of additional network resources (see Algorithm 1).

In SA, we use $NSqueue(n)$ and $\mathcal{B}$ to denote the n^{th} slice request and the set of deployed NSIs. First, SA compares the NFs required by the n^{th} slice request $\boldsymbol{\lambda}_n$ with those associated with each running NSI, by calculating the Jaccard similarity as follows [19]:

$$\Lambda_{i,n} = \frac{\boldsymbol{\lambda}_i \boldsymbol{\lambda}_n}{\|\boldsymbol{\lambda}_i\| + \|\boldsymbol{\lambda}_n\| - \boldsymbol{\lambda}_i \boldsymbol{\lambda}_n},\ i \in \mathcal{B}, \tag{7}$$

where $\boldsymbol{\lambda}_i$ and $\boldsymbol{\lambda}_n$ are defined in (1). Based on the computed similarities, SA chooses an NSI $i^* \in \mathcal{B}$ according to one of the association algorithms presented in Section III-A. If the selected NSI can integrate the slice request, e.g., there are no resource isolation constraints, the slice analytics computes the degree of similarity for each NF required by the $NSqueue(n)$ and shared with the other slices associated to the selected NSI i^* (see (13)), and accordingly evaluates the additional amount of resources required by the NSI to accommodate the slice request n, $\boldsymbol{T}'_n$ (15). If the slice request cannot be assigned to any of the existing NSIs, a new NSI will be prepared. If the request is accepted by the CSACC, presented in Section IV, either the new NSI is instantiated or the NSI i^* is updated. Finally, the virtual infrastructure manager (see Fig. 1) executes the resource allocation procedure.

A. *Similarity-Based Slice Association*

The first step in SA is to assign a new slice request to an existing NSI. This assignment is based on one of the two proposed strategies:

- The SA assigns a new slice demand to the existing NSI which maximizes the Jaccard similarity. The advantage of this solution is the low complexity, and may be implemented in a fast time-scale, in the order of one second.
- The SA uses spectral clustering to create new NSIs which maximize the resource sharing in the 5G system. This is an iterative solution, and thus more complex than the fast time-scale approach. Accordingly, it may be implemented at a slower time-scale, e.g., each 60 seconds.

Algorithm 1 Slice Analytics

```
Input: $NSI = \{NSI\_id, \boldsymbol{\lambda}, \boldsymbol{T}\} \leftarrow$ deployed NSIs;
    $NSqueue = \{NSI\_id, \boldsymbol{\lambda}, \boldsymbol{T}\} \leftarrow$ queued NSs;
    $NSassociated = \{NSI\_id, \boldsymbol{\lambda}, \mathcal{L}, \boldsymbol{T}\} \leftarrow$ NSs associated to deployed NSIs;
Output: Updated $NSassociated$.
1: for $n = 1$:Length(NSqueue) do
2:     Calculate Jaccard similarity (7) between $NSqueue(n)$ and each NSI $i \in \mathcal{B}$.
3:     Use the selected association algorithm to identify the proper NSI $i^* \in \mathcal{B}$.
4:     if NSI $i^*$ can integrate $NSqueue(n)$ then
5:         Calculate cosine similarity (12) between $NSqueue(n)$ and $NSassociated$ to $i^*$.
6:         Evaluate the new $\boldsymbol{T'}$ of $NSI(i^*)$ (15).
7:         if $\boldsymbol{T'}$ of $NSI(i^*)$ can be satisfied (see CSACC) then
8:             Update $NSI\_id$ of $NSqueue(n)$, $\boldsymbol{T'}$ of $NSI(i^*)$, and $NSassociated$ to $i^*$.
9:         end if
10:        if $NSI\_id$ of $NSqueue(n)$ is empty then
11:            if $\boldsymbol{T}$ of $NSqueue(n)$ can be satisfied (see CSACC) then
12:                Create the new $NSI$ $n$ and add $n$ to $\mathcal{B}$.
13:                Set $\boldsymbol{T}$ of $NSI(n)$, $\boldsymbol{\lambda}$ of $NSI(n)$, and $NSI\_id$ of $NSqueue(n)$.
14:            end if
15:        end if
16:    end if
17: end for
```

1) Jaccard similarity-based assignment: With the first proposed strategy, the selected NSI is the one that shares the highest number of NFs with the slice request. Accordingly, given a set of deployed NSI $\mathcal{B}$, the SA attempts to assign the slice request j to the NSI i^* such that

$$i^* = \underset{i \in \mathcal{B}}{\operatorname{argmax}}\ \Lambda_{ji}. \tag{8}$$

2) Spectral clustering-based assignment: To further optimize the resource usage and increase the number of accepted slice requests, the second proposed strategy periodically re-clusters the running slices in new NSIs. In order to achieve this goal, we implement a normalized spectral clustering algorithm [20] based on the Jaccard similarity among running network slices. In the spectral clustering, the deployed slices are represented as nodes of a connected graph and clusters are found by partitioning this graph based on their spectral decomposition into subgraphs. This clustering scheme is simple to implement and typically outperforms other clustering algorithms such as the K-means algorithm [21].

Let $\mathcal{B} = \{1, \ldots, N_R\}$ denote the set of running slices with NF requests $\{\lambda_1, \ldots, \lambda_{N_R}\}$; we compute the affinity matrix $\mathbb{A}$, whose elements are the Jaccard similarity between two running slices as follows:

$$\mathbb{A}_{n,n'} = \begin{cases} \Lambda_{n,n'} & \text{if } n \neq n', \\ 0 & \text{otherwise,} \end{cases} \quad n, n' \in \mathcal{B}, \tag{9}$$

where $\Lambda_{nn'}$ is computed as in (7). Then, we derive from $\mathbb{A}$ the corresponding diagonal matrix $\mathbb{D}$, whose (n, n) element is the sum of the nth row of $\mathbb{A}$, i.e., $d_{nn} = \sum_{n' \in \mathcal{B} \backslash n} \Lambda_{n,n'}$ and the associated normalized laplacian matrix as:

$$\mathbb{L} = \mathbb{D}^{-1/2} \mathbb{A} \mathbb{D}^{-1/2}. \tag{10}$$

Afterwards, SA computes the normalized eigenvectors of $\mathbb{L}$ [20] and clusters the first k eigenvectors using K-means [22]. The number of clusters k is obtained such that all eigenvalues $(1, \dots, k)$ are very small, but the $k^{th}+1$ is relatively large [21].

By deploying new NSIs that maximize the similarity within their slices, this strategy aims to achieve an improved performance as compared to the previously discussed Jaccard similarity-based assignment. However, the spectral clustering increases the complexity of SA to $\mathcal{O}(N_R^3)$ [23] as compared to $\mathcal{O}(N_R)$, which characterizes the Jaccard similarity-based assignment. When considering the expected number of slices in the 5G network, the spectral cluster performance is not limited by its complexity; however, this approach should be implemented with a limited frequency or only when the system is overloaded.

B. Intra-NSI Similarity for Resource Mutualization

Both the SA strategies proposed in Section III-A allow to identify an already existing NSI, which is appropriate to support a new communication service. However, to satisfy the specific requirements of the new service request, the deployed NSI may require additional network resources. Thus, before finalizing this process, the new NSI network requirements must be evaluated and the associated resource request must be accepted by the CSACC (see Section IV). In this work, we assume that the amount of resources that can be mutualized by the existing NSI and the new service request depends on the shared NFs and their associated configuration parameters. Specifically, we use $\mathcal{B}_{i,k}$ to denote the set of slices included in NSI $i \in \mathcal{B}$ and requiring NF $k \in \mathcal{K}$; then, for each pair of slices $n,\ n' \in \mathcal{B}_{i,k}$, we define the similarity between the sets of configuration parameters $\mathcal{L}_{n,k}$ and $\mathcal{L}_{n',k}$ (see (2)) as follows:

$$C(\mathcal{L}_{n,k}, \mathcal{L}_{n',k}) = \sum_{j=1}^{J} h(\boldsymbol{l}_{n,k,j}, \boldsymbol{l}_{n',k,j}), \tag{11}$$

where J is the number of parameters of the NF k. Moreover, h computes the cosine similarity[4] between two values of the j^{th} parameter of NF k, as follows:

$$h(\boldsymbol{l}_{n,k,j}, \boldsymbol{l}_{n',k,j}) = \frac{\boldsymbol{l}_{n,k,j} \boldsymbol{l}_{n',k,j}}{\|\boldsymbol{l}_{n,k,j}\| \|\boldsymbol{l}_{n',k,j}\|}, \tag{12}$$

where $\| \cdot \|$ is the euclidean norm operator. Considering all network slices $n' \neq n \in \mathcal{B}_{i,k}$, we define the largest similarity for slice request n with respect to NF k as follows:

$$\sigma^*_{n,i,k} = \max_{\forall n' \neq n \in \mathcal{B}_{i,k}} \{C(\mathcal{L}_{n,k}, \mathcal{L}_{n',k})\}, i \in \mathcal{B}. \tag{13}$$

[4]Cosine similarity may not be suitable for all NFs, e.g., for measuring the similarity among contents cached in mobile edge clouds, other schemes can be adopted, for example, from [24], [25].

Then, the communication, computing, and cloud storage resources required for NF k when including the slice request n as part of the NSI i can be computed respectively as follows:

$$\begin{aligned} d'_{n,k,r} =& \rho_k \beta(\sigma^*_{n,i,k}) + (1 - \sigma^*_{n,i,k}) f_{k,r}(\mathcal{L}_{n,k}), \\ d'_{n,k,c} =& \chi_k \beta(\sigma^*_{n,i,k}) + (1 - \sigma^*_{n,i,k}) f_{k,c}(\mathcal{L}_{n,k}), \\ d'_{n,k,m} =& \mu_k \beta(\sigma^*_{n,i,k}) + (1 - \sigma^*_{n,i,k}) f_{k,m}(\mathcal{L}_{n,k}). \end{aligned} \tag{14}$$

We recall that ρ_k, χ_k, and μ_k are the minimum amount of required resources for activating a given NF, and $f_{k,r}(\mathcal{L}_{n,k})$, $f_{k,c}(\mathcal{L}_{n,k})$, and $f_{k,m}(\mathcal{L}_{n,k})$ denote the amount of resources required by the NF k as a function of its configuration parameters $\mathcal{L}_{n,k}$ (see (3)). Moreover, $\beta(\cdot)$ is a step function that is equal to one if its argument is positive and zero otherwise, i.e., the static part of the resource requirements of NF k is not needed if $\sigma^*_{n,i,k} > 0$. To conclude, the refined resource demands of a network slice request n after the SA procedure can be computed as follows:

$$\boldsymbol{T}'_n = \left(d'_{n,r}, d'_{n,c}, d'_{n,m}\right), \tag{15}$$

where $d'_{n,j} = \sum_{NF_k \in \mathcal{D}_n} d'_{n,k,j}, j \in \{r, c, m\}$.

IV. Cross-Slice Admission and Congestion Controller

In this section, we propose a reinforcement learning framework that optimizes the trade-off between the number of accepted slices and the dropping probability of the network slice requests with high priorities, which in turns affects the long-term system revenue. This is a challenging problem as the slice request arrival and departure probabilities are unknown and the network resources are limited (see Section II-B). Thus, minimizing the dropping probability of high-priority slice requests requires to limit the acceptance of low-priority slice requests, thereby reducing the operator revenue. To achieve this goal, we design a functionality named CSACC (shown in Fig. 1) aiming of maximizing the resource utilization by accepting new slice requests while taking into account the queue status, the resource availability, the resource requirements, and slice priorities at the runtime. The CSACC decides to 1) scale down the resource allocated to a deployed BE slice and 2) accept one or multiple requests. More specifically, by scaling down resources allocated to a BE slice, the CSACC can accept new GS slice requests or approve the demand for additional resources from a deployed GS slice.

A. Action and Reward Models

The CSACC monitors the overall system state space, defined in (5), and uses a reinforcement learning algorithm to learn the optimal policy that finds the proper admission and congestion control actions to maximize the long-term system reward. At each time slot t (see Fig. 2), the CSACC selects an action $\boldsymbol{a} \in \mathcal{A}$ that determines the number of accepted GS and BE requests a_g and a_b, respectively. Thus, the CSACC action space is defined as follows:

$$\mathcal{A} \triangleq \{\boldsymbol{a} = (a_g, a_b)\}. \tag{16}$$

It is important to note that the slice requests in each queue are served in a first-input-first-output fashion, e.g., if $a_g = 2$, the first two slice requests of the GS queue are served. The actions chosen at each time slot must ensure that the number of BE (resp. GS) slice requests accepted does not exceed the actual number of BE (resp. GS) slice requests in the queue:

$$a_g(t) \leq s_g(t) \quad \text{and} \quad a_b(t) \leq s_b(t). \tag{17}$$

In addition, the constraints in (18) guarantee that, for the accepted BE slices, the sum of allocated resources for each resource type is less than or equal to the current resource availability:

$$\begin{aligned} \sum_{n=1}^{a_b(t)} d'^{b}_{n,r} &\leq r(t), \\ \sum_{n=1}^{a_b(t)} d'^{b}_{n,c} &\leq c(t), \\ \sum_{n=1}^{a_b(t)} d'^{b}_{n,m} &\leq m(t). \end{aligned} \tag{18}$$

Finally, the constraints in (19) ensure that the acceptance of new GS slices does not excessively degrade the quality of service of the running BE slices, by guaranteeing that a minimum amount of resources $\delta_j,\ j \in \{r, c, m\}$, is maintained at each deployed BE slice:

$$\begin{aligned} \sum_{n=1}^{a_g(t)} d'^{g}_{n,r} + \sum_{n=1}^{a_b(t)} d'^{b}_{n,r} &\leq r(t) + x_r(t) - \delta_r u_b(t), \\ \sum_{n=1}^{a_g(t)} d'^{g}_{n,c} + \sum_{n=1}^{a_b(t)} d'^{b}_{n,c} &\leq c(t) + x_c(t) - \delta_c u_b(t), \\ \sum_{n=1}^{a_g(t)} d'^{g}_{n,m} + \sum_{n=1}^{a_b(t)} d'^{b}_{n,m} &\leq m(t) + x_m(t) - \delta_m u_b(t). \end{aligned} \tag{19}$$

The aim of CSACC is to increase the operator revenue by maximizing the number of accepted slices, while limiting the probability that an arriving GS slice request is dropped because the queue is full, which leads to large revenue losses. Accordingly, to model the instantaneous reward associated with a state-action pair, we define a function that takes into account the numbers of accepted GS and BE slices as well as the number of dropped GS requests as follows:

$$R(\mathbf{s}(t), \mathbf{a}(t)) = a_g(t) w_g + a_b(t) w_b - n_d(t) l_d, \tag{20}$$

where w_g and w_b are the rewards for each accepted GS and BE slice, respectively. In addition, l_d is the cost for dropping a GS slice request, and $n_d(t)$ is the number of instantaneous dropped GS slices, which can be computed as follows:

$$n_d(t) = \max\{s_g(t) - a_g(t) + n_g(t) - Q_g, 0\}, \tag{21}$$

where Q_g is the maximum length of the GS queue (see Section II-B).

B. The SARSA Algorithm

SARSA is an online reinforcement learning algorithm aiming to find a stationary policy that associates a given system state with a proper action such that the expected total discounted reward is maximized. We define the expected total discounted reward counting from an initial state-action pair $(\mathbf{s}, \mathbf{a})$ over an infinite time horizon as follows [15]:

$$Q(\mathbf{s}, \mathbf{a}) = \mathbb{E}\left\{\sum_{t=0}^{\infty} \gamma^t R(\mathbf{s}(t), \mathbf{a}(t)) \mid \mathbf{s}_0 = \mathbf{s},\ \mathbf{a}_0 = \mathbf{a}\right\}, \tag{22}$$

where $R(\mathbf{s}(t), \mathbf{a}(t))$ is the instantaneous reward defined in (20) and $\gamma \in [0, 1)$ is a discount factor that determines the importance of future rewards. Reinforcement learning algorithms typically represent Q-function with a tabular format (i.e., a look-up-table), and the Q-values are arbitrarily initialized. Then, during the learning phase, the algorithm iteratively updates the Q-values when moving between state-action pairs (see the Algorithm 2), until convergence. At the end of the learning phase, if each state-action pair has been visited sufficiently often, the Q-values converge to an intermediate minimum, from which the optimal action can be computed as follows:

$$\mathbf{a}^* = \underset{\mathbf{a} \in \mathcal{A}}{\text{argmax}}\ Q(\mathbf{s}, \mathbf{a}),$$

where $\mathcal{A}$ is the set of actions defined in (16). To ensure the convergence of SARSA to the optimal policy and avoid the local minimum, it is necessary, during the learning phase, to randomly switch from one policy to another using e.g., ϵ-greedy or Boltzmann exploration schemes [26], while slowly decaying the exploration rate, e.g., ϵ for the ϵ-greedy, to zero.

Algorithm 2 SARSA

```
Input: action set $\mathcal{A}$, reward function $R$, and parameters $\alpha, \gamma \in [0, 1)$, and $\epsilon > 0$.
1: Randomly initialize $Q(\mathbf{s}, \mathbf{a}), \forall \mathbf{s} \in \mathcal{S}, \mathbf{a} \in \mathcal{A}$;
2: for $t := 1$ to $T$ do
3:     Observe the current state $\mathbf{s}$;
4:     Select an action $\mathbf{a} \in \mathcal{A}$, using policy derived from $Q$ (e.g. $\epsilon$-greedy), observe the reward $R$ and the new state $\mathbf{s}'$;
5:     Take $\mathbf{a}' \in \mathcal{A}$ from $\mathbf{s}'$, using policy derived from $Q$;
6:     Update $Q(\mathbf{s}, \mathbf{a})$ as follows: $Q(\mathbf{s}, \mathbf{a}) \leftarrow Q(\mathbf{s}, \mathbf{a}) + \alpha [R + \gamma Q(\mathbf{s}', \mathbf{a}') - Q(\mathbf{s}, \mathbf{a})]$;
7:     $\mathbf{s} \leftarrow \mathbf{s}'$, $\mathbf{a} \leftarrow \mathbf{a}'$;
8: end for
```

Although the reinforcement learning model can be successfully used to address small-scale MDP problems, it becomes impractical in realistic cases characterized by large state space and action set, as computational requirements of reinforcement learning grow exponentially with the number of state variables, i.e., the so-called *curse of dimensionality*.

In this work, when the SA is implemented, the amount of resources required by a new slice to be deployed depends on the similarity with the other slices being in the same NSI (14). Therefore, in (5), the sizes of sets of the available resources $\mathcal{S}_p$ and the resources allocated to the BE slices $\mathcal{X}_p$ become very large, and conventional reinforcement learning algorithms cannot be used to solve the cross-slice congestion and admission control. The state space size can be computed as follows:

$$|\mathcal{S}| = |\mathcal{S}_g| \cdot |\mathcal{S}_b| \cdot |\mathcal{S}_p| \cdot |\mathcal{U}_g| \cdot |\mathcal{U}_b| \cdot |\mathcal{X}_p|.$$

Let k denote the number of possible similarity values that can be obtained in (13), when the SA is implemented. In this

Algorithm 3 SARSA With Linear Function Approximation

```
Input: action set $\mathcal{A}$, reward function $R$, feature vector $\boldsymbol{\phi}(\mathbf{s},\mathbf{a})$, and
    parameters $\alpha,\gamma \in [0,1)$, and $\epsilon > 0$.
1: Randomly initialize the feature weights $\boldsymbol{\theta}(\mathbf{s},\mathbf{a})$.
2: for $t := 1$ to $T$ do
3:     Select an action $\mathbf{a} \in \mathcal{A}$, using policy derived from $Q$ (e.g. $\epsilon$-greedy), observe the reward $R$ and the new state $\mathbf{s}'$;
4:     Take $\mathbf{a}' \in \mathcal{A}$ from $\mathbf{s}'$, using policy derived from $Q$;
5:     Let $\boldsymbol{e}_t = \boldsymbol{\phi}(\mathbf{s},\mathbf{a})$, and $k_t = R_t(\mathbf{s},\mathbf{a}) + \gamma Q_{\boldsymbol{\theta}_t}(\mathbf{s}',\mathbf{a}') - Q_{\boldsymbol{\theta}_t}(\mathbf{s},\mathbf{a})$
6:     Update weights as follows: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t k_t \boldsymbol{e}_t$
7:     Replace $\mathbf{s} \leftarrow \mathbf{s}'$ and $\mathbf{a} \leftarrow \mathbf{a}'$
8: end for
```

case, the size of $\mathcal{X}_p$ is equal to $k^3(X_r+1)(X_c+1)(X_m+1)$; thus, $|\mathcal{S}|$ increases dramatically and learning the optimal policy with a standard reinforcement learning algorithm becomes infeasible. To deal with this issue, in the next section, we design an approximation function that enables to generalize the Q-function and accordingly to find the optimal policy in high complexity systems. Note that, SARSA, being an on-policy scheme, is characterized by milder convergence conditions compared to more classical (off-policy) reinforcement learning schemes (such as Q-Learning) when used with linear function approximation (LFA) [15]. Both linear and non-linear function approximation methods can be used to learn a generalized policy and apply reinforcement learning in realistic and complex problems. On the one hand, LFA requires skillful design of the features used as a linear basis to represent the Q-function. On the other hand, non-linear methods are characterized by a larger design complexity: for instance, when using artificial neural networks, it is necessary to select a proper class of the neural network (such as fully connected, convolution, or recurrent), the number of layers, the number of weights, and the activation functions. For this reason, in the next section we present a scheme based on SARSA with LFA to find the optimal strategy for cross-slice admission and congestion control.

