Admission Control with Resource Efficiency Using Reinforcement Learning in Beyond-5G Networks

Luis A. Garrido, Kostas Ramantas, Anestis Dalgkitsis
Iquadrat Informatica, S.L.
Barcelona, Spain
{l.garrido, kramantas, a.dalgkitsis}@iquadrat.com

Adlen Ksentini
Eurecom
Sophia Antipolis, France
adlen.ksentini@eurecom.fr

Christos Verikoukis
University of Patras
Patras, Greece
cveri@ceid.upatras.gr

***Abstract*—Managing network slices in 5G networks and in communication technologies Beyond-5G (B5G) requires intelligent mechanisms to ensure users' service access and to maximize the utility and efficiency of the network's physical resources. To achieve this, we propose a mechanism based on Reinforcement Learning (RL) for the Admission Control (AC) of User Service Requests (USRs) into network slices through dynamic bandwidth (BW) reallocation. Our approach admits, delays or rejects USRs into service depending on the BW of the slice and the utility this generates for the infrastructure provider (InP). This approach achieves very low USR rejection rates (RRs) with very high resource efficiency, even when peak traffic loads considerably exceed the BW capacity causing resource scarcity scenarios. When compared against a *static* BW allocation mechanism, our approach achieves RRs that are a fraction (0,33) of those achieved by *static* (smaller RRs are better), with 33,2x less resource overallocation, significantly achieving a very high resource efficiency.**

***Index Terms*—Admission Control, Bandwidth Reallocation, Reinforcement Learning, Network Slicing, B5G**

I. Introduction

Network slicing in 5G, introduced by 3GPP [1], enables the creation of multiple virtual networks, i.e. *network slices*, over the same physical network, which improves resource utilization with more flexibility on the deployment of mobile services [2], [3]. Each slice has its own performance and functional requirements specified by the slice owner [4], [5], or tenant, that rents a subset of the physical resources from the Infrastructure Provider (InP). Multiple network slices can be deployed independently and simultaneously in isolation with security guarantees [2], [6]–[8], allowing dynamic reallocation of resources among them, a property known as elasticity [9].

Given the massive number of slices simultaneously active providing services to an even larger number of users, it is expected that these slices will have dynamic traffic profiles with peak traffic loads exceeding the allocated slice resources or the capacity of the networking elements, causing resource scarcity, i.e. resource pressure [3], [4]. Depending on the frequency of the peak loads and the traffic profile of the slices, the incoming User Service Requests (USRs) won't obtain access into their services simultaneously, highlighting the need of a policy to determine which USRs are either admitted, delayed or rejected into service. Defining this policy is an instance of the Admission Control (AC) problem [9].

On the other hand, if a slice has enough allocated resources to ensure access when peak traffic loads occur, then the slice overallocates resources in a large degree. Since the peaks occur with a given frequency, and the average load is usually much smaller than these peaks, then a slice with overallocated resources is highly inefficient in terms of its resource utilization. This generates an increased Operational Expenditure (OPEX) on the part of the InP, and potentially prevents other USRs from service admission, resulting in further revenue loss.

The trade-off between resource efficiency and maximizing USRs' admission, i.e. reducing rejection rates (RRs), in the presence of dynamic slice traffic profiles with peaks makes the AC problem one of a considerable complexity, making solutions based on Artificial Intelligence (AI) and Machine Learning (ML) an attractive option. This surge in AI/ML integration in 5G to solve AC-like problems has partly ushered in the era of Beyond-5G (B5G) networks [10]. In this paper, we present RL-BAC: an AC solution based on Reinforcement Learning (RL) that maximizes InP utility by minimizing RRs of USRs and maximizing resource efficiency, critical for B5G [11]. Our contributions are summarized as:

- An AC problem formulation as an optimization problem that considers BW reallocation and resource efficiency with respect to peak traffic load at the Radio Access Network (RAN) domain of a B5G network.
- A *tunable* solution based on the Markov Decision Process (MDP) reformulation of the AC optimization problem with a parameter space enriched with B5G network models, resource reallocation and KPI awareness.
- An RL-based solution called RL-BAC for the AC problem that maximizes benefits for the InP, tenants and users, achieving this by reducing RRs of USRs, which is our main KPI, while improving BW utilization efficiency.

This paper is organized as follows. Section II provides background for AC in B5G. Section III explains the system model contextualizing our contributions, which are explained in Section IV. Section V provides the experimental methodology and the baseline used to evaluate RL-BAC. Section VI shows the results, and Section VII concludes this paper.

