Guaranteeing User Rates with Reinforcement Learning in 5G Radio Access Networks

Ioan-Sorin Comșa[1], Sijing Zhang[2], Mehmet Emin Aydin[3], Pierre Kuonen[4], Ramona Trestian[5] and Gheorghiță Ghinea[1]

[1]*Brunel University London, United Kingdom*

[2]*University of Bedfordshire, United Kingdom*

[3]*University of West of England, United Kingdom*

[4]*University of Applied Sciences of Western Switzerland, Switzerland*

[5]*Middlesex University London, United Kingdom*

ABSTRACT

The user experience constitutes an important quality metric when delivering high-definition video services in wireless networks. Failing to provide these services within requested data rates, the user perceived quality is strongly degraded. On the radio interface, the packet scheduler is the key entity designed to satisfy the users' data rates requirements. In this chapter, a novel scheduler is proposed to guarantee the bit rate requirements for different types of services. However, the existing scheduling schemes satisfy the user rate requirements only at some extent because of their inflexibility to adapt for a variety of traffic and network conditions. In this sense, we propose an innovative framework able to select each time the most appropriate scheduling scheme. This framework makes use of Reinforcement Learning and neural network approximations to learn over time the scheduler type to be applied on each momentary state. The simulation results show the effectiveness of the proposed techniques for a variety of data rates' requirements and network conditions.

Keywords: Packet Scheduling, Radio Resource Management, Reinforcement Learning, Neural Networks.

1. INTRODUCTION

The accelerated acquisition of powerful mobile devices is significantly contributing to the growing market of immersive multimedia applications. According to Cisco (2017), it is envisioned that by 2012 more than 80% of total mobile data will be represented by video traffic at different data rate requirements. In this context, the end-user Quality of Experience (QoE) will make the difference between network operators while providing these pretentious services (Trestian, Comsa, and Tuysuz, 2018). According to Ghinea, Timmerer, Lin, and Gulliver (2014), the concept of Multiple Sensorial Media (Mulsemedia) can

enhance the user perceived QoE when experiencing poor video quality by incorporating additional senses such as: olfaction, wind, haptic, etc. But the real factor that impacts the video quality degradation is denoted by the QoS provisioning schemes on the wireless interface that can differ from one operator to another. By providing higher video rates than the requested limit, the rate of packet drops is increased in order to keep the normal functionality of video decoders. On the other side, lower data rates for video services will increase the packet delays which have as a consequence a larger number of lost packets at the radio interface. Thus, guaranteeing certain data rates for video traffic is crucial in order to avoid the degradation of user perceived quality.

In 5th Generation (5G) of mobile communications standard, guaranteeing certain bit rate requirements is even stricter since the new bandwidth hungry applications (i.e. high definition video, virtual reality traffic) are generated (Elbamby, Perfecto, Bennis, and Doppler, 2018). This puts a significant pressure on Radio Resource Management (RRM) to provide these immersive services with very stringent QoS in multi-user scenarios (Li et al. 2017). Alongside of other RRM functions, the packet scheduler is in charge of allocating user data packets in frequency domain at each predefined Transmission Time Intervals (TTIs). According to Comşa (2014a), the scheduling process is conducted based on the scheduling rules aiming to maximize the satisfaction of particular QoS requirements. In literature, several scheduling rules are proposed to deal with the Guaranteed Bit Rate (GBR) objective. For example the scheduling rule proposed by Lundevall et al. (2004) is designed to work for WCDMA access networks and very low data rates of video services. Andrews, Qian, and Stolyar (2005) propose a scheduler for CDMA downlink networks in which a maximum number of 40 users are scheduled with the maximum rate of 160kbps. In the same type of access networks, the scheduler proposed by Kolding (2006) outperforms other GBR oriented schedulers for the considered networking scenarios. However, the proposed schedulers work appropriate only for particular scheduler states in terms of: access technologies, user rates, channel conditions, traffic load, etc. On one hand, these scheduling techniques must be upgraded for the novel access technologies imposed by 5G standard. On the other hand, the aim would be to use each of these scheduling rules on the best matching scheduler state in order to maximize over time the satisfaction of user rate requirements for various traffic types.