V. SARSA With Linear Function Approximation

In this section, we discuss how the LFA can be integrated into the SARSA to generalize the optimal policy and deal with the curse of dimensionality for large-scale problems. When using SARSA with LFA, the Q-function can be represented as a linear combination of a designed feature vector $\boldsymbol{\phi}(\mathbf{s},\mathbf{a}) = [\phi_1(\mathbf{s},\mathbf{a}),\phi_2(\mathbf{s},\mathbf{a}),\ldots,\phi_{\mathrm{F}}(\mathbf{s},\mathbf{a})]$. Accordingly, the Q-function can be approximated as follows:

$$Q(\mathbf{s},\mathbf{a}) \approx Q_\theta(\mathbf{s},\mathbf{a}) = \theta_0 + \sum_{i=1}^{\mathrm{F}} \theta_i \phi_i(\mathbf{s},\mathbf{a}), \qquad (23)$$

where θ_i is the weight of the i^{th} feature and θ_0 is a bias term. Therefore, instead of explicitly learning the optimal Q-values for all state-action pairs, it is sufficient to learn the values of the $(F+1)$ weights related to the feature functions, i.e., the complexity of the learning process scale down from $\mathcal{O}(|\mathcal{S}||\mathcal{A}|)$, with SARSA, to $\mathcal{O}(F+1)$ with SARSA with LFA. To learn the weights, the classical approach is to minimize the Mean-Squared Error (MSE) over a probability distribution of the state-action pairs $\mathcal{Z}$, i.e.,

$$\zeta(\theta) = \mathbb{E}_{\mathcal{Z}}[(Q(\mathbf{s},\mathbf{a}) - Q_\theta(\mathbf{s},\mathbf{a}))^2]. \qquad (24)$$

Stochastic Gradient Descent (SGD) is the most effective and popular method used in machine learning to find the local minimum of the MSE in (24) by updating the set of weights in (23) in the opposite direction of the gradient of the objective function $\nabla_\theta \zeta(\theta)$. Moreover, instead of computing the full expectations in this gradient, the SGD iteratively updates the weights after each training sample, as follows:

$$\theta \leftarrow \theta - \frac{\alpha}{2}\nabla_\theta \zeta(\theta) = \theta + \alpha[(Q(\mathbf{s},\mathbf{a}) - Q_\theta(\mathbf{s},\mathbf{a}))\boldsymbol{\phi}(\mathbf{s},\mathbf{a})], \quad (25)$$

where the learning rate α determines the size of the steps in the SGD. For reinforcement learning, $Q(\mathbf{s},\mathbf{a})$ is assumed to be unknown, and therefore, in (25), we use the bootstrapping technique [27], which consists in replacing the missing observation with an unbiased estimate $R(\mathbf{s},\mathbf{a}) + \gamma Q_{\boldsymbol{\theta}}(\mathbf{s}',\mathbf{a}')$. Since each estimate is an unbiased estimate, the SGD converges to a local minimum of the MSE if the learning rate α decreases appropriately during the learning phase. The algorithm of SARSA with LFA is sketched in Algorithm 3. To conclude this section, we discuss the design of the feature functions for the proposed CSACC. LFA methods have attracted the attention of the research community because of not only their convergence guarantees, but also the possibility to integrate feature functions that add prior knowledge to reinforcement learning systems. Intuitively, the features design should take into account different state and action spaces along which approximation may be appropriate. Accordingly, we construct feature functions that consider the actions, the state space characteristics, and how state and action pairs jointly define the instantaneous reward in (20). Fig. 5 summarizes all the feature functions adopted by the CSACC and shows its main procedures. The CSACC evaluates whether to accept slice requests based on the queue status. If both queues are empty, no slice is accepted. If the GS queue is empty, the CSACC evaluates the required resources to accept BE slices and decides the number of accepted BE slice requests based on the available resources. In contrast, when the GS queue has slice requests awaiting to be accepted, the CSACC checks the resource availability while prioritizing GS slice requests over BE slice requests, in particular if the status of the GS queue is critical, and future GS requests may be dropped. If the resources are not sufficient, the CSACC evaluates the amount of resources that can be reduced from running BE slices, and accepts GS slice requests accordingly. Otherwise, if the available resources allow to accept both BE and GS slices, the CSACC accepts slices from both queues. In this work, we consider that the GS queue is critical if the number of GS slice requests, s_g, is higher than a given threshold $\hat{q}_g$, and additional future requests may be dropped.

VI. Simulation Results

To evaluate the performance of proposed framework for slice deployment and orchestration, we consider a 5G system using network slicing to meet the heterogeneous requirements

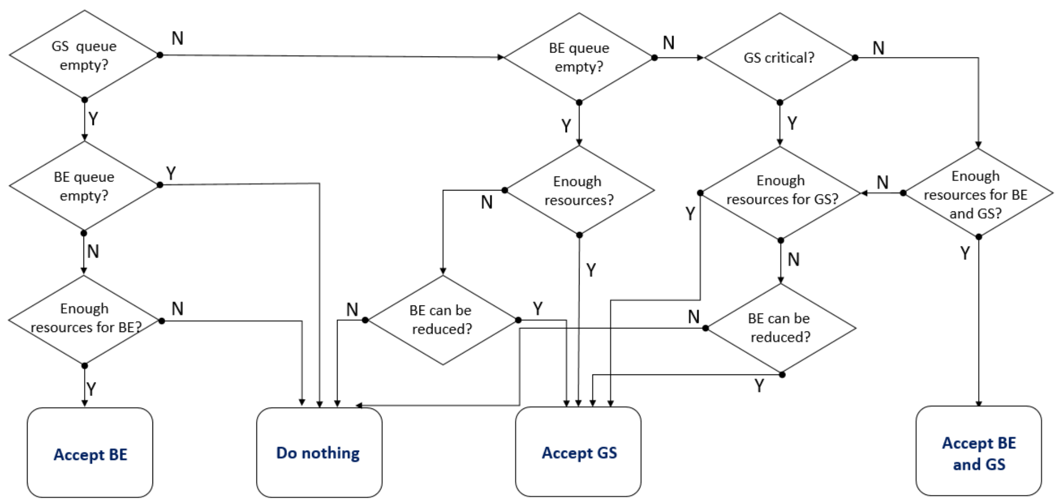


Figure 5: Cross-slice admission and congestion control flowchart.

Table II: Slice templates in terms of required NFs.

	NF 1	NF 2	NF 3	NF 4	NF 5	NF 6	NF 7	NF 8	NF 9	NF 10
Slice template 1		x		x	x			x		x
Slice template 2		x		x	x		x			x
Slice template 3	x			x		x		x	x	
Slice template 4		x		x		x		x	x	
Slice template 5	x		x		x		x		x	
Slice template 6	x		x			x	x		x	

of future 5G services. We consider that each slice request arriving in the system is related to one of the 6 slice templates shown in Table II. Specifically, the request n requires a set $\mathcal{D}_n$ of 5 NFs among the $K = 10$ NFs available in the 5G system, and each NF $NF_k \in \mathcal{D}_n$ is characterized by a set of configuration parameters $\mathcal{L}_{n,k}$ represented by one binary vector $J = 1$ of length 2. We assume that each NF needs 40 MHz, 40 GFLOPS/s, and 40 GB in terms of communication, computing, and cloud storage resources, respectively; therefore, the total communication $d_{n,r}$, computing $d_{n,c}$, and cloud storage $d_{n,m}$ demand for a slice is equal to 200 MHz, 200 GFLOPS/s, and 200 GB (see (4)).

During the congestion control phase (see (19)), the minimum amount of resources for running a BE slice i.e., δ_j, $j = \{r, c, m\}$, is equal to 100 MHz, 100 GB, and 100 GFLOPS/s. Overall, we assume that the network has 800 MHz, 800 GFLOPS/s, and 800 GB communication, computing, and cloud storage resources to satisfy the requirements of the deployed slices. Regarding the learning parameters, we initially set $\alpha = 0.5$ and $\epsilon = 1$; then, during the learning process, their values are iteratively reduced to strike a balance between performance and learning speed. We set the discount factor $\gamma = 0.8$ by random search, i.e., by cross-checking the obtained results for different values of the hyper-parameter. Finally, Table III describes the main simulation parameters.

Table III: Simulation parameters.

Notation	Parameter	Value
$d_{n,j}$	n^{th} slice resource request	200
R, C, M	Network resources	800
p^b	BE request arrival probability	0.85
f^b, f^g	Departure probabilities of GS, BE slices	0.35
N_b, N_g	Maximum number of arriving BE, GS requests	2
Q_b, Q_g	Maximum length of the slice request queues	4
w_b	Revenue for accepting a BE request	1
w_g	Revenue for accepting a GS request	1.7
l_d	Cost for dropping a GS request	$5w_g$
ρ_k, χ_k, μ_k	Resources for activating a NF	0

A. SARSA with LFA for Cross-Slice Admission Control

In this section, we focus only the slice admission control function (AC), i.e., we do not allow the controller to reduce the resources allocated to the BE slices, in order to increase the number of deployed slices. Specifically, we evaluate the results obtained when using SARSA with LFA for different values of GS acceptance reward w_g and GS arrival probability p^g. We compare the performance of the reinforcement learning scheme with a greedy scheme that selects the actions that maximize an immediate reward, i.e., the number of accepted slices at each time slot.

Fig. 6 shows the average reward obtained by the greedy scheme and SARSA with LFA as a function of the GS admission reward w_g and arrival probability p^g. First, we can see that, for each value of w_g, there is an optimal value of p^g that maximizes the average reward obtained by the SARSA with LFA. Increasing p^g enhances the average reward until when the slice arrival process can be efficiently managed by the AC; however, continuing to increase p^g beyond this optimal value decreases the system performance due to the limited queue size and available resources. In addition, we observe that for $w_g = 0.7$ and $w_g = 1$, the SARSA with LFA

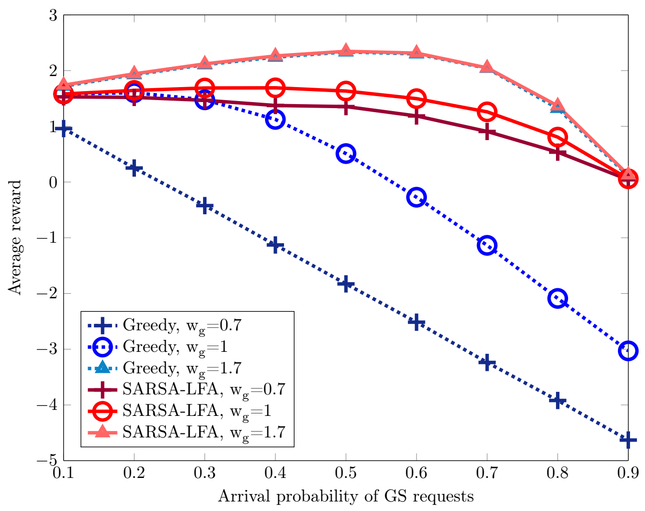


Figure 6: Average reward obtained by the greedy scheme and SARSA with LFA as a function of GS admission reward and arrival probability.

achieves a much greater average reward than that of the greedy scheme, especially for higher values of p^g. In fact, the greedy scheme is not able to predict the event of dropping new GS slice requests. In contrast, for very large value of GS reward, i.e., $w_g = 1.7$, both the greedy and SARSA with LFA achieve the same performance, since, in this case, they both prefer to always accept a GS slice request rather than a BE one.

To better understand this performance, we show in Fig. 7 the probabilities of accepting and dropping a slice request under different values of p^g and w_g when using SARSA with LFA and the greedy scheme. In Fig. 7(a), we observe that, for $w_g = 0.7$, SARSA with LFA achieves higher slice acceptance probabilities than those of the greedy scheme. Moreover, Fig. 7(d) shows that SARSA with LFA can reduce the GS dropping probability at the cost of a higher BE dropping probability. This is due to the fact that when $w_g = 0.7$, the greedy scheme prioritizes BE slices, which provides a larger instantaneous reward. In contrast, SARSA with LFA can learn and balance the reward perceived for accepting new slices and the cost for dropping new GS requests, and thus achieving the best performance in terms of average reward as shown in Fig. 6.

Fig. 7(b) shows that, for $w_g = 1$, the greedy scheme achieves higher slice acceptance probabilities as compared to SARSA with LFA when p^g is larger than 0.5. Also, we can see in Fig. 7(e) that SARSA with LFA and the greedy scheme have similar performance in terms of dropping probabilities when p^g is lower than 0.3. However, for larger values of p^g, the greedy scheme leads to GS and BE dropping probability close to 0.5. This performance is not acceptable for GS slices; in fact, for large values of p^g, SARSA with LFA prioritizes GS slices over the BE requests in order to limit the associated dropping probability. Finally, as expected, when $w_g = 1.7$, SARSA with LFA and the greedy scheme achieve similar performance in terms of slice acceptance and dropping probabilities (see Fig. 7(c) and 7(f)).

These results confirm the capability of the proposed AI algorithm to adjust the CSACC strategy to the arrival and departure statistics of the slice requests accordingly, in order to maximize the long-term average reward.

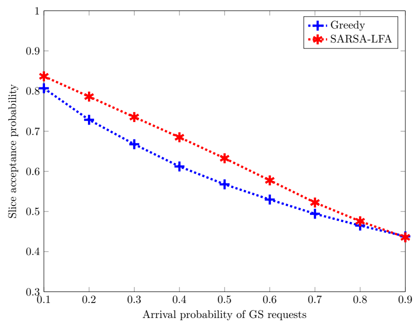
B. SARSA with LFA for Cross-Slice Admission and Congestion Control

To assess the impact of the congestion control function for slice deployment and orchestration, we compare the system performance by implementing two strategies, i.e., the strategy using only admission control (AC) and the strategy using both cross-slice admission and congestion control (CSACC). Fig. 8(a) shows the average reward achieved by SARSA with LFA when implementing AC (plus marked line) and CSACC (star marked line). As expected, the CSACC enhances the average reward of the system since it enables to increase the number of the average accepted slices. Specifically, CSACC enables to improve the average reward up to 23% with respect to the reward achieved by the simpler AC scheme. However, we can observe that when the GS arrival probability p^g is larger than 0.7 the two approaches have the same performance. The system under investigation is not well-dimensioned for this range of values, where both the controllers stop to accept BE slices requests to limit the GS slice dropping probability. Therefore, here, implementing the CC may not lead to significant benefits. This finding is verified in the results in Fig. 8(b), where we can observe the impact of the CSACC in terms of slice dropping probability. Specifically, for low and medium values of p^g, the CSACC improves the system performance by reducing the number of dropped BE slice requests. Without the CSACC, the system has to limit the acceptance of low-priority BE slices, in order to save resources for future GS slice requests. In contrast, the proposed scheme allows to increase the percentage of accepted BE slices since part of their resources can be opportunistically allocated to new GS requests when needed.

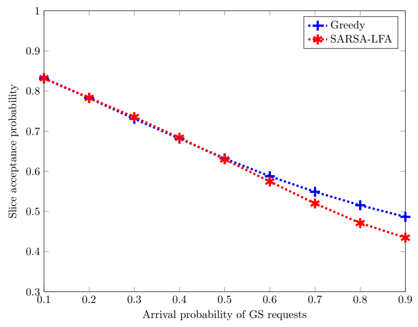
C. Cross-Slice Admission and Congestion Control with the Slice Analytics

In this section, we study the impact of the slice analytics (SA) on the system performance. The SA function optimizes the assignment of new slice requests to existing NSIs such that the additional required resources to deploy them are limited. Here, our goal is to evaluate the performance of two assignment algorithms presented in Section III-A, i.e., the Jaccard similarity-based and spectral clustering-based assignment schemes.

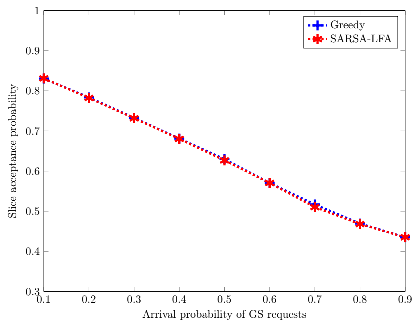
Fig. 9(a) shows the slice dropping probability as a function of GS arrival rate achieved when using only the cross-slice AC, the AC in conjunction with the Jaccard similarity scheme, and the AC in conjunction with the spectral clustering scheme. We observe that both the proposed strategies enhance the system performance achieved by the AC by decreasing the dropping probabilities for both BE and GS slice requests. However, in this case the two SA algorithms achieve the same performance. Fig. 9(b) shows the slice dropping probability as a function of GS arrival rate achieved when using only the CSACC, the CSACC in conjunction with the Jaccard similarity scheme, and



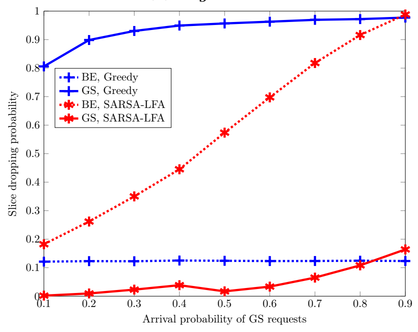
(a) $w_g = 0.7$



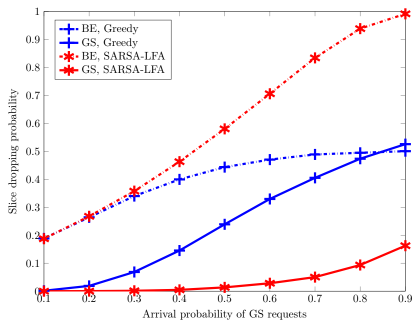
(b) $w_g = 1$



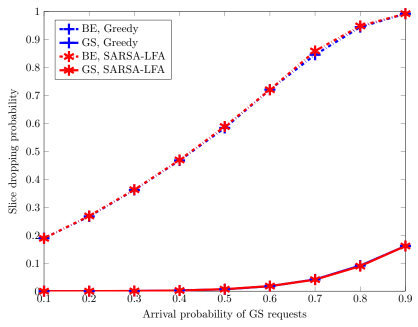
(c) $w_g = 1.7$



(d) $w_g = 0.7$



(e) $w_g = 1$



(f) $w_g = 1.7$

Figure 7: Slice acceptance and dropping probabilities obtained by the greedy scheme and SARSA with LFA as a function of GS admission reward and arrival probability.

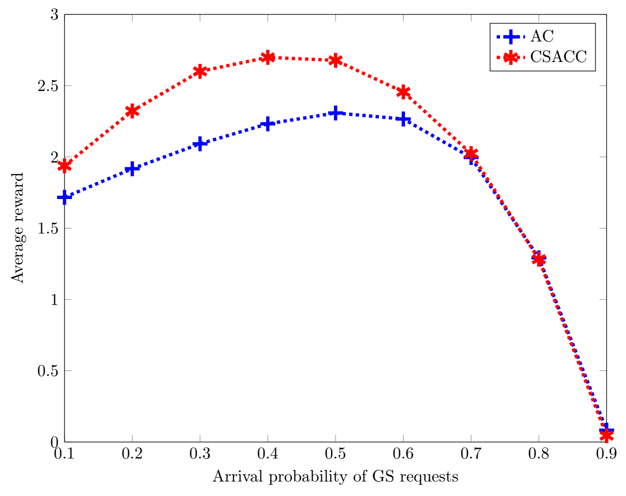
the CSACC in conjunction with the spectral clustering scheme. Simulation results show that in this case, the spectral clustering outperforms the Jaccard similarity by decreasing the dropping probabilities for both BE and GS slice requests. When used with CSACC, the spectral clustering algorithm increases the number of BE slices that can be deployed thus augmenting the amount of resources that can be opportunistically subtracted to the GS slice to the BE slices. This results in reducing the BE requests dropping probability.