II. Admission control in Beyond-5G Networks

The AC problem in B5G varies depending on the given implementation of network slicing supported by the communication infrastructure and the targeted scope of optimization [9].

Whether if inter-slice or intra-slice AC is considered [2], [12]–[14], the scope changes from admitting slice deployments by tenants over the network in the former to admitting or rejecting USRs on the latter. Resource limitation plays a significant part in both cases, by limiting the deployment of slices or the admission of USRs, respectively. In this context, RL-BAC exploits dynamic BW reallocation *between* the slices in order to increase the admission rate of USRs or, equivalently, reduce the RRs of the USRs *within* the slice.

There are many mechanisms that target this type of AC problem [3], [15]–[19], many of which rely on some form of AI/ML. However, their formulations and solutions for AC vary significantly. RL-BAC's contributions go beyond these works in many aspects, and in other instances it differs in scope altogether. For example, [16] addresses the problem of users' AC for a URLLC slice while meeting performance constraints. They consider different traffic intensities per user and only one slice. RL-BAC, on the other hand, considers USRs from multiple slices and the resulting AC policy differentiates between slices achieving different AC objectives.

In [17], the authors formulate the AC problem as an ℓ_0 minimization problem on the Signal-to-Noise Ratio (SINR) which depends on the allocated and effective BW of the user. They use approximation methods and sequential convex programming, while RL-BAC uses an RL algorithm for dynamic BW reallocation and formulates the problem as a maximization problem on the *utility* of the InP. In [3], the authors address the resource allocation problem using a discrete-action RL agent (Dueling Deep Q-Networks [20]) and do not consider resource pressure conditions, where resource demand of USRs' traffic flows exceeds the available BW capacity (common situation in B5G), while RL-BAC uses a continuous-action agent for BW reallocation under resource pressure conditions.

III. System Model for RL-BAC

Consider a 5G communication infrastructure with network slicing and a set B of base stations (BSs) in the RAN domain, as shown in Fig. 1. Attached to BS $b \in B$, there's a Multi-Edge Computing (MEC) cloud, that runs the AC solution for dynamic BW reallocation and USR admission, and supports a set of slices S_b providing each a mobile service. At any time t, multiple USRs require service from s demanding BW, a demand that can be predicted for posterior times to t [12].

The USRs of s can be admitted, delayed or rejected into service according to the AC policy in b which, in our case, is driven by the BW resources required by the USRs' traffic flows and the total available link BW allocated to s. Even though many factors can have an effect on USR admission such as delay, noise or channel quality, our model seeks to isolate the effect of dynamic BW reallocation on system-wide KPIs such as RRs and efficient resource utilization.

Optimally, the total BW demand of S_b will match the total BW capacity of b. But due to peaks and the dynamics of USRs' traffic flows, the BW demand may exceed this capacity, making insufficient the BW allocated for some $s \in S_b$. To cope with this, the exceeding flows will be delayed from

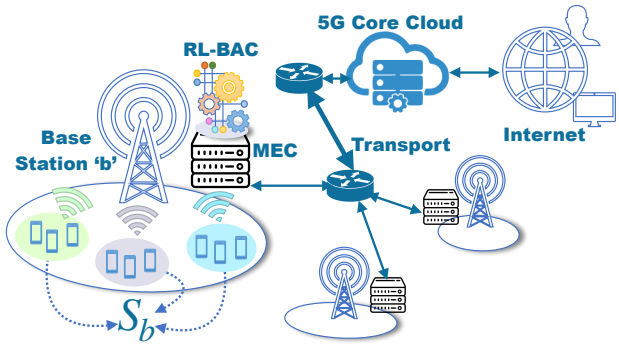


Fig. 1: System Model Under Consideration.

service access, and if b is unable to admit them within a time interval, referred to as *delay window*, they will be rejected. When traffic flows are delayed, the BW allocated to s is shared between the arriving and the delayed traffic flows, increasing the resource pressure in b. When traffic gets delayed/rejected, the InP suffers a *utility* reduction, i.e. penalty, resulting in a revenue loss due to non-compliance with tenants' requirements regarding the access to services of their end users.

On the other hand, if the allocated BW capacity to slice s exceeds the peak traffic load of s, then resource overallocation occurs because the average load is significantly smaller than these peaks, usually half or less. Overallocation is a measure of inefficient resource utilization, and causes a penalty to the InP due to the additional OPEX required for having idle resources not used by any mobile service, thus generating no revenue. Reallocating BW dynamically among slices avoids this condition, while maximizing USR admission across slices.