The proposed scheduling optimization problem must determine at each TTI the radio resources to be selected for active users as well as the most convenient scheduling rule to be followed in order to get the maximum possible GBR satisfaction outcome according to the scheduler momentary states. According to the selected scheduling rule, the resource allocation performs the frequency prioritization by favoring those users with poorer GBR satisfaction profile. If this frequency prioritization can be easily computed each TTI by simply calculating given scheduling metric for each user and radio resource, the scheduling rule selection must be a priori decided based on the momentary scheduler states. The idea is learn over time some preference values by applying random scheduling rules in many visits of scheduler states. But for the GBR-based maximization problem, this solution is unfeasible since the scheduler state space is continuous and multidimensional and hence, action-state values cannot be enumerated exhaustively. Thus, the only way is only to approximate the best scheduling rule selection on each scheduler state. The aim is to learn these approximations in such a way that eventually, these solutions of the scheduling optimization problem would converge to optimal solutions. These approximations are represented by non-linear functions or neural networks whose weights are refined and adapted each TTI according to the selected scheduling rule. According to Van Hasselt (2011), the Reinforcement Learning (RL) principles and algorithms can be used to update these non-linear functions by reinforcing for each action function the corresponding time-based error.

This chapter proposes an innovative scheduler framework that aims to improve the user QoE of video services by maximizing the satisfaction of users' rates requirements under three different traffic type scenarios, such as: full buffer, Constant Bit Rate (CBR) and Variable Bit Rate (VBR). A new scheduling rule is proposed to maximize the satisfaction of GBR requirements. Then, the proposed scheduling

framework makes use of RL algorithms to learn over time the policy of scheduling rules to be applied on each state. We study the impact of five actor-critic RL algorithms on the aforementioned traffic types with various channel and networking conditions. The scheduling process is conducted in Orthogonal Frequency Division Multiple Access (OFDMA) downlink systems. OFDMA is one of the access scheme candidates in 5G radio networks as proposed by FANTASTIG-5G (2016) European project.

2. RELATED WORK

The user QoE degradation of multimedia applications can be partially masked by introducing the mulsemedia services as stated by Yuan, Chen, Ghinea and Muntean (2014). According to Ghinea and Ademoye (2012), by adding olfaction sensorial components to the conventional video content, the user perceived experienced can be strongly improved if the perceived olfactory synchronization (Ademoye and Ghinea, 2009) is considered. Also, for some particular video content enhanced with olfaction sensory effects, the audio stream can be masked as suggested by Ademoye, Murray, and Ghinea (2016). An adaptation technique for the mulsemedia content based on the available bandwidth is proposed by Yuan, Ghinea and Muntean (2015). Comşa, Trestian and Ghinea (2018a) propose the concept of 360˚ mulsemedia in which, the QoE degradation of 360˚ video content due to fast head movements or poor network conditions can be masked if additional 360˚ sensory objects (i.e. wind, olfaction, heat) are perfectly mapped with the conventional content in both spatial and temporal domains. In order to satisfy the stringent QoS requirements of bandwidth-hungry 360˚ mulsemedia applications in multi-user scenario, a novel 5G scheduler framework is proposed by Comşa et al. (2018a) that makes use of the reinforcement learning.