The above discussed results highlight the benefits of SA used in conjunction with the proposed CSACC. To assess the feasibility of implementing this framework, we now evaluate the convergence of SARSA with LFA when used for CSACC only and when implemented for CSACC in conjunction with SA. Specifically, simulation results in Fig.10 show the average values of the feature weights as a function of the iterations number. We observe that the proposed LFA scheme allows the feature weights to converge in a reasonable time, for both the scheme, in the order of 5×10^4 iterations.

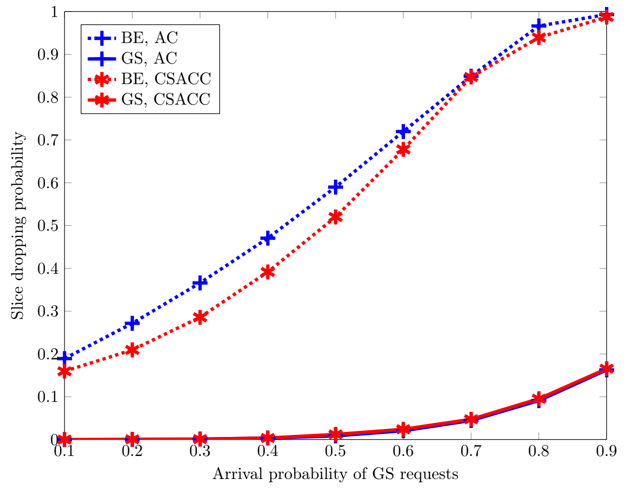
We now conclude our analysis by summarizing the performance achieved by the pillars of our slice deployment and orchestration framework. Fig. 11(a) shows the average reward obtained by our system when using only the cross-slice AC, CSACC, AC with spectral clustering (AC with SA), and CSACC with spectral clustering (CSACC with SA). We can notice that the CSACC leads to a large reward gain as compared to the AC for low values of GS request arrival probabilities; in fact, for these values, the number of BE slices accepted in the system is significant, and a large amount of their resources can be opportunistically re-used to serve GS slice requests. In contrast, the gain brought by the SA increases with the GS request arrival probability, i.e., the larger p^g, the larger the amount of resources that can be shared among the deployed slices. Moreover, we observe that with CSACC with SA, the reward gain with respect to the simple AC is larger than the sum of the gains lead by each single strategy. This gain is due to the large reduction in the BE dropping probability as shown in Fig. 11(b). In fact, the CSACC with SA enables to reduce the GS dropping probability and the BE dropping probability up to 28% and 44%, respectively. Overall, we can conclude that our framework is able, by optimizing the network resource usage, to increase the operator revenue while satisfying the priorities among the different services.

VII. Conclusion

In this paper, we have designed a novel AI framework for network slice deployment and orchestration. Specifically, in the context of the 3GPP NSMF, we have proposed two novel functions for the resource efficient management of slice requests in 5G systems: SA and CSACC. SA assigns each new slice request to an existing NSI, based on the shared NFs, such that the additional amount of resource required to deploy the new slice is minimized. CSACC aims to maximize the system revenue while taking into account slice priorities, network slice requirements, and resource availability. We have formulated this problem using reinforcement learning and designed the on-policy scheme, based on SARSA with LFA, to find the optimal control strategy that maximizes the long term expected reward of the system. Our results show that using SA and CSACC jointly can reduce the GS and BE slice

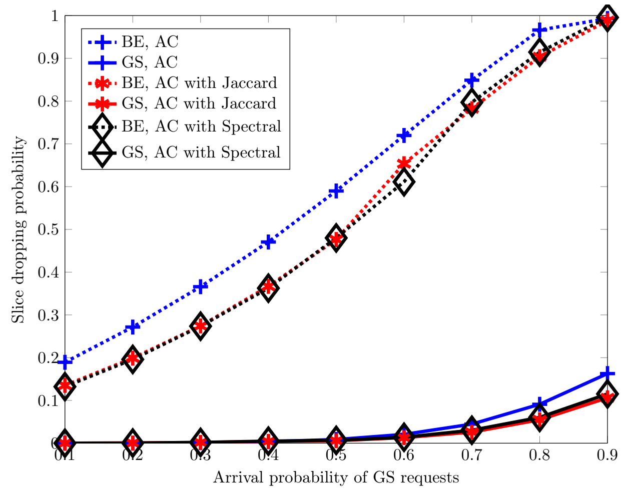


(a)

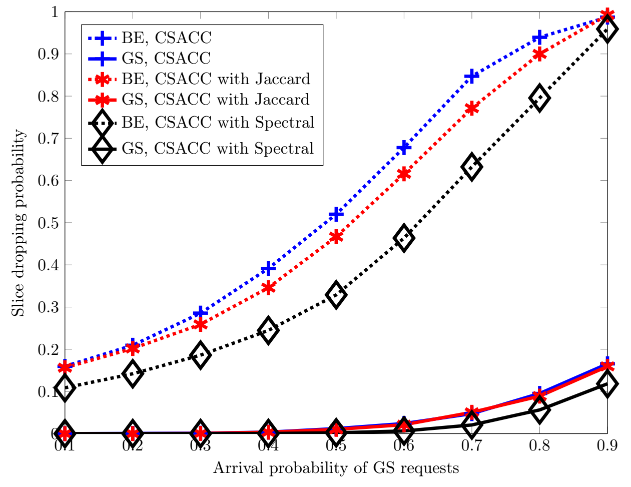


(b)

Figure 8: (a) Average reward and (b) slice dropping probability with AC and CSACC as a function of the GS arrival probability.



(a) Cross-slice admission control



(b) Cross-slice admission and congestion control

Figure 9: Slice dropping probability as a function of the GS request arrival probability with and without slice analytics.

request dropping probability up to 28% and 44%, respectively. In future studies, we will focus on a system dealing with slice requests characterized by more heterogeneous requirements and priorities. Moreover, we will investigate how to size the substrate network and the associated resources, such that the slice dropping probability is further minimized.

References

[1] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. D. Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5G: A Tutorial Overview of Standards, Trials, Challenges, Deployment, and Practice," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1201–1221, Jun. 2017.

[2] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker, "Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 72–79, May 2017.

[3] 3GPP TSG Services and System Aspects, "TR 28.801, Telecommunication management; Study on management and orchestration of network slicing for next generation network," *V15.0.0*, Dec. 2017.

[4] ETSI White Paper No. 22, "Improved operator experience through Experiential Networked Intelligence (ENI)," Oct. 2017.

[5] D. T. Hoang, D. Niyato, P. Wang, A. De Domenico, and E. Calvanese Strinati, "Optimal Cross Slice Orchestration for 5G Mobile Services," *Proc. IEEE VTC Fall*, Aug. 2018.

[6] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis, and X. Costa-Perez, "Optimising 5G infrastructure markets: The business of network slicing," in *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.

[7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[8] P. Caballero, A. Banchs, G. de Veciana, X. Costa-Pérez, and A. Azcorra, "Network Slicing for Guaranteed Rate Services: Admission Control and Resource Allocation Games," *IEEE Transactions on Wireless Communications*, vol. 17, no. 19, pp. 6419–6432, Oct. 2018.

[9] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network slicing in 5G: An auction-based model," in *Proc. IEEE ICC*, May 2017, pp. 1–6.

[10] M. Leconte, G. S. Paschos, P. Mertikopoulos, and U. C. Kozat, "A resource allocation framework for network slicing," in *Proc. IEEE INFOCOM*, May 2018, pp. 2177–2185.

[11] C. Delgado, M. Canales, J. Ortín, J. R. Gállego, A. Redondi, S. Bousnina, and M. Cesana, "Joint Application Admission Control and Network Slicing in Virtual Sensor Networks," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 28–43, Feb. 2018.

[12] A. Bosneag and M. X. Wang, "Intelligent network management mechanisms as a step towards 5G," in *Proc. 8th International Conference on the Network of the Future (NOF)*, Nov. 2017, pp. 52–57.

[13] Y. H. Chen, C. J. Chang, and C. Y. Huang, "Fuzzy Q-Learning Admission Control for WCDMA/WLAN Heterogeneous Networks with

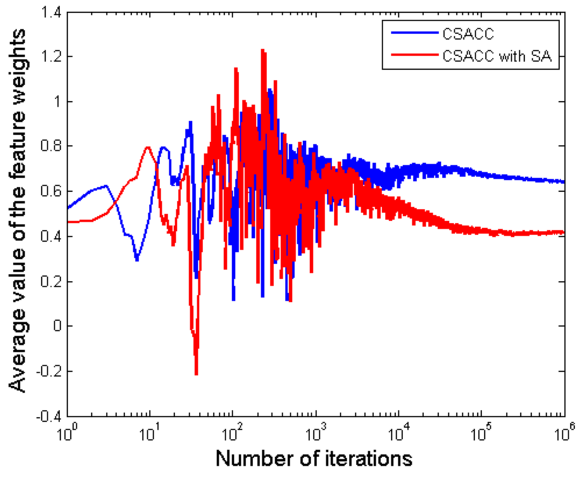
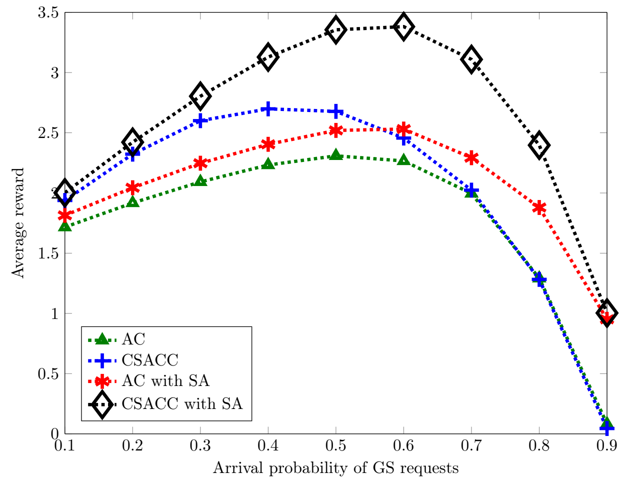
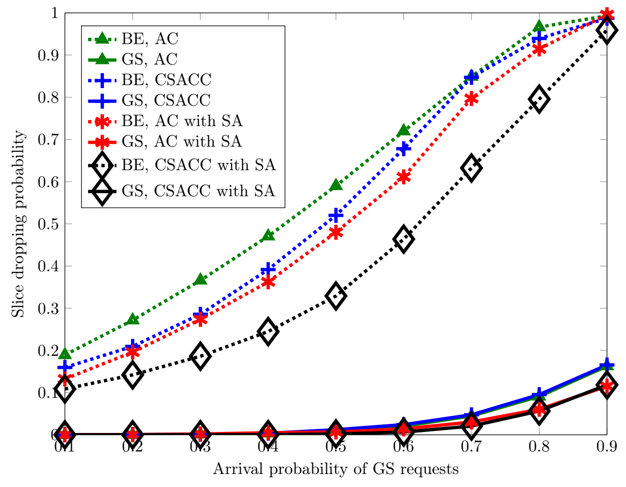


Figure 10: Convergence of feature weights in SARSA with LFA.



(a)



(b)

Figure 11: (a) Average reward and (b) slice dropping probability for different schemes as a function of the GS arrival probability.

Multimedia Traffic," *IEEE Transactions on Mobile Computing*, vol. 8, no. 11, pp. 1469–1479, Nov 2009.

[14] M. Al-Maitah, O. O. Semenova, A. O. Semenov, P. I. Kulakov, and V. Y. Kucheruk, "A Hybrid Approach to Call Admission Control in 5G Networks," *Advances in Fuzzy Systems*, 2018.

[15] F. S. Melo, S. P. Meyn, and M. I. Ribeiro, "An Analysis of Reinforcement Learning with Function Approximation," in *Proc. 25th International Conference on Machine Learning*, July 2008, pp. 664–671.

[16] D. M. Gutierrez-Estevez, M. Gramaglia, A. De Domenico, G. Dandachi, S. K. U. Elzur, and Y. Wang, "Artificial Intelligence for Elastic Management and Orchestration of 5G Networks," *to appear in IEEE Wireless Communications Magazine*.

[17] 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 *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.

[18] NGMN Alliance, "Description of network slicing concept." [Online]. Available: https://www.ngmn.org/

[19] P. Jaccard, "Étude comparative de la distribution florale dans une portion des alpes et des jura," *Bull Soc Vaudoise Sci Nat*, vol. 37, pp. 547–579, 1901.

[20] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in neural information processing systems*, 2002, pp. 849–856.

[21] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[22] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[23] S. Tsironis, M. Sozio, M. Vazirgiannis, and L. Poltechnique, "Accurate spectral clustering for community detection in MapReduce," in *Proc. Advances in Neural Information Processing Systems (NIPS) Workshops*, 2013.

[24] Q. Chen, W. Wang, Y. Wang, P. Zhou, and Z. Zhang, "Content caching clustering based on piecewise interest similarity," in *Proc. IEEE GLOBECOM*, Dec. 2017, pp. 1–6.

[25] P. Hassanzadeh, A. Tulino, J. Llorca, and E. Erkip, "Cache-aided coded multicast for correlated sources," in *Proc. of 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Sept 2016, pp. 360–364.

[26] V. Kuleshov and D. Precup, "Algorithms for multi-armed bandit problems," *CoRR*, vol. abs/1402.6028, 2014. [Online]. Available: http://arxiv.org/abs/1402.6028

[27] M. Geist and O. Pietquin, "Algorithmic Survey of Parametric Value Function Approximation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 6, pp. 845–867, June 2013.

An Artificial Intelligence Framework for Slice Deployment and Orchestration in 5G networks

Ghina Dandachi[1], Antonio De Domenico[1], Dinh Thai Hoang[2], and Dusit Niyato[3]

[1]CEA-LETI, Grenoble, France

[2]School of Electrical and Data Engineering, University of Technology Sydney, Australia

[3]School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore

Abstract

Network slicing is a key enabler to successfully support 5G services with specific requirements and priorities. Due to the diversity of these services, slice deployment and orchestration are essential to guarantee service performance in a cost-effective way. In this paper, we propose an Artificial Intelligence framework for cross-slice admission and congestion control that simultaneously considers communication, computing, and storage resources with the aims of maximizing resources utilization and operator revenue. First, we propose a smart feature extraction solution to analyze the characteristics of incoming requests together with the already deployed slices, and then automatically evaluates the request requirements to make appropriate decisions. Second, we design an online algorithm that controls the slice admission based on their priorities, the arrival and departure characteristics, and the available resources. To mitigate system overloading, our framework dynamically adjusts resources allocated to low priority slices, thereby reducing the dropping probability of new slice requests. The proposed algorithm offers outstanding advantages over traditional static approaches by automatically adapting the controller decisions to the system changes. Simulation results show that our framework significantly improves the resource utilization and reduces the slice request dropping probabilities up to 44% as compared to the baseline schemes.

I. Introduction

The technological revolution for the last decade has been changing the way we communicate by introducing emerging applications and services. Massive machine type communications (mMTC), autonomous vehicles, smart factories, and virtual reality are changing the network requirements in terms of number of connections per user, traffic volume, and end-to-end latency [1]. To efficiently support use cases and applications with heterogeneous requirements, the fifth generation (5G) communication systems will deploy a novel and flexible architecture where the network infrastructure is logically split into different instances, i.e., network slices, each designed for a specific service and running in the cloud environment [2]. A network slice is composed of physical and virtual network functions (NFs), where a virtual NF represents the software implementation of the traditional functions, such as routing or packet scheduling.

In this context, depending on the network load and service requirements, the 5G systems will need to manage network resources smartly according to different radio, transport, and cloud domains. Moreover, it is necessary to take into account that different slice requests may have diverse priorities and constraints, and the network resources have to be managed accordingly. Therefore, 3GPP has been developing network slice management and orchestration solutions based on the European Telecommunications Standards Institute (ETSI) Management and Orchestration and Network Function Virtualization frameworks [3]. In addition, the ETSI Experiential Network Intelligence (ENI) group has been investigating Artificial Intelligence (AI) to achieve autonomous, and thus cost-effective, slice management and orchestration in future communication networks [4]. However, these frameworks define only brief guidelines on architectural aspects (interfaces and requirements) and design principles without providing specific solutions. Accordingly, there is an urgent need for new schemes and frameworks able to provide cross-slice resource orchestration and management. In this paper, we focus on AI-based cross-slice admission and congestion control with the goal of maximizing the operator revenue by improving the network resource utilization.

A. Related Work

In the literature, the admission control problem for 5G network slicing has been investigated under two main directions: by using Markov Decision Process (MDP) and game theory. Specifically, Hoang et al. characterized the slice admission control problem as an MDP problem that can be solved by using the value iteration algorithm, which is based on dynamic programming [5]. However, this approach requires a minimum knowledge of service model characteristics (e.g., the slice request arrival and departure rates) and can only find the optimal solution for problems with a limited number of states. Bega et al. [6] introduced the concepts of elastic and inelastic slices, modeled the slice admission control as an MDP, and implemented the Q-learning algorithm to maximize the operator revenue. ~~However, the Q-learning algorithm suffers from the so-called “curse of dimensionality” [7], meaning that its computational requirements~~

~~grow exponentially with the number of state variables in the problem. Accordingly, this approach cannot be efficiently implemented in realistic scenarios.~~

Caballero et al. [8] proposed a game theoretic approach for the slice admission control problem to enable fair resource partitioning between deployed slices. ~~However, this study only focuses on radio resources and does not consider priorities among different slice requests.~~ Jiang et al. [9] investigated the virtual resource allocation for network slicing by using auction mechanisms in order to maximize the network revenue under different slice priorities. ~~However, this study does not consider admission control and is unable to limit the slice request dropping probability.~~ Leconte et al. [10] proposed an elastic framework for cross-slice resource allocation that achieves a Pareto-efficient solution ensuring fair radio and computing resource allocation. ~~However, the obtained solution considers only instantaneous optimization and does not take into account the priorities between slices.~~ Delgado et al. [11] studied a greedy algorithm for slice resource allocation in wireless sensor networks. Four resource types are considered, i.e., radio, storage, computing and energy, with the energy resources being the most important one. ~~Nevertheless, the work in [11] did not consider priorities among different slices. In addition, game theory-based solutions cannot deal with system dynamics, and thus their outcome strategies are not able to provide long-term optimization. Moreover, none of the above works propose a solution for resource shortage problem in the context of slice deployment and orchestration.~~

Early works on AI for mobile networks highlighted the importance of introducing intelligence and automation in admission control function [12]. Chen et al. [13] studied admission control using fuzzy Q-learning to achieve a lower blockage rate in WCDMA/WLAN heterogeneous networks. Al-Maitah et al. [14] employed a genetic neurofuzzy controller to improve the quality of call admission in mobile networks. Although these methods proposed AI solutions, to the best of our knowledge, there is a lack of AI-based framework that can simultaneously address the admission and congestion control problems for 5G slice management and orchestration.

B. Contributions and Organization

The key contributions of this paper can be summarized as follows:

1) We propose a new architecture for joint cross-slice admission control and congestion control to maximize the operator revenue while considering different slice types and resource requirements. Accordingly, we introduce three new functions: Slice Analytics (SA), Admission Control (AC), and Congestion Control (CC). This novel architecture is

fully compatible with the Network Slice Management layer proposed by 3GPP [3]. By adopting it, we jointly improve the usage of communication, computing, and cloud storage resources while guaranteeing the slice priorities.

2) The proposed SA function attempts to assign a new slice request to an existing Network Slice Instance (NSI) based on the common NFs to maximize the resource sharing across slices and accelerate the slice deployment process. In particular, we propose two different algorithms that can implement this functionality: Jaccard similarity-based assignment and spectral clustering-based assignment. The first scheme assigns each new slice request to the NSI that maximizes the Jaccard similarity. Alternatively, the spectral clustering approach tries to iteratively optimize the resource sharing by clustering all slices into new NSIs at each slice arrival. Our simulation results show that the proposed SA function can significantly reduce the resource footprint for each accepted slice.
3) The proposed cross-slice AC aims to optimize the trade-off between resource utilization and dropping probability of slice with high priorities. We then introduce the concept of cross-slice AC based on AI, where the controller is able to deal with system dynamics and its decision enables a long-term optimal solution in terms of the slice resource management. To achieve this goal, we design an AI scheme based on the State-Action-Reward-State-Action (SARSA) scheme [15] with linear function approximation (LFA), which is a model-free reinforcement learning approach that can deal with high dimensional and complex problems.
4) We also introduce a CC function that limits the dropping of slice requests, especially when the system becomes overloaded. The CC can scale down resources allocated to slices with low priorities, so that slices with high priorities can be admitted. This function is tightly coupled with the AC, as it determines the resources available for accepting new slice requests, while the amount of resources that can be scaled down is the results of the AC action. Therefore, we develop a joint implementation of both controllers using SARSA with LFA, which we denote as cross-slice admission and congestion control (CSACC).