IV. RL-BAC for Beyond-5G Networks

A. Admission Control and Dynamic BW Reallocation

We formulate the AC problem with a maximization objective on the utility of the InP depending on the BW allocation of slices S_b in BS b. The maximum utility is obtained by allocating the necessary BW to each $s \in S_b$ in order to match its demand in time t, thus maximizing the admission rate across all s, while resource efficiency is obtained when the BW allocated for each s does not significantly exceeds the demand. This prevents service starvation for the USRs of $s \in S_b$. The problem is formulated as an optimization problem defined in (1), (2), (3) and (4).

$$\max U(t) = \sum_{s \in S_b}^{S_b} (W_a^s(t) - W_d^s(t)) * G_{alloc}^s \tag{1}$$

$$\text{s.t.} \quad \sum_{s \in S_b}^{S_b} W_a^s(t) \leq C_b \tag{2}$$

$$Pr(W_a^s(t) < W_d^s(t)) < RR_{th}^s \tag{3}$$

$$\frac{W_a^s(t)}{W_d^s(t)} < z = 1 + \kappa \tag{4}$$

Equation 1 is the InP's utility, which depends on the BW allocated $W_a^s(t)$ and the BW demanded $W_d^s(t)$ by s at time t. The gain factor G_{alloc}^s calibrates the utility that admittance into s generates, depending on the benefits (i.e. revenue) obtained when a USR accesses its services. For example, in a B5G network it is likely that $G_{alloc}^{urllc} > G_{alloc}^{embb}$, which are associated to a Ultra Reliable Low-Latency Communication (URLLC) and a Enhanced Mobile Broad-Band (EMBB) slice, respectively. This ensures that slices with stronger guarantees for USR admission, such as URLLC slices, have their necessities met in relation to other less constrained slices, like EMBBs.

The constraints shown in (2), (3) and (4) denote the conditions in which (1) obtains a satisfactory solution. If either of these conditions fail, then maximization of $U(t)$ won't be possible. Equation 2 defines a limit on the capacity constraint, where the aggregated BW allocated to slices S_b cannot exceed the capacity of BS C_b, disallowing overbooking [21].

To avoid s from reaching service starvation, (3) imposes a constraint on the probability of underallocation given by $Pr(W_a^s(t) < W_d^s(t))$, which is used as a proxy for the probability of USR rejection: whenever s has less resources than those demanded at time t, USRs for s will be delayed or rejected, if they exceed the delay window. The probability of rejection is the RR, and has a threshold value RR_{th}^s, usually agreed upon by the InPs and tenants through Service Level Agreements (SLAs) or other means.

The constraint on (4) imposes a limit, given by z, on the BW overallocation of a slice s with respect to its demand and prevents $U(t)$ from growing naively through excessive BW overallocation. Thus, z is defined as $1 + \kappa$ to prevent an increase on reject/delay rates and excessive resource overallocation (resulting in idle resources, reducing resource efficiency), with a minimized value of κ. BW overallocation for one slice may result in resource starvation by other slices making them unable to comply with the constraint of (3).

B. The Admission Control Problem as an MDP

The AC problem just defined can be re-formulated into an RL problem by describing the environment with which an RL agent will interact with and perform actions on, and defining the agent itself. The environment is formulated as an MDP consisting of the tuple (S, A, P, R, γ). In our formulation, S denotes the environment's state space, assumed to be continuous. A denotes a continuous action space specifying the primitives used by the agent to interact with the environment. The transition function $P : S \times A \times S \to [0, \infty]$ defines the probability distribution of transitions from state s_t in time t to another state $s_{t+T_p} \in S$ in response to an action a_t, where T_p is the duration of a control cycle.

The reward function R generates an instantaneous value $r : S \times A \to [r_{minx}, r_{max}]$ in response to a_t in state s_t, giving the agent an indication of how beneficial the action was. The agent seeks to maximize the discounted reward $R_{dc} = \sum_{t=n}^{n+T} \gamma^t * r(s_t, a_t)$, where T is the length of the horizon, usually corresponding to the agent's training and inference episode, and $0 \leq \gamma \leq 1$. For $\gamma = 0$, the agent prioritizes obtaining the best reward in time t, while for $\gamma = 1$ the agent prioritize accumulated rewards over the episode.

Based on this, we can define the MDP for our AC problem:

1) State Space S: We define the state space as consisting of: 1) C_b, 2) cross-slice delay rate (DR), 2) cross-slice RR, 2) DR for each $s \in S_b$, 3) RR for each s, 4) the aggregated traffic of the USRs of each slice, i.e. BW demand of s, 5) the total delayed USRs for each s and, optionally, 6) BW demand forecasts of each s. The predictions are obtained from a traffic predictor enhanced with 5G network and resource management models [22], and using them as part of the state space increases the convergence of RL-BAC for different experiments in which traffic flows have different profiles.