The rest of the paper is organized as follows. The proposed architecture and the associated system model is presented in Section II. In Section III, we define the SA function and the proposed algorithms. In Section IV, we introduce the mathematical model of the designed CSACC, and then we present the SARSA algorithm with LFA in Section V. Simulation results

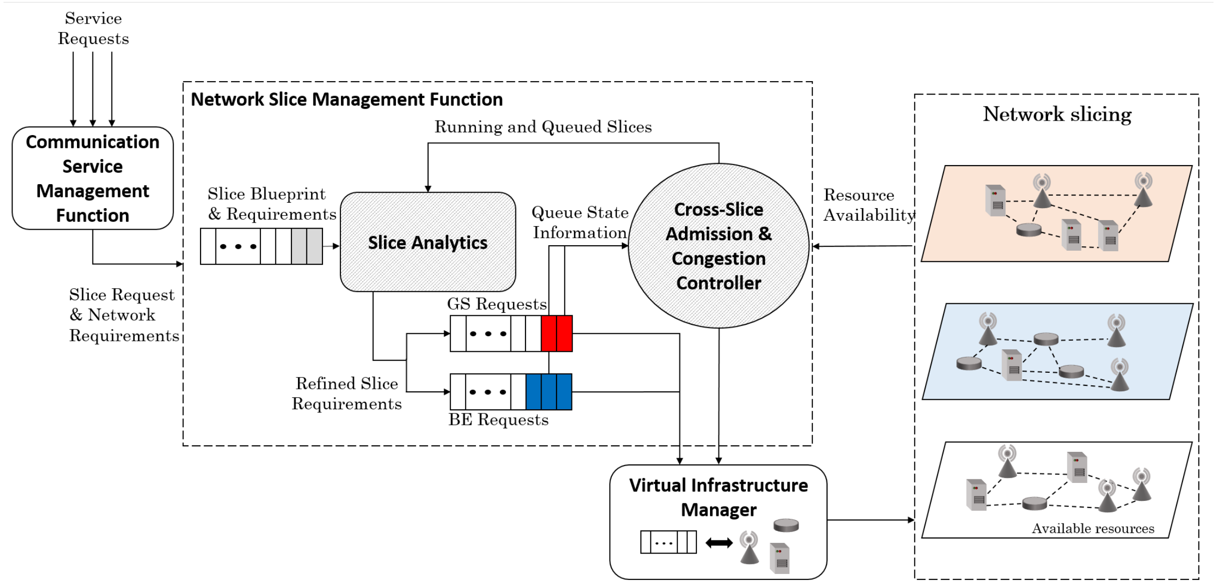

Service Requests
Communication Service Management Function
Slice Request & Network Requirements
Network Slice Management Function
Running and Queued Slices
Slice Blueprint & Requirements
Slice Analytics
Queue State Information
Cross-Slice Admission & Congestion Controller
GS Requests
Refined Slice Requirements
BE Requests
Virtual Infrastructure Manager
Resource Availability
Network slicing
Available resources

Figure 1: The proposed framework for network slice deployment and orchestration.

are provided in Section VI. Finally, the conclusions and future directions are discussed in Section VII.

II. System Model

We consider a system supporting network slicing to satisfy the diverse 5G service requirements ~~of the services related to the 5G ecosystem~~. In this network, a slice management framework (described in Fig. 1) is used to instantiate network slices and orchestrate the ~~network~~ resources across the accepted slices by exploiting the so called *resource elasticity* paradigm [16]. ~~To design an efficient, multi-service, multi-slice, and multi-tenant architecture, we take advantage of the so called *resource elasticity* paradigm.~~ In a nutshell, with resource elasticity, the network is able to ~~the proposed slice management framework~~ gracefully adapts ~~the network~~ its configuration to system changes ~~in an automatic manner~~ through AI, such that ~~at each point in time~~ the available resources match the service demand as closely and efficiently as possible. ~~Although elasticity in mobile networks has traditionally been exploited in the context of communications resources (e.g., when a base station gracefully downgrades the spectral efficiency of a given communication link), here~~ Here, we apply this concept to communication, computational, and storage resources jointly. More specifically, we focus on elastic slices that enable a certain level of flexibility in the resource orchestration: we consider guaranteed quality-of-service (GS) slices that do not require fully dedicated network resources (i.e., resource isolation) and best effort (BE) slices that, in addition, allow the system to temporarily reduce their resources in order to deal with GS slices, which have higher priorities and tighter requirements.

In the 3GPP specification [3], the Communication Service Management Function (CSMF) receives requests for new services and transforms the consumer-facing service descriptions into slice-related network requirements such as connection density, end-to-end latency, and coverage. Then, the CSMF sends ~~the network slice requests together with the associated requirements~~ this information to the Network Slice Management Function (NSMF), which derives accordingly the so-called network slice blueprint [3]. The network slice blueprint is a description of the network slice in terms of required NFs, their interconnection and configuration ~~according to the specific service request~~. The main role of the NSMF is to manage and orchestrate the overall network slice life-cycle.

~~More specifically, in~~ In this work, we study how, based on the slice blueprint, the NSMF identifies the resource requirements for each slice and decides whether and when to instantiate the slice request. Then, the NSMF will distribute the available resources across the activated slices and try to maximize the resource utilization, i.e., the number of slices that can be handled at the same time. However, maximizing the resource utilization may prevent the system from accepting new slice requests with high priorities, which provide high revenue for the operator, due to the potential scarcity of available resources. Hence, this procedure needs to be carefully managed to optimize the trade-off between resource utilization and operator revenue.

To optimize such trade-off, we investigate two main functionalities represented by the grey blocks in Fig. 1:

- The SA ~~block~~ receives the slice blueprint and resource requirements related to the accepted and queued slice requests. It refines the amount of resources to be allotted to each network slice by analyzing the similarities in the slice requirements. Additionally, ~~this block~~ it classifies the slice requests into two types, GS and BE slices, according to the associated requirements[1].
- The CSACC ~~is a resource orchestrator that~~ performs jointly cross-slice AC and CC functions. ~~This block~~ It monitors the requests of the queued slices and the available resources, and

[1]This model can be straightforwardly extended to a more general classification that may include classical 5G use cases such as eMBB, URLLC, and mMTC.

manages the slice deployment and the resource allocation accordingly.

Finally, in the proposed framework, the role of the virtual infrastructure manager is to execute the instructions received from the CSACC.

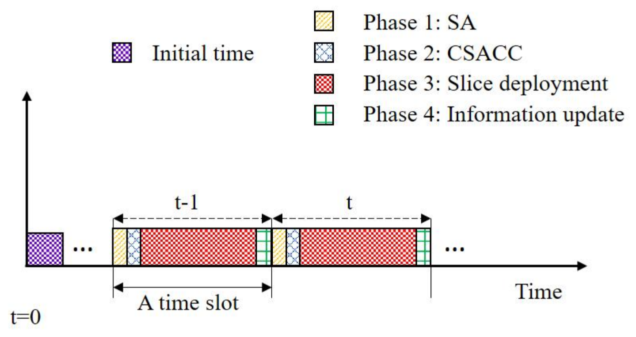


Figure 2: Decision making cycle for network slice deployment and orchestration.

According to the above system operations, we consider a decision-making cycle that can be divided into four phases as illustrated in Fig. 2. At the beginning of each time slot, the SA phase evaluates the effective resource requests based on the similarities between the queued network slices and the active network slices. Then, in the CSACC phase, based on the current states of the GS and BE queues, and the momentary available resources of the system, the CSACC first decides whether to scale down the resources allocated to the BE slices, and then selects the slice requests to admit. Then, in the third phase, i.e., the slice deployment phase, the virtual infrastructure manager allocates the network resources according to the CSACC decisions, and the accepted slices are instantiated. Finally, in the last phase, i.e., the information updating phase, new requests arrive, and they are processed by the CSMF, and the input of the SA is updated accordingly. The duration of the above discussed cycle should be short enough to monitor the variation in the slice resource requests and enable the NSMF functions to react accordingly. For the slice admission control period, a duration in the order of one second was considered in [17]. The main notations used in this paper are described in Table I.

A. Blueprint and Slice Resource Requirements

The NGMN Alliance defines the slice blueprint as a complete description of the structure, configuration, and the work flows to instantiate and control a network slice during its life-cycle [18]. In this work, we characterize a slice blueprint as follows:

Table I: Main Notations and System Parameters.

Notation	**Parameter**
$\mathcal{S} \triangleq \mathcal{S}_g \times \mathcal{S}_b \times \mathcal{S}_p \times \mathcal{U}_g \times \mathcal{U}_b \times \mathcal{X}_p$	State space
$\mathbf{s}(t) \in \mathcal{S}$	System state
$s_g(t) \in \mathcal{S}_g, s_b(t) \in \mathcal{S}_b$	Slice requests in GS and BE queues
$\boldsymbol{s_p}(t) = (r, c, m) \in \mathcal{S}_p$	Communication [MHz], computing [GFLOPS/s], and storage [GB] resources
$u_g(t) \in \mathcal{U}_g, u_b(t) \in \mathcal{U}_b$	Deployed GS and BE slices
$\boldsymbol{x_p}(t) = (x_r, x_c, x_m) \in \mathcal{X}_p$	Allocated resources for deployed BE slices
$\boldsymbol{a}(t) = (a_g, a_b) \in \mathcal{A}$	Admission and Congestion Control action and action space
$a_g(t), a_b(t)$	Number of accepted GS and BE requests
$\Lambda_{n,n'}$	Jaccard similarity between slices n and n'
$\mathcal{B}$	Deployed Network Slice Instances
$d_{n,k,j}$	Resources of type j required by the k^{th} NF in the n^{th} request
$d_{n,j}$, $d'_{n,j}$	Resources of type j required by n^{th} slice before, after slice analytics
w_g, w_b	Rewards for each accepted GS and BE slices
$n_d(t), l_d$	Instantaneous dropped GS slices and dropping loss
$n_b(t)$, $n_g(t)$	New BE and GS requests
$R(\mathbf{s}(t), \mathbf{a}(t))$	Instantaneous reward
$Q(\mathbf{s}, \mathbf{a})$	Q-value of a state-action pair
α, γ	Learning rate and discount factor
$\boldsymbol{\phi}(\mathbf{s}, \mathbf{a})$	Feature vector
θ_i	Weight of the i^{th} feature

- A list of NFs that the slice requires and the description of the interactions among the NFs.
- The parameters that describe the configuration of each NF.

Depending on the type of service, the system will deploy a specific version of each NF, which enables to satisfy the service requirements. For example, in the case of eMBB service, the NF providing 5G PHY layer is characterized by precise configuration parameters as large bandwidth, modulation and coding schemes with high spectral efficiency, etc. The configuration of a NF, in turn, defines the amount of resources that it requires to be deployed in the 5G system. We consider that a 5G network slice can be composed by three sets of NFs: radio $\mathcal{K}^{\mathrm{R}}$, transport $\mathcal{K}^{\mathrm{T}}$, and cloud $\mathcal{K}^{\mathrm{C}}$ (see Fig. 3), each consisting of a finite number of NFs requiring three types of resources:

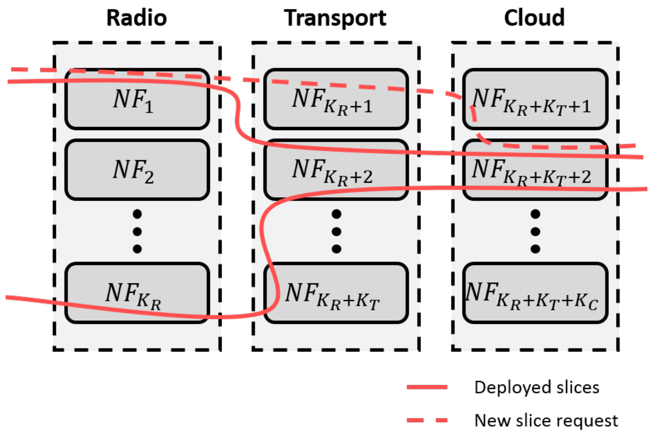


Figure 3: Mapping of network slices to the required network functions.

communication (in terms of bandwidth [MHz]), computing (in [GFLOPS/s]), and cloud storage (in [GB]). For example, typical NFs in the radio, transport, and cloud sets are the physical layer, routing, and caching functions, respectively. We use $\mathcal{K} = \mathcal{K}^{\mathrm{R}} \cup \mathcal{K}^{\mathrm{T}} \cup \mathcal{K}^{\mathrm{C}} = \{NF_1, \ldots, NF_K\}$ to denote the set of all $K = (K_R + K_T + K_C)$ NFs that can be deployed in the 5G system. Let $N \geq 0$ be the number of slice requests at the NSMF. We suppose that the slice blueprint of the n^{th} slice request includes a set of NFs, denoted by $\mathcal{D}_n \subseteq \mathcal{K}$, and a set of configuration parameters for each $NF_k \in \mathcal{D}_n$, indicated by $\mathcal{L}_{n,k}$. Then, to indicate which NFs compose a request $\mathcal{D}_n$, we use the vector $\boldsymbol{\lambda_n} \in \{0,1\}^K$, whose k^{th} entry is defined as:

$$\lambda_{n,k} = \begin{cases} 1 & \text{if } NF_k \in \mathcal{D}_n, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

For each NF $NF_k \in \mathcal{D}_n$, we represent the associated configuration parameters as a set of J binary vectors[2] as follows:

$$\mathcal{L}_{n,k} = \{\boldsymbol{l}_{n,k,1}, \boldsymbol{l}_{n,k,2}, \ldots, \boldsymbol{l}_{n,k,J}\}. \tag{2}$$

Then, we map the configuration parameters with the amount of communication ($d_{n,k,r}$), computing ($d_{n,k,c}$), and cloud storage ($d_{n,k,m}$) resources required by the NF NF_k of the n^{th} slice

[2]For the ease of presentation, we consider that all NFs have the same number of configuration parameters.

request using a model composed of a static part and a dynamic part:

$$\begin{aligned} d_{n,k,r} =& \rho_k + f_{k,r}(\mathcal{L}_{n,k}), \\ d_{n,k,c} =& \chi_k + f_{k,c}(\mathcal{L}_{n,k}), \\ d_{n,k,m} =& \mu_k + f_{k,m}(\mathcal{L}_{n,k}), \end{aligned} \tag{3}$$

where $\rho_k, \chi_k,$ and μ_k are the minimum amount of required resources for activating a given NF, $NF_k \in \mathcal{D}_n$, and $f_{k,r}(\mathcal{L}_{n,k})$, $f_{k,c}(\mathcal{L}_{n,k})$, and $f_{k,m}(\mathcal{L}_{n,k})$ are the additional amount of required resources that can be defined as a function of the configuration parameters $\mathcal{L}_{n,k}$ characterized for the NF_k. In summary, the total communication, computing, and cloud storage resource demand of the n^{th} slice request can be computed as follows:

$$\boldsymbol{T}_n = \left(d_{n,r}, d_{n,c}, d_{n,m}\right), \tag{4}$$

where $d_{n,j} = \sum_{NF_k \in \mathcal{D}_n} d_{n,k,j}, j \in \{r, c, m\}$.

B. State Space of the Proposed Slice Deployment and Orchestration Framework

Here we describe the state space of the proposed framework for the network slice deployment and orchestration. ~~In the following sections, we describe the proposed functions to optimize the network resource utilization, while considering the priorities of the different service requests.~~ The state space of the system, denoted by $\mathcal{S}$, includes the sets that describe the states of the GS queue, the BE queue, the available network resources, the number of deployed GS slices, the number of deployed BE slices, and the resources allocated to the BE slices. Accordingly, $\mathcal{S}$ is defined as follows:

$$\mathcal{S} \triangleq \mathcal{S}_g \times \mathcal{S}_b \times \mathcal{S}_p \times \mathcal{U}_g \times \mathcal{U}_b \times \mathcal{X}_p. \tag{5}$$

At each time slot of the proposed cycle for slice deployment and orchestration (see Fig. 2), the numbers of slice requests in the GS and BE queues are denoted respectively by $s_g \in \mathcal{S}_g = \{0, 1, \ldots, Q_g\}$ and $s_b \in \mathcal{S}_b = \{0, 1, \ldots, Q_b\}$, where Q_g and Q_b are the maximum length of the GS and BE queues, respectively. As for the network resource state, let $\boldsymbol{s_p} = (r, c, m) \in \mathcal{S}_p$, where $r \in \{0, 1, \ldots, R\}$, $c \in \{0, 1, \ldots, C\}$, and $m \in \{0, 1, \ldots, M\}$, denote the available communication, computing, and cloud storage network resources, respectively. We also indicate the number of deployed GS slices, deployed BE slices, and the network resources associated to the running BE slices respectively as $u_g \in \mathcal{U}_g = \{0, 1, \ldots, U_g\}$, $u_b \in \mathcal{U}_b = \{0, 1, \ldots, U_b\}$

and $\boldsymbol{x_p} = (x_r, x_c, x_m) \in \mathcal{X}_p \subseteq \mathcal{S}_p$, respectively, where U_g and U_b are the maximum numbers of GS and BE slices, which can be simultaneously accepted by the system. Note that, in our optimization framework, we scale down only the resources allocated to BE slices, in order to increase the number of accepted slices. Therefore, we do not need to define a set related to the resources allocated to the GS slices.

Then, we use $n_g \in \mathcal{N}_g = \{0, 1, \ldots, N_g\}$ and $n_b \in \mathcal{N}_b = \{0, 1, \ldots, N_b\}$ to denote the number of new GS and BE slice requests at each time slot, where N_g and N_b are the maximum numbers of arriving requests for the GS and BE services, respectively. Let p_n^g and p_n^b be the probabilities of arriving n_g and n_b slice requests at a given time slot; accordingly, we have:

$$\sum_{n=0}^{N_g} p_n^g = 1 \qquad \text{and} \qquad \sum_{n=0}^{N_b} p_n^b = 1. \tag{6}$$

It is important to note that, a slice request is dropped when it arrives in a queue that is full[3]. Finally, we denote f^g and f^b as the departure probabilities of an accepted GS slice and an accepted BE slice at a given time slot, respectively. In addition, when a given slice leaves the system, all its allocated resources are immediately released.

III. Slice Analytics Procedure

As mentioned in Section II, one of the role of the NSMF is to elaborate and update the slice blueprint, which clearly identifies the NFs required by each network slice in the 5G system (see Fig. 3). In the proposed framework, we classify each required NF either as a dedicated NF or as a shared NF with respect to the other NFs required by the other (queued or actively deployed) network slices. Accordingly, the resource usage of the 5G system can be optimized if multiple NFs shared across different slices can (even partially) share common network resources at the same time. For instance, two slices deployed in the same geographical area can share the same base stations if the available capacity is large enough to satisfy the overall demands. Moreover, during a sport event, a single operator can use network slicing to accommodate the wireless services required by different broadcasters. In this case, a large part of the communication resources can be shared by these slices to broadcast the common content (e.g., the video) to different users, while specific content (e.g., the speaker's voice) can use dedicated resources. From the operator's point of view, this approach enables to increase the number of accepted

[3]The queues capacity is limited to avoid long delay for the slice requests waiting in the queues.

slice requests (generating higher revenue), decrease the slice deployment cost, and limit the service creation time.

Using the 3GPP terminology, we denote the set of NF instances and the associated network resources deployed to satisfy one or multiple slice requests as a Network Slice Instance (NSI). An NSI is composed of NFs shared between two or more slices, as well as dedicated NFs, as shown in Fig. 4. To efficiently map a network slice request to an NSI, we propose a procedure, namely SA, that analyzes the network slice blueprints, in order to evaluate the *similarities* among network slices and identify the NSI that can serve the new slice request with a minimum amount of additional network resources (see Algorithm 1).

In SA, we use $NSqueue(n)$ and $\mathcal{B}$ to denote the n^{th} slice request and the set of deployed NSIs. First, SA compares the NFs required by the n^{th} slice request $\boldsymbol{\lambda}_n$ with those associated with each running NSI, by calculating the Jaccard similarity as follows [19]:

$$\Lambda_{i,n} = \frac{\boldsymbol{\lambda}_i \boldsymbol{\lambda}_n}{\|\boldsymbol{\lambda}_i\| + \|\boldsymbol{\lambda}_n\| - \boldsymbol{\lambda}_i \boldsymbol{\lambda}_n}, \; i \in \mathcal{B}, \tag{7}$$

where $\boldsymbol{\lambda}_i$ and $\boldsymbol{\lambda}_n$ are defined in (1). Based on the computed similarities, SA chooses an NSI $i^* \in \mathcal{B}$ according to one of the association algorithms presented in Section III-A. If the selected NSI can integrate the slice request, e.g., there are no resource isolation constraints, the slice analytics computes the degree of similarity for each NF required by the $NSqueue(n)$ and shared with the other slices associated to the selected NSI i^* (see (13)), and accordingly evaluates the additional amount of resources required by the NSI to accommodate the slice request n, $\boldsymbol{T}'_n$ (15). If the slice request cannot be assigned to any of the existing NSIs, a new NSI will be prepared. If the request is accepted by the CSACC, presented in Section IV, either the new NSI is instantiated or the NSI i^* is updated. Finally, the virtual infrastructure manager (see Fig. 1) executes the resource allocation procedure.

A. Similarity-Based Slice Association

The first step in SA is to assign a new slice request to an existing NSI. This assignment is based on one of the two proposed strategies:

- The SA assigns a new slice demand to the existing NSI which maximizes the Jaccard similarity. The advantage of this solution is the low complexity, and may be implemented in a fast time-scale, in the order of one second.

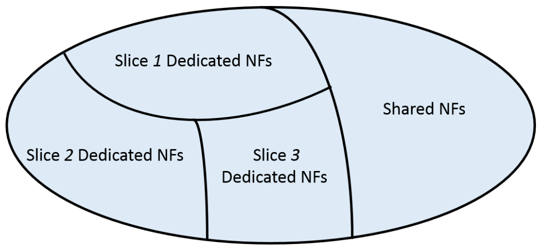


Figure 4: An illustration of shared and dedicated NFs in a network slice instance. A slice is composed of dedicated NFs and shared NFs with other slices.

Algorithm 1 Slice Analytics

```
Input: $NSI = \{NSI\_id, \boldsymbol{\lambda}, \boldsymbol{T}\} \leftarrow$ deployed NSIs;
    $NSqueue = \{NSI\_id, \boldsymbol{\lambda}, \boldsymbol{T}\} \leftarrow$ queued NSs;
    $NSassociated = \{NSI\_id, \boldsymbol{\lambda}, \mathcal{L}, \boldsymbol{T}\} \leftarrow$ NSs associated to deployed NSIs;
Output: Updated $NSassociated$.
 1: for $n = 1$:Length(NSqueue) do
 2:     Calculate Jaccard similarity (7) between $NSqueue(n)$ and each NSI $i \in \mathcal{B}$.
 3:     Use the selected association algorithm to identify the proper NSI $i^* \in \mathcal{B}$.
 4:     if NSI $i^*$ can integrate $NSqueue(n)$ then
 5:         Calculate cosine similarity (12) between $NSqueue(n)$ and $NSassociated$ to $i^*$.
 6:         Evaluate the new $\boldsymbol{T'}$ of $NSI(i^*)$ (15).
 7:         if $\boldsymbol{T'}$ of $NSI(i^*)$ can be satisfied (see CSACC) then
 8:             Update $NSI\_id$ of $NSqueue(n)$, $\boldsymbol{T'}$ of $NSI(i^*)$, and $NSassociated$ to $i^*$.
 9:         end if
10:         if $NSI\_id$ of $NSqueue(n)$ is empty then
11:             if $\boldsymbol{T}$ of $NSqueue(n)$ can be satisfied (see CSACC) then
12:                 Create the new $NSI$ $n$ and add $n$ to $\mathcal{B}$.
13:                 Set $\boldsymbol{T}$ of $NSI(n)$, $\boldsymbol{\lambda}$ of $NSI(n)$, and $NSI\_id$ of $NSqueue(n)$.
14:             end if
15:         end if
16:     end if
17: end for
```

- The SA uses spectral clustering to create new NSIs which maximize the resource sharing in the 5G system. This is an iterative solution, and thus more complex than the fast time-scale approach. Accordingly, it may be implemented at a slower time-scale, e.g., each 60 seconds.

1) Jaccard similarity-based assignment: With the first proposed strategy, the selected NSI is the one that shares the highest number of NFs with the slice request. Accordingly, given a set of deployed NSI $\mathcal{B}$, the SA attempts to assign the slice request j to the NSI i^* such that

$$i^* = \underset{i \in \mathcal{B}}{\text{argmax}} \; \Lambda_{ji}. \tag{8}$$

2) Spectral clustering-based assignment: To further optimize the resource usage and increase the number of accepted slice requests, the second proposed strategy periodically re-clusters the running slices in new NSIs. In order to achieve this goal, we implement a normalized spectral clustering algorithm [20] based on the Jaccard similarity among running network slices. In the spectral clustering, the deployed slices are represented as nodes of a connected graph and clusters are found by partitioning this graph based on their spectral decomposition into subgraphs. This clustering scheme is simple to implement and typically outperforms other clustering algorithms such as the K-means algorithm [21].

Let $\mathcal{B} = \{1, \dots, N_R\}$ denote the set of running slices with NF requests $\{\lambda_1, \dots, \lambda_{N_R}\}$; we compute the affinity matrix $\mathbb{A}$, whose elements are the Jaccard similarity between two running slices as follows:

$$\mathbb{A}_{n,n'} = \begin{cases} \Lambda_{n,n'} & \text{if } n \neq n', \\ 0 & \text{otherwise,} \end{cases} \quad n, n' \in \mathcal{B}, \tag{9}$$

where $\Lambda_{nn'}$ is computed as in (7). Then, we derive from $\mathbb{A}$ the corresponding diagonal matrix $\mathbb{D}$, whose (n, n) element is the sum of the n^{th} row of $\mathbb{A}$, i.e., $d_{nn} = \sum\limits_{n' \in \mathcal{B} \setminus n} \Lambda_{n,n'}$ and the associated normalized laplacian matrix as:

$$\mathbb{L} = \mathbb{D}^{-1/2} \mathbb{A} \mathbb{D}^{-1/2}. \tag{10}$$

Afterwards, SA computes the normalized eigenvectors of $\mathbb{L}$ [20] and clusters the first k eigenvectors using K-means [22]. The number of clusters k is obtained such that all eigenvalues $(1, \dots, k)$ are very small, but the $k^{th} + 1$ is relatively large [21].

By deploying new NSIs that maximize the similarity within their slices, this strategy aims to achieve an improved performance as compared to the previously discussed Jaccard similarity-based assignment. However, the spectral clustering increases the complexity of SA to $\mathcal{O}(N_R^3)$

[23] as compared to $\mathcal{O}(N_R)$, which characterizes the Jaccard similarity-based assignment. When considering the expected number of slices in the 5G network, the spectral cluster performance is not limited by its complexity; however, this approach should be implemented with a limited frequency or only when the system is overloaded.

B. Intra-NSI Similarity for Resource Mutualization

Both the SA strategies proposed in Section III-A allow to identify an already existing NSI, which is appropriate to support a new communication service. However, to satisfy the specific requirements of the new service request, the deployed NSI may require additional network resources. Thus, before finalizing this process, the new NSI network requirements must be evaluated and the associated resource request must be accepted by the CSACC (see Section IV). In this work, we assume that the amount of resources that can be mutualized by the existing NSI and the new service request depends on the shared NFs and their associated configuration parameters. Specifically, we use $\mathcal{B}_{i,k}$ to denote the set of slices included in NSI $i \in \mathcal{B}$ and requiring NF $k \in \mathcal{K}$; then, for each pair of slices $n,\ n' \in \mathcal{B}_{i,k}$, we define the similarity between the sets of configuration parameters $\mathcal{L}_{n,k}$ and $\mathcal{L}_{n',k}$ (see (2)) as follows:

$$C(\mathcal{L}_{n,k}, \mathcal{L}_{n',k}) = \sum_{j=1}^{J} h(\boldsymbol{l}_{n,k,j}, \boldsymbol{l}_{n',k,j}), \tag{11}$$

where J is the number of parameters of the NF k. Moreover, h computes the cosine similarity[4] between two values of the j^{th} parameter of NF k, as follows:

$$h(\boldsymbol{l}_{n,k,j}, \boldsymbol{l}_{n',k,j}) = \frac{\boldsymbol{l}_{n,k,j}\boldsymbol{l}_{n',k,j}}{||\boldsymbol{l}_{n,k,j}||||\boldsymbol{l}_{n',k,j}||}, \tag{12}$$

where $||\cdot||$ is the euclidean norm operator. Considering all network slices $n' \neq n \in \mathcal{B}_{i,k}$, we define the largest similarity for slice request n with respect to NF k as follows:

$$\sigma^*_{n,i,k} = \max_{\forall n' \neq n \in \mathcal{B}_{i,k}} \{C(\mathcal{L}_{n,k}, \mathcal{L}_{n',k})\}, i \in \mathcal{B}. \tag{13}$$

[4]Cosine similarity may not be suitable for all NFs, e.g., for measuring the similarity among contents cached in mobile edge clouds, other schemes can be adopted, for example, from [24], [25].

Then, the communication, computing, and cloud storage resources required for NF k when including the slice request n as part of the NSI i can be computed respectively as follows:

$$\begin{aligned} d'_{n,k,r} =& \rho_k \beta(\sigma^*_{n,i,k}) + (1-\sigma^*_{n,i,k}) f_{k,r}(\mathcal{L}_{n,k}), \\ d'_{n,k,c} =& \chi_k \beta(\sigma^*_{n,i,k}) + (1-\sigma^*_{n,i,k}) f_{k,c}(\mathcal{L}_{n,k}), \\ d'_{n,k,m} =& \mu_k \beta(\sigma^*_{n,i,k}) + (1-\sigma^*_{n,i,k}) f_{k,m}(\mathcal{L}_{n,k}). \end{aligned} \tag{14}$$

We recall that ρ_k, χ_k, and μ_k are the minimum amount of required resources for activating a given NF, and $f_{k,r}(\mathcal{L}_{n,k})$, $f_{k,c}(\mathcal{L}_{n,k})$, and $f_{k,m}(\mathcal{L}_{n,k})$ denote the amount of resources required by the NF k as a function of its configuration parameters $\mathcal{L}_{n,k}$ (see (3)). Moreover, $\beta(\cdot)$ is a step function that is equal to one if its argument is positive and zero otherwise, i.e., the static part of the resource requirements of NF k is not needed if $\sigma^*_{n,i,k} > 0$. To conclude, the refined resource demands of a network slice request n after the SA procedure can be computed as follows:

$$\boldsymbol{T'_n} = \left(d'_{n,r}, d'_{n,c}, d'_{n,m}\right), \tag{15}$$

where $d'_{n,j} = \sum_{NF_k \in \mathcal{D}_n} d'_{n,k,j}, j \in \{r, c, m\}$.

IV. Cross-Slice Admission and Congestion Controller

In this section, we propose a reinforcement learning framework that optimizes the trade-off between the number of accepted slices and the dropping probability of the network slice requests with high priorities, which in turns affects the long-term system revenue. This is a challenging problem as the slice request arrival and departure probabilities are unknown and the network resources are limited (see Section II-B). Thus, minimizing the dropping probability of high-priority slice requests requires to limit the acceptance of low-priority slice requests, thereby reducing the operator revenue. To achieve this goal, we design a functionality named CSACC (shown in Fig. 1) aiming of maximizing the resource utilization by accepting new slice requests while taking into account the queue status, the resource availability, the resource requirements, and slice priorities at the runtime. The CSACC decides to 1) scale down the resource allocated to a deployed BE slice and 2) accept one or multiple requests. More specifically, by scaling down resources allocated to a BE slice, the CSACC can accept new GS slice requests or approve the demand for additional resources from a deployed GS slice.

A. Action and Reward Models

The CSACC monitors the overall system state space, defined in (5), and uses a reinforcement learning algorithm to learn the optimal policy that finds the proper admission and congestion control actions to maximize the long-term system reward. At each time slot t (see Fig. 2), the CSACC selects an action $\boldsymbol{a} \in \mathcal{A}$ that determines the number of accepted GS and BE requests a_g and a_b, respectively. Thus, the CSACC action space is defined as follows:

$$\mathcal{A} \triangleq \{\boldsymbol{a} = (a_g, a_b)\}. \tag{16}$$

It is important to note that the slice requests in each queue are served in a first-input-first-output fashion, e.g., if $a_g = 2$, the first two slice requests of the GS queue are served. The actions chosen at each time slot must ensure that the number of BE (resp. GS) slice requests accepted does not exceed the actual number of BE (resp. GS) slice requests in the queue:

$$a_g(t) \leq s_g(t) \quad \text{and} \quad a_b(t) \leq s_b(t). \tag{17}$$

In addition, the constraints in (18) guarantee that, for the accepted BE slices, the sum of allocated resources for each resource type is less than or equal to the current resource availability:

$$\begin{aligned} \sum_{n=1}^{a_b(t)} d'^{b}_{n,r} &\leq \; r(t), \\ \sum_{n=1}^{a_b(t)} d'^{b}_{n,c} &\leq \; c(t), \\ \sum_{n=1}^{a_b(t)} d'^{b}_{n,m} &\leq \; m(t). \end{aligned} \tag{18}$$

Finally, the constraints in (19) ensure that the acceptance of new GS slices does not excessively degrade the quality of service of the running BE slices, by guaranteeing that a minimum amount of resources $\delta_j,\ j \in \{r, c, m\}$, is maintained at each deployed BE slice:

$$\begin{aligned} \sum_{n=1}^{a_g(t)} d'^{g}_{n,r} + \sum_{n=1}^{a_b(t)} d'^{b}_{n,r} &\leq r(t) + x_r(t) - \delta_r u_b(t), \\ \sum_{n=1}^{a_g(t)} d'^{g}_{n,c} + \sum_{n=1}^{a_b(t)} d'^{b}_{n,c} &\leq c(t) + x_c(t) - \delta_c u_b(t), \\ \sum_{n=1}^{a_g(t)} d'^{g}_{n,m} + \sum_{n=1}^{a_b(t)} d'^{b}_{n,m} &\leq m(t) + x_m(t) - \delta_m u_b(t). \end{aligned} \tag{19}$$

The aim of CSACC is to increase the operator revenue by maximizing the number of accepted slices, while limiting the probability that an arriving GS slice request is dropped because the queue is full, which leads to large revenue losses. Accordingly, to model the instantaneous reward associated with a state-action pair, we define a function that takes into account the numbers of accepted GS and BE slices as well as the number of dropped GS requests as follows:

$$R(\mathbf{s}(t), \mathbf{a}(t)) = a_g(t)w_g + a_b(t)w_b - n_d(t)l_d, \tag{20}$$

where w_g and w_b are the rewards for each accepted GS and BE slice, respectively. In addition, l_d is the cost for dropping a GS slice request, and $n_d(t)$ is the number of instantaneous dropped GS slices, which can be computed as follows:

$$n_d(t) = \max\{s_g(t) - a_g(t) + n_g(t) - Q_g, 0\}, \tag{21}$$

where Q_g is the maximum length of the GS queue (see Section II-B).

B. The SARSA Algorithm

SARSA is an online reinforcement learning algorithm aiming to find a stationary policy that associates a given system state with a proper action such that the expected total discounted reward is maximized. We define the expected total discounted reward counting from an initial state-action pair $(\mathbf{s}, \mathbf{a})$ over an infinite time horizon as follows [15]:

$$Q(\mathbf{s}, \mathbf{a}) = \mathbb{E}\left\{\sum_{t=0}^{\infty} \gamma^t R(\mathbf{s}(t), \mathbf{a}(t)) \mid \mathbf{s}_0 = \mathbf{s},\ \mathbf{a}_0 = \mathbf{a}\right\}, \tag{22}$$

where $R(\mathbf{s}(t), \mathbf{a}(t))$ is the instantaneous reward defined in (20) and $\gamma \in [0, 1)$ is a discount factor that determines the importance of future rewards. Reinforcement learning algorithms typically represent Q-function with a tabular format (i.e., a look-up-table), and the Q-values are arbitrarily initialized. Then, during the learning phase, the algorithm iteratively updates the Q-values when moving between state-action pairs (see the Algorithm 2), until convergence. At the end of the learning phase, if each state-action pair has been visited sufficiently often, the Q-values converge to an intermediate minimum, from which the optimal action can be computed as follows:

$$\mathbf{a}^* = \underset{\mathbf{a}\in\mathcal{A}}{\text{argmax}}\ Q(\mathbf{s}, \mathbf{a}),$$

where $\mathcal{A}$ is the set of actions defined in (16). To ensure the convergence of SARSA to the optimal policy and avoid the local minimum, it is necessary, during the learning phase, to randomly switch from one policy to another using e.g., ϵ-greedy or Boltzmann exploration schemes [26], while slowly decaying the exploration rate, e.g., ϵ for the ϵ-greedy, to zero.

Although the reinforcement learning model can be successfully used to address small-scale MDP problems, it becomes impractical in realistic cases characterized by large state space and

Algorithm 2 SARSA

```
Input: action set $\mathcal{A}$, reward function $R$, and parameters $\alpha, \gamma \in [0,1)$, and $\epsilon > 0$.
1: Randomly initialize $Q(\mathbf{s},\mathbf{a}), \forall \mathbf{s} \in \mathcal{S}, \mathbf{a} \in \mathcal{A}$;
2: for $t := 1$ to $T$ do
3:     Observe the current state $\mathbf{s}$;
4:     Select an action $\mathbf{a} \in \mathcal{A}$, using policy derived from $Q$ (e.g. $\epsilon$-greedy), observe the reward $R$ and the new
   state $\mathbf{s}'$;
5:     Take $\mathbf{a}' \in \mathcal{A}$ from $\mathbf{s}'$, using policy derived from $Q$;
6:     Update $Q(\mathbf{s},\mathbf{a})$ as follows: $Q(\mathbf{s},\mathbf{a}) \leftarrow Q(\mathbf{s},\mathbf{a}) + \alpha\,[R + \gamma Q(\mathbf{s}',\mathbf{a}') - Q(\mathbf{s},\mathbf{a}))]$;
7:     $\mathbf{s} \leftarrow \mathbf{s}'$, $\mathbf{a} \leftarrow \mathbf{a}'$;
8: end for
```

action set, as computational requirements of reinforcement learning grow exponentially with the number of state variables, i.e., the so-called *curse of dimensionality*.

In this work, when the SA is implemented, the amount of resources required by a new slice to be deployed depends on the similarity with the other slices being in the same NSI (14). Therefore, in (5), the sizes of sets of the available resources $\mathcal{S}_p$ and the resources allocated to the BE slices $\mathcal{X}_p$ become very large, and conventional reinforcement learning algorithms cannot be used to solve the cross-slice congestion and admission control. The state space size can be computed as follows:

$$|\mathcal{S}| = |\mathcal{S}_g| \cdot |\mathcal{S}_b| \cdot |\mathcal{S}_p| \cdot |\mathcal{U}_g| \cdot |\mathcal{U}_b| \cdot |\mathcal{X}_p|.$$

Let k denote the number of possible similarity values that can be obtained in (13), when the SA is implemented. In this case, the size of $\mathcal{X}_p$ is equal to $k^3(X_r+1)(X_c+1)(X_m+1)$; thus, $|\mathcal{S}|$ increases dramatically and learning the optimal policy with a standard reinforcement learning algorithm becomes infeasible. To deal with this issue, in the next section, we design an approximation function that enables to generalize the Q-function and accordingly to find the optimal policy in high complexity systems. Note that, SARSA, being an on-policy scheme, is characterized by milder convergence conditions compared to more classical (off-policy) reinforcement learning schemes (such as Q-Learning) when used with linear function approximation (LFA) [15]. Both linear and non-linear function approximation methods can be used to learn a generalized policy and apply reinforcement learning in realistic and complex problems. On the one hand, LFA requires skillful design of the features used as a linear basis to represent the Q-

Algorithm 3 SARSA With Linear Function Approximation

```
Input: action set $\mathcal{A}$, reward function $R$, feature vector $\boldsymbol{\phi}(\mathbf{s},\mathbf{a})$, and parameters $\alpha, \gamma \in [0,1)$, and $\epsilon > 0$.
1: Randomly initialize the feature weights $\boldsymbol{\theta}(\mathbf{s},\mathbf{a})$.
2: for $t := 1$ to $T$ do
3:     Select an action $\mathbf{a} \in \mathcal{A}$, using policy derived from $Q$ (e.g. $\epsilon$-greedy), observe the reward $R$ and the new state $\mathbf{s}'$;
4:     Take $\mathbf{a}' \in \mathcal{A}$ from $\mathbf{s}'$, using policy derived from $Q$;
5:     Let $\boldsymbol{e}_t = \boldsymbol{\phi}(\mathbf{s},\mathbf{a})$, and $k_t = R_t(\mathbf{s},\mathbf{a}) + \gamma Q_{\boldsymbol{\theta}_t}(\mathbf{s}',\mathbf{a}') - Q_{\boldsymbol{\theta}_t}(\mathbf{s},\mathbf{a})$
6:     Update weights as follows: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t k_t \boldsymbol{e}_t$
7:     Replace $\mathbf{s} \leftarrow \mathbf{s}'$ and $\mathbf{a} \leftarrow \mathbf{a}'$
8: end for
```

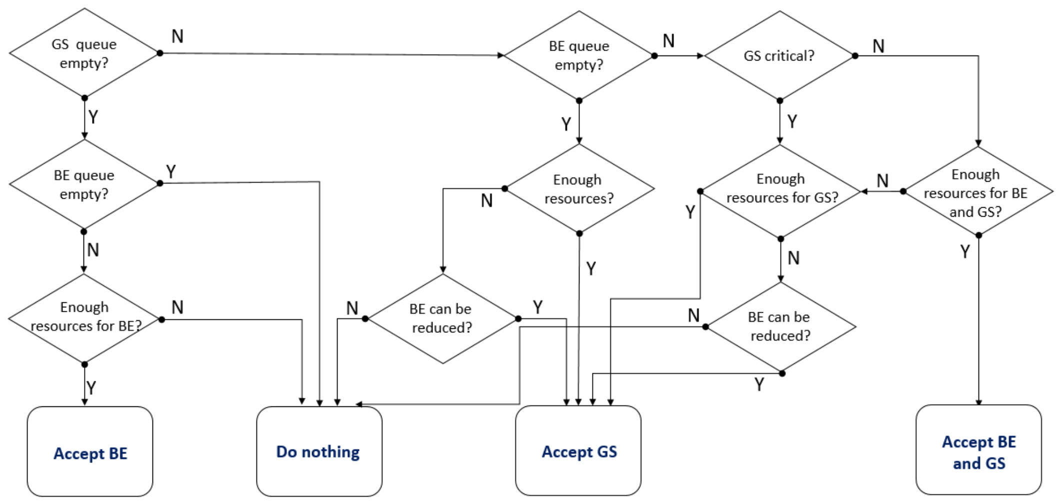


Figure 5: Cross-slice admission and congestion control flowchart.

function. On the other hand, non-linear methods are characterized by a larger design complexity: for instance, when using artificial neural networks, it is necessary to select a proper class of the neural network (such as fully connected, convolution, or recurrent), the number of layers, the number of weights, and the activation functions. For this reason, in the next section we present a scheme based on SARSA with LFA to find the optimal strategy for cross-slice admission and congestion control.