2) Action Space A: The agent allocates BW to each slice for the USRs in time t, and for the delayed USRs within the delay window. Thus, the action space has two actions per slice: BW allocation for current USRs, and BW allocation for delayed USRs. The total BW allocated to a slice in time t can be defined as $W_a^s(t) = W_{a-current}^s(t) + W_{a-delayed}^s(t)$.

3) Reward Function: The R consists of two components:

- A quadratic component modeling a utility function shown in (5), where $x^s(t) = W_a^s(t) - W_d^s(t)$, establishing a relationship between the demanded and allocated BW of $s \in S_b$, accounting for (1), (2) and (4).
- Two piece-wise functions with linear components as negative utility functions, i.e. penalties, shown in (6) and (7), generating penalties when the RRs and DRs of s exceed their thresholds that account for (3).

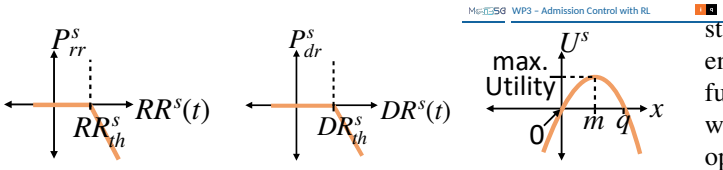
$$U^s(t) = \frac{G_{alloc}^s * (x^s(t))^2}{m * (m - q)} + \frac{G_{alloc}^s * x^s(t)}{m * (q - m)} \tag{5}$$

$$P_{rr}^s(t) = \begin{cases} 0 & RR^s(t) \leq RR_{th}^s \\ G_{rr}^s * (RR_{th}^s - RR^s(t)) & RR^s(t) > RR_{th}^s \end{cases} \tag{6}$$

$$P_{dr}^s(t) = \begin{cases} 0 & DR^s(t) \leq DR_{th}^s \\ G_{dr}^s * (DR_{th}^s - DR^s(t)) & DR^s(t) > DR_{th}^s \end{cases} \tag{7}$$

The plots generated by (5), (6) and (7) are shown in Fig. 2. The quadratic component given by U^s, represents the overall utility a slice $s \in S_b$ achieves when allocating enough BW to fit the demand, reducing the amount of USRs that get delayed or rejected. This component generates a negative utility ($U^s(t) < 0$) for s when $x^s(t) < 0$, inferring the condition $W_a^s(t) < W_d^s(t)$ that reduces USR admission. However, U^s will become larger or equal to zero if $W_d^s(t) \leq W_a^s(t) \leq W_d^s(t)+q$, where $q > m > 0$. Thus, the agent favours a degree of BW overallocation for s as long as $0 \leq W_a^s(t) - W_d^s(t) \leq q$, but when the difference $W_a^s(t) - W_d^s(t)$ increases beyond q, then $U^s(t)$ becomes negative, i.e. generates penalties for s, preventing excessive BW overallocation for s.

The parameters $m > 0$ and $q > 0$ in (5) can be tuned in order calibrate the tolerance of the agent with regards to BW overallocation. Likewise, the parameter G_{alloc}^s can be used to



(a) P^s_{rr} from (6). (b) P^s_{dr} from (7). (c) U^s from (5).

Fig. 2: Plots for P^s_{rr}, P^s_{dr} and U^s

calibrate the agent in order to favor allocation for a slice s within the range in which $W^s_d(t) \leq W^s_a(t) \leq W^s_d(t) + q$.

Even though (5) provides a representative model for the AC problem in (1)-(4), it is also necessary to impose additional constraints on the agent in order to satisfy the admission rate requirements of the slice. For this reason, (6) and (7) generate additional penalties if the RR threshold RR^s_{th} and the DR threshold DR^s_{th} values are not met. These values are configurable and can differ from slice to slice, making it possible to enforce a BW allocation preference for certain slices with more demanding requirements, resulting in overall greater benefits for the InP. The resulting reward function $R_s(t)$ associated to s is given by (8).

$$R^s(t) = U^s(t) + P^s_{rr}(t) + P^s_{dr}(t) \tag{8}$$

The total reward $R(t)$ is given by the sum of the rewards from all the network slices $s \in S_b$, shown in (9).

$$R(t) = \sum_{s \in S_b}^{S_b} R^s(t) \tag{9}$$

C. RL Agent for the MDP

Now it is necessary to define an RL algorithm, i.e. agent, to solve the MDP with continuous state and action spaces, making the Soft-Actor Critic (SAC) algorithm [23] a suitable alternative. Using continuous-action algorithms avoids the combinatorial explosion problem [24] introduced by the discretization of states and actions, increasing the probability and speed of convergence of RL-BAC.