V. SARSA With Linear Function Approximation

In this section, we discuss how the LFA can be integrated into the SARSA to generalize the optimal policy and deal with the curse of dimensionality for large-scale problems. When using SARSA with LFA, the Q-function can be represented as a linear combination of a designed feature vector $\boldsymbol{\phi}(\mathbf{s},\mathbf{a}) = [\phi_1(\mathbf{s},\mathbf{a}), \phi_2(\mathbf{s},\mathbf{a}), \ldots, \phi_{\mathrm{F}}(\mathbf{s},\mathbf{a})]$. Accordingly, the Q-function can be approximated as follows:

$$Q(\mathbf{s},\mathbf{a}) \approx Q_\theta(\mathbf{s},\mathbf{a}) = \theta_0 + \sum_{i=1}^{\mathrm{F}} \theta_i \phi_i(\mathbf{s},\mathbf{a}), \tag{23}$$

where θ_i is the weight of the i^{th} feature and θ_0 is a bias term. Therefore, instead of explicitly learning the optimal Q-values for all state-action pairs, it is sufficient to learn the values of the $(F+1)$ weights related to the feature functions, i.e., the complexity of the learning process scale down from $\mathcal{O}(|\mathcal{S}||\mathcal{A}|)$, with SARSA, to $\mathcal{O}(F+1)$ with SARSA with LFA. To learn the weights, the classical approach is to minimize the Mean-Squared Error (MSE) over a probability distribution of the state-action pairs $\mathcal{Z}$, i.e.,

$$\zeta(\theta) = \mathbb{E}_{\mathcal{Z}}[(Q(\mathbf{s},\mathbf{a}) - Q_\theta(\mathbf{s},\mathbf{a}))^2]. \tag{24}$$

Stochastic Gradient Descent (SGD) is the most effective and popular method used in machine learning to find the local minimum of the MSE in (24) by updating the set of weights in (23) in the opposite direction of the gradient of the objective function $\nabla_\theta \zeta(\theta)$. Moreover, instead of computing the full expectations in this gradient, the SGD iteratively updates the weights after each training sample, as follows:

$$\theta \leftarrow \theta - \frac{\alpha}{2}\nabla_\theta \zeta(\theta) = \theta + \alpha[(Q(\mathbf{s},\mathbf{a}) - Q_\theta(\mathbf{s},\mathbf{a}))\boldsymbol{\phi}(\mathbf{s},\mathbf{a})], \tag{25}$$

where the learning rate α determines the size of the steps in the SGD. For reinforcement learning, $Q(\mathbf{s},\mathbf{a})$ is assumed to be unknown, and therefore, in (25), we use the bootstrapping technique [27], which consists in replacing the missing observation with an unbiased estimate $R(\mathbf{s},\mathbf{a}) + \gamma Q_{\boldsymbol{\theta}}(\mathrm{s}',\mathrm{a}')$. Since each estimate is an unbiased estimate, the SGD converges to a local minimum of the MSE if the learning rate α decreases appropriately during the learning phase. The algorithm of SARSA with LFA is sketched in Algorithm 3. To conclude this section, we discuss the design of the feature functions for the proposed CSACC. LFA methods have attracted the attention of the research community because of not only their convergence guarantees, but also the possibility to integrate feature functions that add prior knowledge to reinforcement learning

Table II: Slice templates in terms of required NFs.

	NF 1	NF 2	NF 3	NF 4	NF 5	NF 6	NF 7	NF 8	NF 9	NF 10
Slice template 1		x		x	x			x		x
Slice template 2		x		x	x		x			x
Slice template 3	x			x		x		x	x	
Slice template 4		x		x		x		x	x	
Slice template 5	x		x		x		x		x	
Slice template 6	x		x			x	x		x	

systems. Intuitively, the features design should take into account different state and action spaces along which approximation may be appropriate. Accordingly, we construct feature functions that consider the actions, the state space characteristics, and how state and action pairs jointly define the instantaneous reward in (20). Fig. 5 summarizes all the feature functions adopted by the CSACC and shows its main procedures. The CSACC evaluates whether to accept slice requests based on the queue status. If both queues are empty, no slice is accepted. If the GS queue is empty, the CSACC evaluates the required resources to accept BE slices and decides the number of accepted BE slice requests based on the available resources. In contrast, when the GS queue has slice requests awaiting to be accepted, the CSACC checks the resource availability while prioritizing GS slice requests over BE slice requests, in particular if the status of the GS queue is critical, and future GS requests may be dropped. If the resources are not sufficient, the CSACC evaluates the amount of resources that can be reduced from running BE slices, and accepts GS slice requests accordingly. Otherwise, if the available resources allow to accept both BE and GS slices, the CSACC accepts slices from both queues. In this work, we consider that the GS queue is critical if the number of GS slice requests, s_g, is higher than a given threshold $\hat{q}_g$, and additional future requests may be dropped.

VI. Simulation Results

To evaluate the performance of proposed framework for slice deployment and orchestration, we consider a 5G system using network slicing to meet the heterogeneous requirements of future 5G services. We consider that each slice request arriving in the system is related to one of the 6 slice templates shown in Table II. Specifically, the request n requires a set $\mathcal{D}_n$ of 5 NFs among the $K = 10$ NFs available in the 5G system, and each NF $NF_k \in \mathcal{D}_n$ is characterized by a

Table III: Simulation parameters.

Notation	**Parameter**	**Value**
$d_{n,j}$	n^{th} slice resource request	200
R, C, M	Network resources	800
p^b	BE request arrival probability	0.85
f^b, f^g	Departure probabilities of GS, BE slices	0.35
N_b, N_g	Maximum number of arriving BE, GS requests	2
Q_b, Q_g	Maximum length of the slice request queues	4
w_b	Revenue for accepting a BE request	1
w_g	Revenue for accepting a GS request	1.7
l_d	Cost for dropping a GS request	$5w_g$
ρ_k, χ_k, μ_k	Resources for activating a NF	0

set of configuration parameters $\mathcal{L}_{n,k}$ represented by one binary vector $J = 1$ of length 2. We assume that each NF needs 40 MHz, 40 GFLOPS/s, and 40 GB in terms of communication, computing, and cloud storage resources, respectively; therefore, the total communication $d_{n,r}$, computing $d_{n,c}$, and cloud storage $d_{n,m}$ demand for a slice is equal to 200 MHz, 200 GFLOPS/s, and 200 GB (see (4)).

During the congestion control phase (see (19)), the minimum amount of resources for running a BE slice i.e., δ_j, $j = \{r, c, m\}$, is equal to 100 MHz, 100 GB, and 100 GFLOPS/s. Overall, we assume that the network has 800 MHz, 800 GFLOPS/s, and 800 GB communication, computing, and cloud storage resources to satisfy the requirements of the deployed slices. Regarding the learning parameters, we initially set $\alpha = 0.5$ and $\epsilon = 1$; then, during the learning process, their values are iteratively reduced to strike a balance between performance and learning speed. We set the discount factor $\gamma = 0.8$ by random search, i.e., by cross-checking the obtained results for different values of the hyper-parameter. Finally, Table III describes the main simulation parameters.

A. SARSA with LFA for Cross-Slice Admission Control

In this section, we focus only the slice admission control function (AC), i.e., we do not allow the controller to reduce the resources allocated to the BE slices, in order to increase the number of deployed slices. Specifically, we evaluate the results obtained when using SARSA with LFA

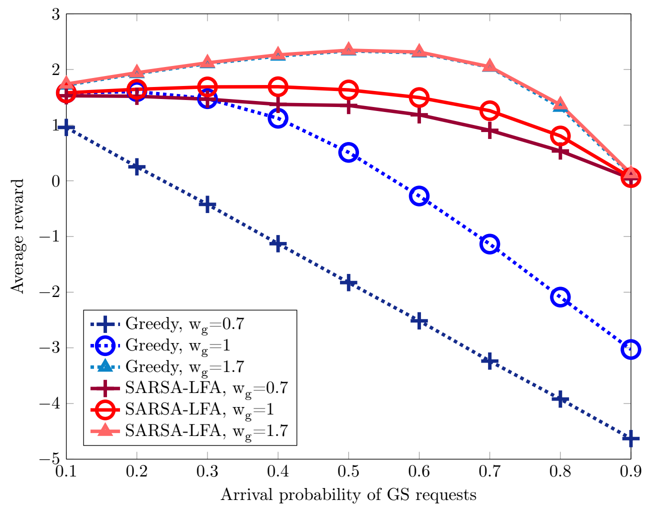
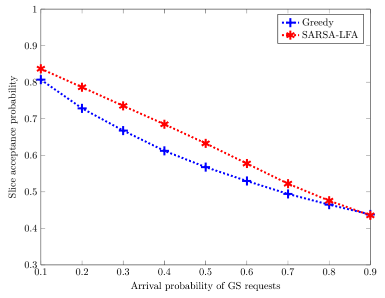


Figure 6: Average reward obtained by the greedy scheme and SARSA with LFA as a function of GS admission reward and arrival probability.

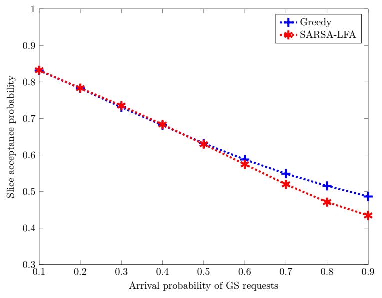
for different values of GS acceptance reward w_g and GS arrival probability p^g. We compare the performance of the reinforcement learning scheme with a greedy scheme that selects the actions that maximize an immediate reward, i.e., the number of accepted slices at each time slot.

Fig. 6 shows the average reward obtained by the greedy scheme and SARSA with LFA as a function of the GS admission reward w_g and arrival probability p^g. First, we can see that, for each value of w_g, there is an optimal value of p^g that maximizes the average reward obtained by the SARSA with LFA. Increasing p^g enhances the average reward until when the slice arrival process can be efficiently managed by the AC; however, continuing to increase p^g beyond this optimal value decreases the system performance due to the limited queue size and available resources. In addition, we observe that for $w_g = 0.7$ and $w_g = 1$, the SARSA with LFA achieves a much greater average reward than that of the greedy scheme, especially for higher values of p^g. In fact, the greedy scheme is not able to predict the event of dropping new GS slice requests. In contrast, for very large value of GS reward, i.e., $w_g = 1.7$, both the greedy and SARSA with LFA achieve the same performance, since, in this case, they both prefer to always accept a GS slice request rather than a BE one.

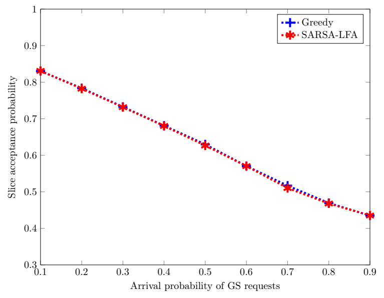
To better understand this performance, we show in Fig. 7 the probabilities of accepting and dropping a slice request under different values of p^g and w_g when using SARSA with LFA and the greedy scheme. In Fig. 7(a), we observe that, for $w_g = 0.7$, SARSA with LFA achieves higher



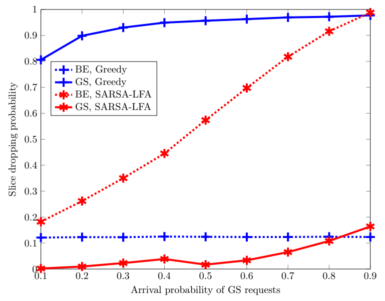
(a) $w_g = 0.7$



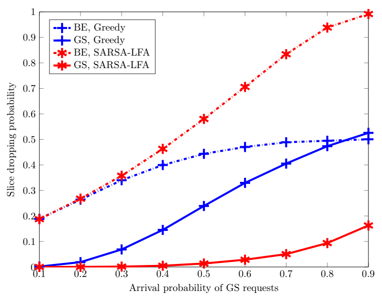
(b) $w_g = 1$



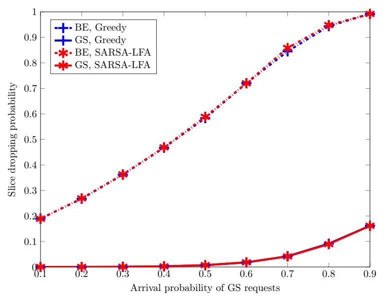
(c) $w_g = 1.7$



(d) $w_g = 0.7$



(e) $w_g = 1$



(f) $w_g = 1.7$

Figure 7: Slice acceptance and dropping probabilities obtained by the greedy scheme and SARSA with LFA as a function of GS admission reward and arrival probability.

slice acceptance probabilities than those of the greedy scheme. Moreover, Fig. 7(d) shows that SARSA with LFA can reduce the GS dropping probability at the cost of a higher BE dropping probability. This is due to the fact that when $w_g = 0.7$, the greedy scheme prioritizes BE slices, which provides a larger instantaneous reward. In contrast, SARSA with LFA can learn and balance the reward perceived for accepting new slices and the cost for dropping new GS requests, and thus achieving the best performance in terms of average reward as shown in Fig. 6.

Fig. 7(b) shows that, for $w_g = 1$, the greedy scheme achieves higher slice acceptance probabilities as compared to SARSA with LFA when p^g is larger than 0.5. Also, we can see in Fig. 7(e) that SARSA with LFA and the greedy scheme have similar performance in terms of dropping probabilities when p^g is lower than 0.3. However, for larger values of p^g, the greedy scheme leads to GS and BE dropping probability close to 0.5. This performance is not acceptable for GS slices; in fact, for large values of p^g, SARSA with LFA prioritizes GS slices over the BE requests in order to limit the associated dropping probability. Finally, as expected, when

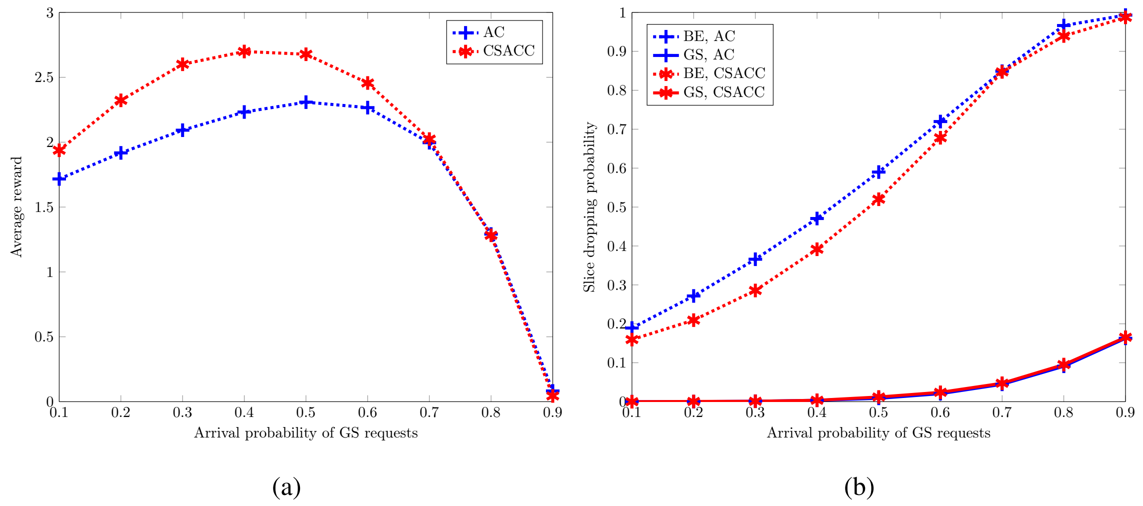


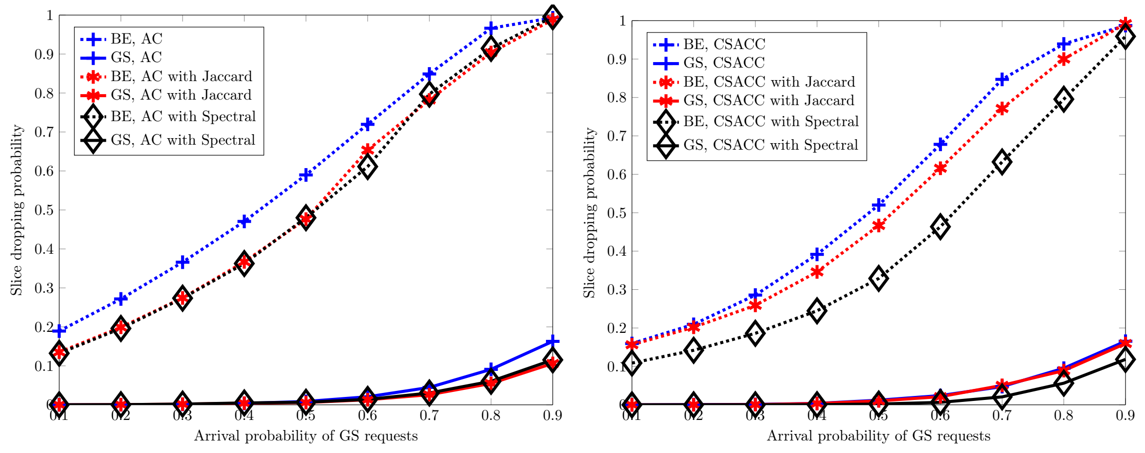
Figure 8: (a) Average reward and (b) slice dropping probability with AC and CSACC as a function of the GS arrival probability.

$w_g = 1.7$, SARSA with LFA and the greedy scheme achieve similar performance in terms of slice acceptance and dropping probabilities (see Fig. 7(c) and 7(f)).

These results confirm the capability of the proposed AI algorithm to adjust the CSACC strategy to the arrival and departure statistics of the slice requests accordingly, in order to maximize the long-term average reward.

B. SARSA with LFA for Cross-Slice Admission and Congestion Control

To assess the impact of the congestion control function for slice deployment and orchestration, we compare the system performance by implementing two strategies, i.e., the strategy using only admission control (AC) and the strategy using both cross-slice admission and congestion control (CSACC). Fig. 8(a) shows the average reward achieved by SARSA with LFA when implementing AC (plus marked line) and CSACC (star marked line). As expected, the CSACC enhances the average reward of the system since it enables to increase the number of the average accepted slices. Specifically, CSACC enables to improve the average reward up to 23% with respect to the reward achieved by the simpler AC scheme. However, we can observe that when the GS arrival probability p^g is larger than 0.7 the two approaches have the same performance. The system under investigation is not well-dimensioned for this range of values, where both the controllers stop to accept BE slices requests to limit the GS slice dropping probability. Therefore,



(a) Cross-slice admission control (b) Cross-slice admission and congestion control

Figure 9: Slice dropping probability as a function of the GS request arrival probability with and without slice analytics.

here, implementing the CC may not lead to significant benefits. This finding is verified in the results in Fig. 8(b), where we can observe the impact of the CSACC in terms of slice dropping probability. Specifically, for low and medium values of p^g, the CSACC improves the system performance by reducing the number of dropped BE slice requests. Without the CSACC, the system has to limit the acceptance of low-priority BE slices, in order to save resources for future GS slice requests. In contrast, the proposed scheme allows to increase the percentage of accepted BE slices since part of their resources can be opportunistically allocated to new GS requests when needed.