The SAC algorithm finds a policy $\pi(a_t|s_t)$, i.e. a state-to-action mapping, that maximizes the sum of rewards with maximum entropy given by $J(\pi) = \sum_{t=n}^{t=n+T} \mathbb{E}_{(s_t,a_t)}[r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))]$, in which α controls the stochasticity of the optimal policy and its importance with respect to the rewards. The SAC algorithm finds this policy through an iteration process consisting of a policy evaluation step and a policy improvement step. In the evaluation step, a soft Q-function $Q(s_t,a_t) \leftarrow J(\pi) + \gamma\mathbb{E}_{s_{t+1}}[V(s_{t+1})]$ is evaluated, where $V(s_t) = \mathbb{E}_{a_t\sim\pi}[Q(s_t,a_t) - log\ \pi(a_t|s_t)]$. By defining a modified Bellman backup operator [23] as $\mathcal{T}^\pi$, a value sequence is obtained in which each value is given by $Q^{k+1} = \mathcal{T}^\pi Q^k$.

In the policy improvement step, π is updated towards the exponential of each new value of the soft Q-function. Every time the policy is updated, SAC guarantees that $Q^{\pi_{new}}(s_t,a_t) \geq Q^{\pi_{old}}(s_t,a_t)$. Updating π and $Q(s_t,a_t)$ in this way will yield an optimal policy, but this algorithm can only be exactly executed in its tabular form. When considering continuous state and action spaces (where tabular representations are not entirely feasible), then Q, V and π are expressed through function approximators implemented using deep neural networks (DNNs). In this case, the process alternates between optimizing the networks with stochastic gradient descent.

V. Experimental Framework

RL-BAC, a simulation of the system model described in Section III and the MDP environment were all implemented in Python 3.8. Likewise, the SAC algorithm was implemented using Python 3.8 with Tensorflow 2.1 [26].

A. Dataset Used For Experiments

RL-BAC was trained using mobile traffic data from the city of Milano [25], assuming the infrastructure model from Section III. We assume that each entry in the dataset corresponds to the aggregated traffic flow of the USRs connected to a BS, and we assume that each traffic flow is independent from each other. We don't consider persistent connections by USRs across multiple entries. The dataset has traffic data for three different service types: *calls*, *SMS* and *internet*, which we associate to separate network slices. Our traffic predictions are also on a per-slice basis. RL-BAC was trained over this traffic trace with the system model in Section III, consequently evaluating the RRs, DRs and the overallocation magnitudes.

RL-BAC was compared against a baseline algorithm that distributed C_b equally amongst the network slices. This algorithm, referred to as *static*, distributes the slice BW amongst its USRs in time t if there are no delayed USRs. If the slice has delayed USRs, *static* shares the slice BW between the delayed and current USRs. The delay window of USRs' traffic flows is of 60 seconds for both RL-BAC and *static*. Since the available dataset only has entries that are 10 minutes apart, traffic flows in the delay window where interpolated, as well as their prediction. Multiple interpolation methods were evaluated (constant, linear, spline), but linear interpolation proved good enough given the relative small delay window size.

B. Parameters for the MDP and SAC

The state space and action space for SAC has 18 and 6 variables, respectively. Table I summarizes the parameters used for the MDP, the associated reward function parameters, and for the SAC algorithm. The small values chosen for m and q in (5), where $q > m$, favor overallocation to a very small degree, and generate negative utilities for BW underallocation. By setting $q = 1.0$ in (5), κ in (4) becomes $\frac{1}{W^s_d(t)}$. If the overallocation goes beyond $z = 1 + \frac{1}{W^s_d(t)}$, then the SAC algorithm will sample a negative utility value, i.e. a penalty.

The values for G^s_{alloc}, G^s_{rr} and G^s_{dr} are large compared to m and q. After an extensive parameter space exploration, the chosen values yielded significantly better results when compared to *static*. The differences between these values allows the InP to tune RL-BAC by giving it a bias amongst the slices, in order to favor BW allocation (and thus higher USR admission) for slices that generate higher utilities (a URLLC slice, for example), as explained in Section IV-B3.

TABLE I: Parameters for the 5G Network Model and SAC.

Parameters for the 5G Network Model (MDP)		
	Parameter	Value
For all slices		
	m (5)	0.5
	q (5)	1.0
For s=sms		
	G^s_{alloc}	500.0
	G^s_{rr}	11000.0
	G^s_{dr}	3000.0
For s=internet		
	G^s_{alloc}	600.0
	G^s_{rr}	13000.0
	G^s_{dr}	4000.0
For s=calls		
	G^s_{alloc}	700.0
	G^s_{dr}	15000.0
	G^s_{dr}	5000.0

Parameters for SAC	
Parameter	Value
γ	0.90
α	0.90
Num. of Layers	8
Layer Size	64
Activation Functions	ReLu
Replay Buffer Size	30720
τ	0.995
Batch Size	256
Optimizer	Adam
Learning Rate	0.0002
Episodes	200

VI. Results and Evaluation

Fig. 3 shows a normalized comparison of the RRs of RL-BAC with respect to *static*. This figure shows their relative RRs for different ratios of the Peak Traffic Load (P_{TL}) to the total BW capacity of the BS (C_b), referred to as $R = \frac{P_{TL}}{C_b}$, which is a measure of resource pressure on the BS that increases with R. The values on Fig. 3 are the average RRs of the last 1000 steps of each experiment, and it shows that RL-BAC achieves RRs that are consistently lower than those of *static*, with the smallest being 0,33x times that of *static* for $R = 1,17$. In $R = 1,0$, corresponding to $P_{TL} = C_b$, *static* generates a smaller RR than RL-BAC, but this is the case where there's no resource pressure on the BS, in which the total resource demand of all slices fits in C_b. However, this comes at the expense of very high overallocation, i.e. resource utilization inefficiency, by *static*, implying a high OPEX (see Fig. 6).

As the values of R increase from $1,0$ to $10,0$, RL-BAC always yields a smaller RR than *static* because RL-BAC makes better use of the available BW considering that the average load of the slices in the BS is around a third of the P_{TL}. This allows RL-BAC to reallocate BW from slices who are not experiencing peak traffic loads towards those facing resource pressure, since the slices don't peak all simultaneously.

Fig. 4 shows the normalized DRs for both algorithms (comparable in magnitude as the RRs), showing the rate at which USRs have been delayed from service due to BW

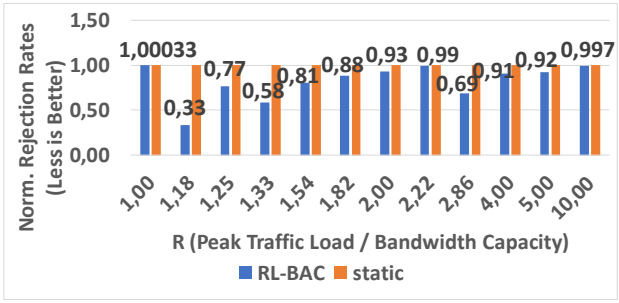


Fig. 3: Normalized Rejection Rates for RL-BAC and *static*.

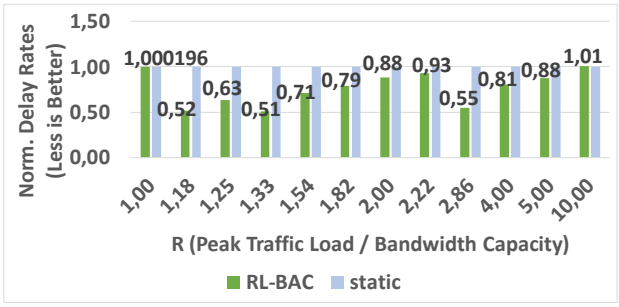
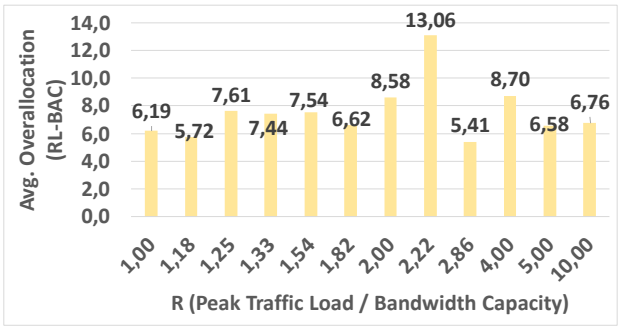
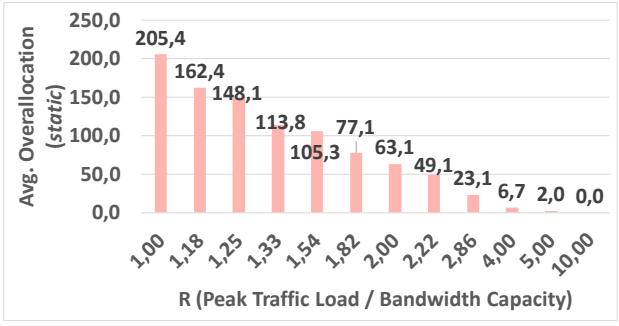


Fig. 4: Normalized Delay Rates for RL-BAC and *static*.