C. Cross-Slice Admission and Congestion Control with the Slice Analytics

In this section, we study the impact of the slice analytics (SA) on the system performance. The SA function optimizes the assignment of new slice requests to existing NSIs such that the additional required resources to deploy them are limited. Here, our goal is to evaluate the performance of two assignment algorithms presented in Section III-A, i.e., the Jaccard similarity-based and spectral clustering-based assignment schemes.

Fig. 9(a) shows the slice dropping probability as a function of GS arrival rate achieved when using only the cross-slice AC, the AC in conjunction with the Jaccard similarity scheme, and the AC in conjunction with the spectral clustering scheme. We observe that both the proposed

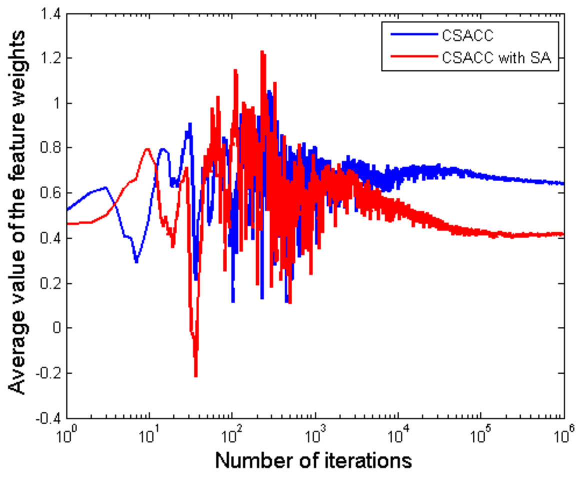
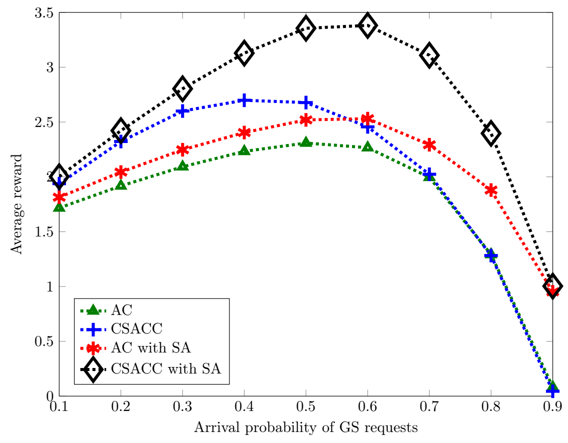


Figure 10: Convergence of feature weights in SARSA with LFA.

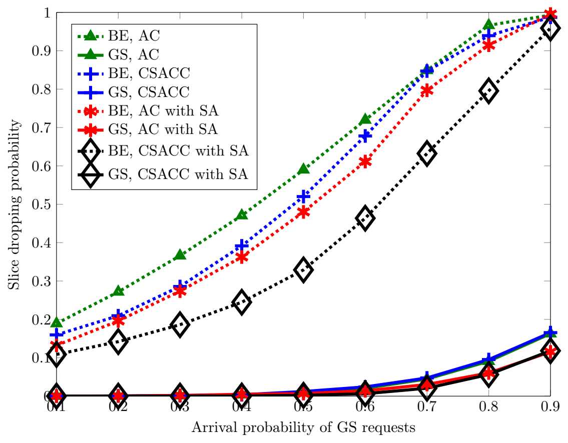
strategies enhance the system performance achieved by the AC by decreasing the dropping probabilities for both BE and GS slice requests. However, in this case the two SA algorithms achieve the same performance. Fig. 9(b) shows the slice dropping probability as a function of GS arrival rate achieved when using only the CSACC, the CSACC in conjunction with the Jaccard similarity scheme, and the CSACC in conjunction with the spectral clustering scheme. Simulation results show that in this case, the spectral clustering outperforms the Jaccard similarity by decreasing the dropping probabilities for both BE and GS slice requests. When used with CSACC, the spectral clustering algorithm increases the number of BE slices that can be deployed thus augmenting the amount of resources that can be opportunistically subtracted to the GS slice to the BE slices. This results in reducing the BE requests dropping probability.

The above discussed results highlight the benefits of SA used in conjunction with the proposed CSACC. To assess the feasibility of implementing this framework, we now evaluate the convergence of SARSA with LFA when used for CSACC only and when implemented for CSACC in conjunction with SA. Specifically, simulation results in Fig.10 show the average values of the feature weights as a function of the iterations number. We observe that the proposed LFA scheme allows the feature weights to converge in a reasonable time, for both the scheme, in the order of 5×10^4 iterations.

We now conclude our analysis by summarizing the performance achieved by the pillars of our slice deployment and orchestration framework. Fig. 11(a) shows the average reward obtained



(a)



(b)

Figure 11: (a) Average reward and (b) slice dropping probability for different schemes as a function of the GS arrival probability.

by our system when using only the cross-slice AC, CSACC, AC with spectral clustering (AC with SA), and CSACC with spectral clustering (CSACC with SA). We can notice that the CSACC leads to a large reward gain as compared to the AC for low values of GS request arrival probabilities; in fact, for these values, the number of BE slices accepted in the system is significant, and a large amount of their resources can be opportunistically re-used to serve GS slice requests. In contrast, the gain brought by the SA increases with the GS request arrival probability, i.e., the larger p^g, the larger the amount of resources that can be shared among the deployed slices. Moreover, we observe that with CSACC with SA, the reward gain with respect to the simple AC is larger than the sum of the gains lead by each single strategy. This gain is due to the large reduction in the BE dropping probability as shown in Fig. 11(b). In fact, the CSACC with SA enables to reduce the GS dropping probability and the BE dropping probability up to 28% and 44%, respectively. Overall, we can conclude that our framework is able, by optimizing the network resource usage, to increase the operator revenue while satisfying the priorities among the different services.

VII. Conclusion

In this paper, we have designed a novel AI framework for network slice deployment and orchestration. Specifically, in the context of the 3GPP NSMF, we have proposed two novel functions for the resource efficient management of slice requests in 5G systems: SA and CSACC.

SA assigns each new slice request to an existing NSI, based on the shared NFs, such that the additional amount of resource required to deploy the new slice is minimized. CSACC aims to maximize the system revenue while taking into account slice priorities, network slice requirements, and resource availability. We have formulated this problem using reinforcement learning and designed the on-policy scheme, based on SARSA with LFA, to find the optimal control strategy that maximizes the long term expected reward of the system. Our results show that using SA and CSACC jointly can reduce the GS and BE slice request dropping probability up to 28% and 44%, respectively. In future studies, we will focus on a system dealing with slice requests characterized by more heterogeneous requirements and priorities. Moreover, we will investigate how to size the substrate network and the associated resources, such that the slice dropping probability is further minimized.

References

[1] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. D. Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5G: A Tutorial Overview of Standards, Trials, Challenges, Deployment, and Practice," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1201–1221, Jun. 2017.

[2] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker, "Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 72–79, May 2017.

[3] 3GPP TSG Services and System Aspects, "TR 28.801, Telecommunication management; Study on management and orchestration of network slicing for next generation network," *V15.0.0*, Dec. 2017.

[4] ETSI White Paper No. 22, "Improved operator experience through Experiential Networked Intelligence (ENI)," Oct. 2017.

[5] D. T. Hoang, D. Niyato, P. Wang, A. De Domenico, and E. Calvanese Strinati, "Optimal Cross Slice Orchestration for 5G Mobile Services," *Proc. IEEE VTC Fall*, Aug. 2018.

[6] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis, and X. Costa-Perez, "Optimising 5G infrastructure markets: The business of network slicing," in *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.

[7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[8] P. Caballero, A. Banchs, G. de Veciana, X. Costa-Pérez, and A. Azcorra, "Network Slicing for Guaranteed Rate Services: Admission Control and Resource Allocation Games," *IEEE Transactions on Wireless Communications*, vol. 17, no. 19, pp. 6419–6432, Oct. 2018.

[9] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network slicing in 5G: An auction-based model," in *Proc. IEEE ICC*, May 2017, pp. 1–6.

[10] M. Leconte, G. S. Paschos, P. Mertikopoulos, and U. C. Kozat, "A resource allocation framework for network slicing," in *Proc. IEEE INFOCOM*, May 2018, pp. 2177–2185.

[11] C. Delgado, M. Canales, J. Ortín, J. R. Gállego, A. Redondi, S. Bousnina, and M. Cesana, "Joint Application Admission Control and Network Slicing in Virtual Sensor Networks," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 28–43, Feb. 2018.

[12] A. Bosneag and M. X. Wang, "Intelligent network management mechanisms as a step towards 5G," in *Proc. 8th International Conference on the Network of the Future (NOF)*, Nov. 2017, pp. 52–57.

[13] Y. H. Chen, C. J. Chang, and C. Y. Huang, "Fuzzy Q-Learning Admission Control for WCDMA/WLAN Heterogeneous Networks with Multimedia Traffic," *IEEE Transactions on Mobile Computing*, vol. 8, no. 11, pp. 1469–1479, Nov 2009.

[14] M. Al-Maitah, O. O. Semenova, A. O. Semenov, P. I. Kulakov, and V. Y. Kucheruk, "A Hybrid Approach to Call Admission Control in 5G Networks," *Advances in Fuzzy Systems*, 2018.

[15] F. S. Melo, S. P. Meyn, and M. I. Ribeiro, "An Analysis of Reinforcement Learning with Function Approximation," in *Proc. 25th International Conference on Machine Learning*, July 2008, pp. 664–671.

[16] D. M. Gutierrez-Estevez, M. Gramaglia, A. De Domenico, G. Dandachi, S. K. U. Elzur, and Y. Wang, "Artificial Intelligence for Elastic Management and Orchestration of 5G Networks," *to appear in IEEE Wireless Communications Magazine*.

[17] 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 *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.

[18] NGMN Alliance, "Description of network slicing concept." [Online]. Available: https://www.ngmn.org/

[19] P. Jaccard, "Étude comparative de la distribution florale dans une portion des alpes et des jura," *Bull Soc Vaudoise Sci Nat*, vol. 37, pp. 547–579, 1901.

[20] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in neural information processing systems*, 2002, pp. 849–856.

[21] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[22] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[23] S. Tsironis, M. Sozio, M. Vazirgiannis, and L. Poltechnique, "Accurate spectral clustering for community detection in MapReduce," in *Proc. Advances in Neural Information Processing Systems (NIPS) Workshops*, 2013.

[24] Q. Chen, W. Wang, Y. Wang, P. Zhou, and Z. Zhang, "Content caching clustering based on piecewise interest similarity," in *Proc. IEEE GLOBECOM*, Dec. 2017, pp. 1–6.

[25] P. Hassanzadeh, A. Tulino, J. Llorca, and E. Erkip, "Cache-aided coded multicast for correlated sources," in *Proc. of 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Sept 2016, pp. 360–364.

[26] V. Kuleshov and D. Precup, "Algorithms for multi-armed bandit problems," *CoRR*, vol. abs/1402.6028, 2014. [Online]. Available: http://arxiv.org/abs/1402.6028

[27] M. Geist and O. Pietquin, "Algorithmic Survey of Parametric Value Function Approximation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 6, pp. 845–867, June 2013.

Response to the Comments of the Reviewers

Ghina Dandachi, Antonio De Domenico, Dinh Thai Hoang, and Dusit Niyato

I. Response to Comments of the Editor

(E1) I have been able to get reviews for your paper ”An Artificial Intelligence Framework for Slice Deployment and Orchestration in 5G networks,” paper number TCCN-TPS-19-0034, submitted for possible publication in the IEEE Transactions on Cognitive Communications and Networking. These reviews, along with the recommendation provided by one of our technical editors, are attached below.

Based on the reviewers’ comments and the editor’s recommendation, I would like you to update your paper and submit a revised version for further processing. .

(R1) We want to thank the editor for handling the timely review of this paper. We would also like to thank the editor for allowing us to resubmit a revised version of the manuscript to IEEE Transactions on Cognitive Communications and Networking. The detailed point-by-point answers to the reviewers’ comments for the previous version of the manuscript follows below. We hope that the reviewers and the editor would find the modifications made in the manuscript convincing and up to their expectations.

References numbering in the present document is independent of that adopted in the paper. We invite reviewers to refer to the bibliography at the end of the document for what concerns the references cited in this response.

The comments of the reviewers are highlighted in bold font, and the response is followed in normal font. Modifications in the revised version are in blue.

Before we begin our response, we would like to introduce the following abbreviation used to denote the reviewers’ comments and the corresponding responses:

- **(CX.Y)**: Yth comment of the Xth reviewer. For example: **(C1.1)** means comment 1 of Reviewer 1.
- **(RX.Y)**: Response to the Yth comment of the Xth reviewer. For example: **(R1.1)** means response to the comment 1 of Reviewer 1.

II. Response to the Comments of the Reviewer 1

(C1.1) This paper proposed a systematic network slice solution to optimize resource utilization, reduce slice request dropping rate and increase operator revenue for the 5G system. Specifically, two functions SA and CSACC are introduced in the 3GPP context. SA performs the evaluation on new slice requests in terms of slice similarity, then the cross-slice resource management and congestion control (CSACC) makes the tradeoff between the slice acceptance rate and dropping probability. In addition, linear approximation functions and stochastic gradient descent is applied to reduce the complexity of calculating optimal Q-values in the reinforcement learning model. Finally, experiment results are given to verify the effectiveness of the proposed solution. A well-articulated paper and a powerful framework.

(R1.1) We thank the reviewer for her/his valuable time for reviewing the manuscript and for recognizing the value of the proposed framework. Also, we would like to thank the reviewer for the remarks that have helped in improving the quality of the manuscript.

(C1.2) Just some comments in typos: (1) page 9, right column, the 1st paragraph, section VI. “Each of these NFs” $->$ “Each of the NF”;

(R1.2) Thanks for this remark; we have rephrased this sentence as suggested by the reviewer.

(C1.3) page 10, right column, the last sentence. “according” $->$ “accordingly”

(R1.3) Thanks for this remark; we have rephrased this sentence as suggested by the reviewer.

(C1.4) page 11, left column, the last sentence is a little ambiguous. “our proposed scheme enables to increase the percentage of accepted BE slices as a part of their resources can be opportunistically allocated to new GS requests when needed.” $->$ “our proposed scheme enables to increase the percentage of accepted BE slices as a part of their resources, which can be opportunistically allocated to new GS requests when needed.”

(R1.4) Thanks for this remark; we acknowledge that the original sentence was ambiguous and we have rephrased it to improve the paper readability. In fact, here, we aim to clarify that, since part of the resources allocated to the BE slices can be momentarily used to deploy high priority GS slice requests, the system can accept a larger number of BE requests, even when the amount of available resources is limited. The rephrased sentence is as follows:

> Without the CSACC, the system has to limit the acceptance of low-priority BE slices, in order to save resources for future GS slice requests. In contrast, the proposed scheme allows to increase the percentage of accepted BE slices since part of their resources can be opportunistically allocated to new GS requests when needed.

III. Response to the Comments of the Reviewer 2

(C2.1) This paper proposes an Artificial intelligence (AI) based framework slice deployment and orchestration in 5G networks. The manuscript proposes an architecture that mainly has three components, namely slice analytics (SA), admission control (AC), and congestion control (CC). The slice analytics attempts to find the similarity index between a submitted NF chain and the existing network slices through two different approaches with different complexities. The admission control and congestion control are considered jointly. The system model assumes two types of requests, namely a best-effort service and guaranteed service, which (as I understood) is used as a way to characterize the priority of services, where best-effort services can be scaled if need be. The joint admission control and congestion control module is implemented using a RL scheme (using SARSA). The paper in general proposes interesting ideas and results.

(R2.1) We thank the reviewer for her/his valuable time for reviewing the the manuscript and for recognizing the value of the proposed ideas and results. Also, we would like to thank the reviewer for the comments that have helped in improving the quality of the manuscript.

(C2.2) In Section I-A, the related works are well-written and organized. However, please focus on stating/elaborating on the works rather than their shortcomings or what has not been considered there.

(R2.2) Thanks for this remark; we have rephrased and shortened this section, removing the shortcomings of the related works, as suggested by the reviewer.

(C2.3) In Section II, the system model’s introduction before Sec. II-A is unnecessarily long. Please try to use more concise sentences and remove unnecessary ones.

(R2.3) Thanks for this remark; we acknowledge that the original system model’s introduction was unnecessarily long. Accordingly, we have shortened it by rephrasing part of the text and removing unnecessary sentences, as suggested by the reviewer. We hope that this has improved the paper readability.

(C2.4) For Sec. II-A, please elaborate more on what is the configuration parameters $L_{n,k}$. How does it relate to the required resources?

(R2.4) We thank the reviewer for raising this concern, and we admit that the concept of configuration parameters in the past version of the document was a bit vague. The network slice blueprint (or template) is a tool to achieve the flexibility and modularity required by network slicing. Specifically, it provides all the information required by the 5G system to design, deploy, and manage a network slice and its components.

The final definition of the end-to-end network slice blueprint is currently under investigation in the 5G community; however, it will be likely be an extension of the ETSI MANO Network Function Virtualization (NFV) descriptor [1], which only focuses on the virtualized deployment of a network service. A NFV descriptor includes of a set of attributes, named as flavors, which provide different options to deploy an instance of a network service/function. In turn, each flavor defines one or more instantiation levels, each specifying a different amount of virtual resources related to that flavor. In the context of an end-to-end network slice, each NF needs to be configured according to the type of supported service. For example, if we consider eMBB service, the network function implementing the 5G PHY layer needs to support a waveform with large bandwidth, modulation and coding schemes with high spectral efficiency, etc. The specific values of these parameters directly determine the end-to-end resources to be allocated to a given NF. For instance, the operational band specifies the bandwidth to be allocated to the NF, and the MCS together with the bandwidth can be used to evaluate the required computational resources [2].

To enhance the paper readability, we have changed the first part of Sec. II-A as follows:

> The NGMN Alliance defines the slice blueprint as a complete description of the structure, configuration, and the work flows to instantiate and control a network slice during its life-cycle [3]. In this work, we characterize a slice blueprint as follows:
>
> - A list of NFs that the slice requires and the description of the interactions among the NFs.
> - The parameters that describe the configuration of each NF.
>
> Depending on the type of service, the system will deploy a specific version of each NF, which enables to satisfy the service requirements. For example, in the case of eMBB service, the NF providing 5G PHY layer is characterized by precise configuration parameters as large bandwidth, modulation and coding schemes with high spectral efficiency, etc. The configuration of a NF, in turn, defines the amount of resources that it requires to be deployed in the 5G system.

(C2.5) Isn't there a network substrate? How is it defined, and how a network slice is defined? What are the nodes of the network. Is routing considered?

(R2.5) Thanks for this remark. In this paper, we mainly focus on the slice admission control problem and not on resource scheduling and instantiation. Thus, as in [4], [5], we assume that there is a pool of resources connected together. Therefore, substrate networks issues such as topology, node resource constraints, routing, and latency within the substrate network are out of our scope. Some of these constraints will be considered in our future work. Accordingly, we have modified the final sentence of the paper conclusion as follows:

> Moreover, we will investigate how to size the substrate network and the associated resources, such that the slice dropping probability is further minimized.

(C2.6) For Section II-B, first two sentences are repeated.

(R2.6) Thanks for this remark; we have deleted the second sentence.

(C2.7) For the state space of the RL scheme, for the available network resources, how is it defined. Is it the amount of residual resource for each type of each node in each NSI or each node in the network substrate?

(R2.7) As, mentioned in **(R2.5)**, in this paper, we mainly focus on the slice admission control problem and we do not consider constraints at the substrate node level. Thus, the set of available resources in the system space $\mathcal{S}_p$ refers to the overall 5G network resources, as there is a pool of resources well connected together.