(a) Average Overallocation for RL-BAC (less is better).



(b) Average Overallocation for *static* (less is better).

Fig. 5: Average Overallocations for RL-BAC and *static*.

resource pressure, and it demonstrates the ability of RL-BAC to achieve DRs smaller than *static*, except when $R = 1,0$ and $R = 10,0$. In these two cases, RL-BAC has a DR that is 0,0196% and 1,1% higher than *static*. However, this lesser DRs by *static* are obtained, once again, at the expense of a large amount of BW overallocation for the slices (see Fig. 5b).

Fig. 5 shows that RL-BAC achieves considerable smaller overallocation for $R < 4,0$, while simultaneously pushing

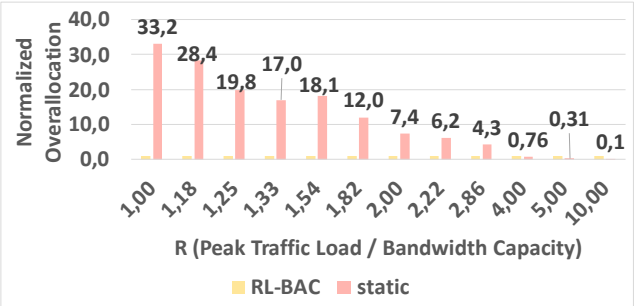


Fig. 6: Normalized overallocation for RL-BAC and *static*.

down RRs and DRs. This small overallocation is explained by observing the reward function of RL-BAC (Section IV-B3). This function was designed to favor a degree of overallocation, in order to account for the fluctuation-prone dynamicity of the USRs' traffic flows. After an extensive exploration over the parameters of this reward function, the values used for m and q yielded RRs considerably smaller than *static* while keeping the average overallocation to the small values shown in Fig. 5a

Notice that RL-BAC allocates *consistently* and *efficiently* the BW amongst the slices, even under resource pressure ($R \geq 4,0$), resulting in the consistent RRs/DRs that RL-BAC achieves. On the other hand, Fig. 5b shows a decreasing trend for the overallocation of *static*, and when $R \geq 4,0$, *static* reaches smaller overallocation than RL-BAC, but with larger RRs/DRs. Fig. 6 shows the normalized overallocation averages for *static* with respect to RL-BAC, showing how *static* overallocates as much as 33,2 times the necessary BW required by RL-BAC to achieve a comparable RR, demonstrating the resource inefficiency of *static*. For $R \geq 4,0$, RL-BAC overallocates more than *static* under such resource pressure, but *static* only allocates BW to current USRs ineffectively (thus the higher RRs and DRs for *static* in Fig. 3 and 4, respectively), while RL-BAC keeps the RRs/DRs lower.

VII. Conclusions

We showed that RL-BAC improves RRs and DRs significantly, by reducing them to 0,33 and 0,52 times, respectively, with 33,2x times less overallocation when comparing to *static*. This promising result shows the ability of RL-BAC to dynamically reallocate BW satisfactorily in B5G networks with network slicing, increasing the benefits of InP and guaranteeing users' admission. For future work, RL-BAC could be enhanced to a federated version, and the reward function modified to further exploit traffic forecasting.

VIII. Acknowledgements

This work was supported by research projects 5GMediaHUB (101016714), 5G-EPICENTRE (101016521) and ADROIT6G (101095363).

References

[1] 3GPP TS 23.501 v15.2.0 Release 15, "5G; System Architecture for the 5G System" (2018-06)

[2] S. Troia et. al., "Admission Control and Virtual Network Embedding in 5G Networks: A Deep Reinforcement-Learning Approach," in IEEE Access, vol. 10, pp. 15860-15875, 2022.

[3] G. Sun et. al., "Resource slicing and customization in RAN with dueling deep Q-Network", Journal of Network and Computer Applications, Volume 157, 2020.

[4] K. Suh, S. Kim, Y. Ahn, S. Kim, H. Ju and B. Shim, "Deep Reinforcement Learning-Based Network Slicing for Beyond 5G," in IEEE Access, vol. 10, pp. 7384-7395, 2022.

[5] Y. Prathyusha and T. -L. Sheu, "Coordinated Resource Allocations for eMBB and URLLC in 5G Communication Networks," in IEEE Trans. on Vehicular Technology, vol. 71, no. 8, pp. 8717-8728, Aug. 2022.