Table I: Main Notations and System Parameters

Notation	Parameter
$\mathcal{S} \triangleq \mathcal{S}_g \times \mathcal{S}_b \times \mathcal{S}_p \times \mathcal{U}_g \times \mathcal{U}_b \times \mathcal{X}_p$	State space
$\mathbf{s}(t) \in \mathcal{S}$	System state
$s_g(t) \in \mathcal{S}_g, s_b(t) \in \mathcal{S}_b$	Slice requests in GS and BE queues
$\boldsymbol{s_p}(t) = (r, c, m) \in \mathcal{S}_p$	Communication [GHz], computing [GFLOPS/s], and storage [GB] resources
$u_g(t) \in \mathcal{U}_g, u_b(t) \in \mathcal{U}_b$	Deployed GS and BE slices
$\boldsymbol{x_p}(t) = (x_r, x_c, x_m) \in \mathcal{X}_p$	Allocated resources for deployed BE slices
$\boldsymbol{a}(t) = (a_g, a_b) \in \mathcal{A}$	Admission and Congestion Control action and action space
$a_g(t), a_b(t)$	Number of accepted GS and BE requests
$\Lambda_{n,n'}$	Jaccard similarity between slices n and n'
$\mathcal{B}$	Deployed Network Slice Instances
$d_{n,k,j}$	Resources of type j required by the k^{th} NF in the n^{th} request
$d_{n,j}$, $d'_{n,j}$	Resources of type j required by n^{th} slice before, after slice analytics
w_g, w_b	Rewards for each accepted GS and BE slices
$n_d(t), l_d$	Instantaneous dropped GS slices and dropping loss
$n_b(t)$, $n_g(t)$	New BE and GS requests
$R(\mathbf{s}(t), \mathbf{a}(t))$	Instantaneous reward
$Q(\mathbf{s}, \mathbf{a})$	Q-value of a state-action pair
α, γ	Learning rate and discount factor
$\boldsymbol{\phi}(\mathbf{s}, \mathbf{a})$	Feature vector
θ_i	Weight of the i^{th} feature

Table II: Simulation parameters.

Notation	Parameter	Value
$d_{n,j}$	n^{th} slice resource request	200
R, C, M	Network resources	800
p^b	BE request arrival probability	0.85
f^b, f^g	Departure probabilities of GS, BE slices	0.35
N_b, N_g	Maximum number of arriving BE, GS requests	2
Q_b, Q_g	Maximum length of the slice request queues	4
w_b	Revenue for accepting a BE request	1
w_g	Revenue for accepting a GS request	1.7
l_d	Cost for dropping a GS request	$5w_g$
ρ_k, χ_k, μ_k	Resources for activating a NF	0

(C2.8) The equations/symbols are confusing. Try to re-organize symbols. For example, the use of a table of notations are encouraged for important symbols.

(R2.8) Thanks for this remark. We acknowledge that the paper contains many symbols and that notation in the previous version of the manuscript could be confusing. We have gone through the document, changing it and correcting whenever possible. Also, we have introduced a table with the main notations (see Table I) and a table with the simulation parameters (see Table II).

(C2.9) Section III.B is not very clear. What do you mean by "new NSI network requirements must be evaluated", what are the requirements? What do you mean by "resources must be mutualized".

(R2.9) Thanks for this comment that gives us the opportunity to clarify the objective of the Slice Analytics. For each new service, the Network Management & Orchestration layer has to provide a Network Slice Instance (NSI) that fits the requested network requirements. This process can be realized either using an existing NSI or creating a new NSI [6]. Similarly, two NSIs can share part of the network functions, e.g., the access networks as in the Figure 1.

Deploying a new service through an existing NSI or creating a new NSI that shares part of its NFs with an existing NSI (or multiple NSIs) enables to optimize the network resource usage and speed up the service deployment. However, this process cannot be always be implemented, either because of isolation constraints or other network requirements, e.g., lack of available resources to deploy an additional service on the same NSI.

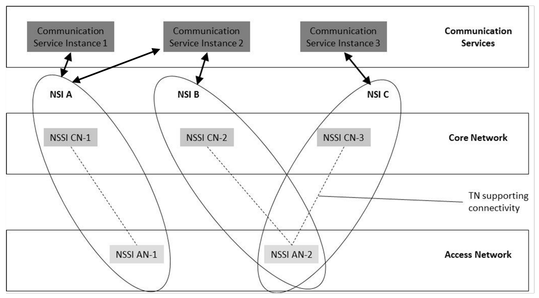

Communication Service Instance 1
Communication Service Instance 2
Communication Service Instance 3
Communication Services
NSI A
NSI B
NSI C
NSSI CN-1
NSSI CN-2
NSSI CN-3
Core Network
TN supporting connectivity
NSSI AN-1
NSSI AN-2
Access Network

Fig. 1. Subnetwork sharing across two or more network slices [6].

In our setting, given a new service request n characterized by requirements $\boldsymbol{T}_n = (d_{n,r}, d_{n,c}, d_{n,m})$ (see eq. (4) in Section II-A), we first identify (if it exists) the deployed NSI i that could efficiently (in terms of resource usage) satisfy the new service request n (see Section III-A). Then, we evaluate, based on the NFs shared between n and i, the resources that can be mutualized between them.

Accordingly, to successfully deploy the service n through the NSI i, this NSI needs additional resources, i.e., it has new resource requirements to successfully satisfy also those of the service n. These additional resources may correspond to the NFs that cannot be shared across the NSI i and the new service n, and are denoted as $\boldsymbol{T'}_n = \left(d'_{n,r}, d'_{n,c}, d'_{n,m}\right)$ (see eq. (15) in Section III-B).

As an example, please consider the case of two different broadcasters covering the same sport event through one network infrastructure: The Network Management & Orchestration can decide to deploy these services through the same NSI. In this way, the two services will share the radio resources allocated to transmit common contents such as the video, and receive specific resources for service-specific content such as the audio part of the show, advertisements, etc. To clarify this point, we have modified the first part of Section III-B, as follows:

> Both the SA strategies proposed in Section III-A allow to identify an already existing NSI, which is appropriate to support a new communication service. However, to satisfy the specific requirements of the new service request, the deployed NSI may require additional network resources. Thus, before finalizing this process, the new NSI network requirements must be evaluated and the associated resource request must be accepted by the CSACC (see Section IV). In this work, we assume that the amount of resources that can be mutualized by the existing NSI and the new service request depends on the shared NFs and their associated configuration parameters.

(C2.10) Section IV-A, please explain and elaborate more on the tradeoff between number of accepted slices and dropping probability.

(R2.10) Thanks for raising this question as the trade-off between resource utilization and long-term revenue is one of the main topics of this work. Specifically, maximizing the resource utilization may drive the system to accept as many slice requests as possible, even if these requests lead to limited revenue. This strategy, however, is myopic and, due to the limited available resources, may prevent the system from accepting future slice requests with high priorities, which provide high revenue for the operator. Thus, minimizing the dropping probability of high-priority slice requests requires to limit the acceptance of low-priority slice requests, thereby reducing the operator revenue. Overall, finding the strategy that optimizes the long-term revenue is a challenging problem as the slice request arrival and departure probabilities are unknown and the network resources are limited. When, at the beginning of Section IV and in Section IV-A, we refer to trade-off between number of accepted slices and GS dropping probability, we refer to the same trade-off. We acknowledge that this may be not clear for the reader. Accordingly, we have rephrased the text in these sections as follows:

> In this section, we propose a reinforcement learning framework that optimizes the trade-off between the number of accepted slices and the dropping probability of the network slice requests with high priorities, which in turns affects the long-term system revenue.

The aim of CSACC is to increase the operator revenue by maximizing the number of accepted slices, while limiting the probability that an arriving GS slice request is dropped because the queue is full, which leads to large revenue losses.

(C2.11) Section IV-B: How to decide the discount factor that determines the importance of future rewards.

(R2.11) The discount factor is a hyper-parameter that determines the present value of future rewards: in reinforcement learning, a reward received k time steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately [7]. When $\gamma < 1$, the infinite sum below (see (22) in the manuscript):

$$Q(\mathbf{s},\mathbf{a}) = \mathbb{E}\left\{\sum_{t=0}^{\infty} \gamma^t R(\mathbf{s}(t),\mathbf{a}(t)) \mid \mathbf{s}_0 = \mathbf{s},\ \mathbf{a}_0 = \mathbf{a}\right\}$$

has a finite value as long as the reward sequence $\{R(\mathbf{s}(t),\mathbf{a}(t))\}$ is bounded.

If γ=0, the agent is *myopic* in being concerned only with maximizing immediate rewards: its objective in this case is to learn how to choose $\mathbf{a}(t)$ so as to maximize only $R(\mathbf{s}(t),\mathbf{a}(t))$. As γ approaches 1, the Q-function takes future rewards into account more strongly and the agent becomes more farsighted. In our work, we set the value of γ=0.8 through random search, i.e., cross-checking the obtained results for different values of the hyper-parameter. To clarify this aspect, we have changed the beginning of Section VI as follows:

Regarding the learning parameters, we initially set $\alpha = 0.5$ and $\epsilon = 1$; then, during the learning process, their values are iteratively reduced to strike a balance between performance and learning speed. We set the discount factor $\gamma = 0.8$ by random search, i.e., by cross-checking the obtained results for different values of the hyper-parameter.

IV. Response to the Comments of the Reviewer 3

(C3.1) For the network state sp, what are the units for r, c and m. For instance, r=1Hz, MHz or GHz? and for m=1 is the unit in Byte, MB or GB? in addition, is it possible to have r=m=c=0? is the state sp= (0,0,0) exist?

(R3.1) Thanks for this remark, which gives us the opportunity to clarify the unity of measure of the available resource variables. In our work we have considered three type of resources, whose availability is described by the variable $\boldsymbol{s_p}(t) = (r, c, m) \in \mathcal{S}_p$. As explained in Section II-A, the unity of measure for the communication resources is [MHz], computing is [GFLOPS/s], and cloud storage is [GB]. To recall this, we have included the unity of measure in the new table, which describes the main notations used in the paper (see Table I of this document).

Also, to improve the paper readability and clarify this aspect, we have modified the first part of the simulation results, as follows:

We assume that each NF needs 40 MHz, 40 GFLOPS/s, and 40 GB in terms of communication, computing, and cloud storage resources, respectively; therefore, the total communication $d_{n,r}$, computing $d_{n,c}$, and cloud storage $d_{n,m}$ demand for a slice is equal to 200 MHz, 200 GFLOPS/s, and 200 GB (see (4)).

During the congestion control phase (see (19)), the minimum amount of resources for running a BE slice i.e., δ_j, $j = \{r, c, m\}$, is equal to 100 MHz, 100 GB, and 100 GFLOPS/s. Overall, we assume that the network has 800 MHz, 800 GFLOPS/s, and 800 GB communication, computing, and cloud storage resources to satisfy the requirements of the deployed slices.

(C3.2) in addition, is it possible to have r=m=c=0? is the state sp= (0,0,0) exist?

(R3.2) Regarding the second question of the reviewer, it is indeed possible to have "r=m=c=0". This scenario occurs when all the resources are allocated to the deployed slices, i.e., temporarily, there are not available resources in the system.

(C3.3) Why the system state has been classified to requested and deployed states and what is the relation between them? Furthermore, why the requests have been considered as part of the system state not as a set of actions?

(R3.3) Thanks for this remark that gives us the opportunity to clarify our problem and the associated system model. Please recall that we are investigating how to optimize the network resource allocation by iteratively selecting the service requests that can be accepted in order to maximize the system revenue while taking into account into account slice priorities, network slice requirements, and resource availability. Therefore, at each time slot, the system state describes the slice requests in the associated queues (i.e., which wait to be deployed) $s_g \in \mathcal{S}_g$ and $s_b \in \mathcal{S}_b$ and the available network resources $\boldsymbol{s_p} \in \mathcal{S}_p$. The system state also indicates the number of deployed slices $u_g \in \mathcal{U}_g$ and $u_b \in \mathcal{U}_b$ and the resources allocated to the running BE slices $\boldsymbol{x_p} \in \mathcal{X}_p$. The action variable models the number of slice requests accepted at each time slot $\boldsymbol{a}(t) = (a_g(t), a_b(t)) \in \mathcal{A}$.

As indicated by the reviewer, these variables are inter-related, i.e., our goal is to find the action to be taken according to the current system state, in order to optimize the long term revenue. Specifically, the action chosen at each time slot must ensure that the number of BE (resp. GS) slice requests accepted does not exceed the actual number of BE (resp. GS) slice requests in the queues (see (17) in the paper). In addition, the constraints in (18) guarantee that, for the accepted BE slices, the sum of allocated resources for each resource type is less than or equal to the current resource availability. Finally, the constraints in (19) ensure that, due to the congestion control, the acceptance of new GS slices does not excessively degrade the quality of service of the running BE slices.

(C3.4) What is the relation between Ng and Sg and also between Nb and Sb? Why different notations have been used?

(R3.4) At each time slot t, $s_g(t) \in \mathcal{S}_g$ and $s_b \in \mathcal{S}_b$ denote the number of slice requests in the GS and BE queues, respectively. During the time slot, according to the available resources, $a_g(t), a_b(t)$ slice requests in the queues are accepted and deployed. Then, new GS and BE requests $n_g(t) \in \mathcal{N}_g = \{0, 1, \dots, N_g\}$ and $n_b(t) \in \mathcal{N}_b = \{0, 1, \dots, N_b\}$ arrive in the system. Since the queues has finite maximum length, a given number of slice requests can be dropped. For instance, as indicated in eq. (21) of the manuscript, the number of GS requests dropped at each time slot can be computed as follows:

$$n_d(t) = \max\{s_g(t) - a_g(t) + n_g(t) - Q_g, 0\},$$

where Q_g is the maximum length of the GS queue.

Finally, the relation between the number of new GS slice requests and the GS slice requests in each queue can be expressed as follows:

$$s_g(t+1) = \min\{s_g(t) - a_g(t) + n_g(t), Q_g\}.$$

Note that in the above equation, $s_g(t+1)$ cannot have a negative value, since $a_g(t) < s_g(t)$ as indicated in (17). The same expressions can be easily derived for the BE slice queue.

(C3.5) What is the total number of the system states?

(R3.5) As described in Section IV-B, the state space size can be computed as follows:

$$|\mathcal{S}| = |\mathcal{S}_g| \cdot |\mathcal{S}_b| \cdot |\mathcal{S}_p| \cdot |\mathcal{U}_g| \cdot |\mathcal{U}_b| \cdot |\mathcal{X}_p|.$$

In our simulations (see Section VI), we have:

- $|\mathcal{S}_g|$=5
- $|\mathcal{S}_b|$=5
- $|\mathcal{U}_g|$=5
- $|\mathcal{U}_b|$=5

In addition, as explained in Section IV-B, when SA is implemented, the amount of resources required by a new slice depends on the similarity with the other slices being in the same NSI (see (14) in the manuscript). Therefore, the sizes of sets of the available resources $\mathcal{S}_p$ and the resources allocated to the BE slices $\mathcal{X}_p$ become very large.

In our simulations, we have considered that this similarity can assume a value in the set $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$, which has length of 6. With these values, the resource requests can assume a value in $\{50, 60, 70, 80, 90, 100, 120, 140, 160, 180, 200\}$ (MHz, GFLOPS/s and GB). Then, the state set of the available resources is $\{0, 10, 20, 30, ..., 800\}$ and the one of the resources allocated to BE slices is $\{0, 50, 60, 70, ..., 800\}$, which have respectively length 81 and 77. Overall, the state space length is $5 * 5 * 81 * 5 * 5 * 77 = 3.898.125$.

(C3.6) Is this system state for one user? or multiple users? and what is the mathematical relation between the system state and the number of users?

(R3.6) Thanks for this question, which enables us to further clarify our problem and the associated system model. In our problem, we focus on the admission and deployment of a service. In this context, the number of users is not considered but only the number and the type of services. In the context of network slicing, each user can be attached to one or multiple slices. However, the resources required by a slice to be deployed are based on the expected peak load of the slice (i.e., the peak number of user). Then, the allocated resources could be adjusted during time depending on the load variations. However, this scheduling function is out of scope of this work.

(C3.7) "It is important to note that, a slice request is dropped when it arrives in a queue that is full". In case the queue was not full while the number of requests is higher than the available slices in the queue, how will the system act in this case?

(R3.7) In this case, the queue will accept as many requests as possible until it is full, following the order of the received requests. For example, if there are 2 requests arriving at the queue and there is only one available slot, only the first request will be included in the queue and the other one will be dropped (see (21) in the manuscript).

(C3.8) As nb and ng start with 0 slices, why 0 slices would be requested? How did the author get equation (6)?

(R3.8) When $n_g(t)$ or $n_b(t) = 0$, there are not additional service requests received in the system in the time slot t. In equation (6) of the manuscript, we consider all the possible number of slice requests to be received in each time slot. Specifically, in a time slot, for each class of service, there could be no request (i.e., 0 request) with probability p_0, or one request arriving with probability p_1, and so on. Note that the total probabilities of all possibilities must be one, implying that one event (receiving 0, 1, 2, or more slice requests) must happen at each a time slot. This is the law of the total probability

(C3.9) How did the author get equation (6)?

(R3.9) Please consider the reply to **(C3.8)**; eq. (6) is just the law of the total probability, one event (receiving 0, 1, 2, or more slice requests) must happen at each a time slot.

(C3.10) Why the action set considers g and b only while does not consider the resources p?

(R3.10) The action set $\mathcal{A}$ in eq. (16) denotes the number of BE and GS slices that can be admitted in the system. The amount of resources allocated to these slices is exactly equal to their resource request $\boldsymbol{T'}_n$ in eq. (15). The amount of available resource $\boldsymbol{s_p}(t)$, however, limits the number of slices that can accepted at each time slot (see (18) and (19)).

(C3.11) Equs. (20) and (21) are not clear. What does it mean in (21) substracting an action from a state? Furthermore, in (20), do the weights change for each state?

(R3.11) First, let us clarify the physical meaning of the notations used in eqs. (20) and (21). $a_g(t)$ is the number of accepted GS slice requests at a time slot t, w_g is the gain obtained by accepting one GS slice request, i.e., the related revenue brought by the slice deployment. Similarly, w_b denote the revenue obtained by accepting each of the $a_b(t)$ BE slice requests admitted at time slot t. Finally, we consider that dropping $n_d(t)$ GS slice requests leads to a cost for the service provider, which is equal to $n_d(t)l_d$, i.e., l_d is the cost for dropping one GS slice request. Therefore, we assume that the weights w_g, w_b, and l_d are constant in the considered problem. Then, $R(\mathbf{s}(t), \mathbf{a}(t))$ in eq. (20) describes an instantaneous revenue perceived by the network when taking action $\mathbf{a}(t)$ in the system state $\mathbf{s}(t)$.

Eq. (21) is related to the evolution of the queue state (see the reply to **(C3.4)**). When we subtract an action to a system state, we compute the amount of slice requests that remains in the queue before receiving the new slice requests $n_g(t)$ and $n_b(t)$.

(C3.12) Present all the values that have been employed to get the results.

(R3.12) Thanks for this remark; all the simulation parameters are presented in Section VI. Also, to improve the paper readability, we have included Table II in Section VI.

(C3.13) At the end, the reviewer believes that the system state, the set of actions and the reward function have not been defined correctly which has a significant impact on the results section.

(R3.13) We would like to thank the reviewer for the constructive comments. We have revised the paper very carefully and we believe that all technical information presented in this paper are described properly. Also, we hope that based on the replies to the reviewer's comments, he/she can agree that our paper is technically sound.

References

[1] O. Adamuz-Hinojosa, J. Ordonez-Lucena, P. Ameigeiras, J. J. Ramos-Munoz, D. Lopez, and J. Folgueira, "Automated network service scaling in nfv: Concepts, mechanisms and scaling workflow," *IEEE Communications Magazine*, vol. 56, no. 7, pp. 162–169, July 2018.
[2] S. Khatibi, K. Shah, and M. Roshdi, "Modelling of computational resources for 5G RAN," in *Proc. EuCNC*, June 2018, pp. 1–5.
[3] NGMN Alliance, "Description of network slicing concept," Jan. 2016. [Online]. Available: https://www.ngmn.org/uploads/media/160113_Network_Slicing_v1_0.pdf
[4] C. Delgado, M. Canales, J. Ortin, J. R. Gallego, A. Redondi, S. Bousnina, and M. Cesana, "Joint Application Admission Control and Network Slicing in Virtual Sensor Networks," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 28–43, Feb. 2018.
[5] P. Caballero, A. Banchs, G. de Veciana, X. Costa-Perez, and A. Azcorra, "Network Slicing for Guaranteed Rate Services: Admission Control and Resource Allocation Games," *IEEE Transactions on Wireless Communications*, vol. 17, no. 19, pp. 6419–6432, Oct. 2018.
[6] 3GPP TSG Services and System Aspects, "TR 28.530, Management and orchestration; Concepts, use cases and requirements," *V15.1.0*, Dec. 2018.
[7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.