[6] D. Loghin et al., "The Disruptions of 5G on Data-Driven Technologies and Applications," in IEEE Trans. on Knowledge and Data Engineering, vol. 32, no. 6, pp. 1179-1198, 1 June 2020.

[7] D. Sattar and A. Matrawy. "Towards Secure Slicing: Using Slice Isolation to Mitigate DDoS Attacks on 5G Core Network Slices." IEEE Conf. on Communications and Network Security (CNS) (2019): 82-90.

[8] B. Khodapanah et al. "Fulfillment of Service Level Agreements via Slice-Aware Radio Resource Management in 5G Networks," IEEE 87th Vehicular Technology Conf. (VTC Spring), Porto, 2018, pp. 1-6.

[9] M. O. Ojijo and O. E. Falowo, "A Survey on Slice Admission Control Strategies and Optimization Schemes in 5G Network," in IEEE Access, vol. 8, pp. 14977-14990, 2020.

[10] K. Samdanis and T. Taleb, "The Road beyond 5G: A Vision and Insight of the Key Technologies," in IEEE Network, vol. 34, no. 2, pp. 135-141. March/April 2020.

[11] Y. Sun et. al., "Service provisioning framework for RAN slicing: User admissibility slice association and bandwidth allocation", IEEE Trans. Mobile Comput., vol. 20, no. 12, pp. 3409-3422, Dec. 2021.

[12] W. Jiang et. al., "Probabilistic-Forecasting-Based Admission Control for Network Slicing in Software-Defined Networks," in IEEE Internet of Things Journal, vol. 9, no. 15, pp. 14030-14047, 1 Aug.1, 2022.

[13] S. Vassilaras et. al., "The Algorithmic Aspects of Network Slicing," in IEEE Comms. Magazine, vol. 55, no. 8, pp. 112-119, Aug. 2017.

[14] W. Ben-Ameur, L. Cano and T. Chahed, "A framework for joint admission control, resource allocation and pricing for network slicing in 5G," 2021 IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 2021, pp. 1-6.

[15] T. V. K. Buyakar et. al., "Resource Allocation with Admission Control for GBR and Delay QoS in 5G Network Slices," 2020 Intl. Conf. on COMmunication Systems & NETworkS (COMSNETS), Bengaluru, India, 2020, pp. 213-220.

[16] F. Mehmeti and T. F. La Porta, "Admission Control for URLLC Users in 5G Networks". In Proc. of the 24th Intl. ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '21). Association for Computing Machinery, New York, NY, USA, 199–206.

[17] N. U. Ginige et. al., "Admission Control in 5G Networks for the Coexistence of eMBB-URLLC Users," 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), Antwerp, Belgium, 2020, pp. 1-6.

[18] Y. Lu et. al., "Research on User Access Selection Mechanism Based on Maximum Throughput for 5G Network Slicing," 2021 International Conference on Computer Communications and Networks (ICCCN), Athens, Greece, 2021, pp. 1-8.

[19] A. T. Ajibare and O. E. Falowo, "Resource Allocation and Admission Control Strategy for 5G Networks Using Slices and Users Priorities," 2019 IEEE AFRICON, Accra, Ghana, 2019, pp. 1-6.

[20] Z. Wang et. al., "Dueling Network Architectures for Deep Reinforcement Learning", in Proc. of The 33rd Intl. Conf. on Machine Learning, 2016.

[21] S. Saxena and K. M. Sivalingam, "DRL-Based Slice Admission Using Overbooking in 5G Networks," in IEEE Open Journal of the Communications Society, vol. 4, pp. 29-45, 2023.

[22] L. A. Garrido et. al., "Context-Aware Traffic Prediction: Loss Function Formulation for Predicting Traffic in 5G Networks," ICC 2021 - IEEE Intl. Conf. on Communications. Montreal, Canada, 2021, pp. 1-6.

[23] T. Haarnoja et. al., "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor." Paper presented at the meeting of the ICML, 2018.

[24] O. Iacoboaiea et. al., "Real-Time Channel Management in WLANs: Deep Reinforcement Learning versus Heuristics," 2021 IFIP Networking Conference, Espoo and Helsinki, Finland, 2021, pp. 1-9.

[25] Telecom Italia, 2015, "Telecommunications - SMS, Call, Internet - MI", https://doi.org/10.7910/DVN/EGZHFV, Harvard Dataverse, V1

[26] M. Abadi et. al., "TensorFlow: Large-scale machine learning on heterogeneous systems", 2015. Software available from tensorflow.org.