To maximize the QoS satisfaction and implicitly the QoE provisioning of real time applications, Reinforcement Learning is used to learn over time the most convenient scheduling rule on each momentary state (as initially proposed by Comşa, Aydin, Zhang, Kuonen, and Wagen, 2011; Comşa, Aydin, Zhang, Kuonen, and Wagen, 2012). The exploited scheduling rules are focusing on satisfying various objectives, such as: user fairness satisfaction, GBR, delay and packet loss rates. The parameterization of Proportional Fair (PF) scheduling rule is used to attain dynamic trade-off between system throughput maximization and user fairness satisfaction according to the network conditions (Comşa, Zhang, Aydin, Kuonen, and Wagen, 2012; Comşa et al., 2014b, Comşa et al., 2014c). Neural networks are used to approximate the best parameterizations of Generalized PF (GPF) scheduling rule at each momentary state. Comşa et al. (2014b) use actor-critic RL algorithms to perform the simple parameterization of GPF rule. Various RL algorithms are used to update the weights of neural networks: some RL algorithms use fixed and discrete steps for GPF simple parameterization and update the neural networks for each step; another RL algorithm entitled continuous actor-critic RL algorithm makes use of only one neural network to output the continuous values that parameterize GPF rule at each TTI. When comparing with other state-of-the-art approaches, the learnt functions with these RL algorithms show gains higher than 10% when matching the user rates against the fairness satisfaction criterion. Moreover, Comşa et al. (2014c) use the same continuous actor-critic RL scheme for the double parameterization of the GPF scheduling rule by adapting the neural network that provides two continuous values at each TTI. The results indicate a gain larger than 5% when compared to actor-critic RL framework that uses the simple parameterization. Also, the double parameterization framework reveals a much higher capacity of recovering from unfeasible state space regions where the user fairness criterion is not satisfied.

Scheduling frameworks that use RL algorithms are also exploited to optimize the QoS objectives concomitantly. For instance, Comşa, De Domenico, and Ktenas (2017) proposes a 5G scheduler able to optimize the satisfaction of GBR, PLR and delay requirements at the same time for different types of traffic classes. The proposed actor-critic RL framework use a pool of three scheduling rule, each of them being oriented on one of the aforementioned particular objectives. When compared with simple

scheduling rules, the proposed RL framework shows gains higher than 10% for different traffic types by monitoring the time when all users are satisfied from the viewpoint of all three objectives. In another work, Comşa, Trestian and Ghinea (2018b) optimize the satisfaction of both packet loss and delay objectives by using various RL algorithms. Different settings for the time windows are considered when computing the online packet loss rates for each user. Larger time windows may involve higher packet loss rates while shorter ones will decrease the probability of losing more data packets. Under various network conditions and time window settings, the learnt scheduling policies are able to outperform the standard scheduling rules when monitoring the number of TTIs with satisfied users from the viewpoint of packet loss and delay requirements.

This chapter considers the satisfaction of GBR requirements for different types of applications for downlink OFDMA scheduling. A variety of RL algorithms is considered to learn the best policy of scheduling rule for different settings of time windows that are used to compute the average user rates. Each RL consider a neural network for each scheduling rule from the pool. We aim to learn over time the best set of non-linear functions that can provide the selection of the best scheduling rule in each momentary state. The rest of this chapter is organized as follows: Section 3 presents the scheduling system model; Section 4 details the proposed RL framework with neural network approximations; Section 5 presents the simulation results for different traffic classes and network settings. Finally, Section 6 concludes this chapter.

3. GBR BASED SCHEDULING MODEL

In this chapter we consider OFDMA downlink scheduling in which the available bandwidth is divided in equal Resource Blocks (RBs) at each TTI *t*. A RB is the smallest resource unit that can be allocated to one user based on the scheduling procedure. Then, we define by $\mathcal{B}=\{1,2,...,B\}$ the set of RBs corresponding to a given system bandwidth where B is the maximum number of RBs. Also, we define the User Equipment (UE) being defined by homogeneous traffic such as: full buffer, CBR and VBR. We denote by $\mathcal{I}_t=\{1,2,...,I_t\}$ the set of active users at TTI t where I_t is the maximum number of users. The set of active users $\mathcal{I}_t$ is time dependent since the total number of users I_t may change from one TTI to another.

The main role of the packet scheduler is to allocate each RB $j\in\mathcal{B}$ to user $i\in\mathcal{I}_t$ such that the satisfaction of GBR requirements is maximized. We define by $\underline{T_i}$ the GBR requirement for the average throughput of user $i\in\mathcal{I}_t$ characterized by a given traffic type. The GBR objective is satisfied for user $i\in\mathcal{I}_t$ at TTI t only if the average user throughput is greater than the GBR requirement $\underline{T_i}$. According to Comşa (2014a), the average user throughput can be calculated by using two types of filters, Exponential Moving Filter (EMF) and Median Moving Filter (MMF):

- EMF based average user throughput : we define by $\overline{T}_i[t]$ the average user throughput of user $i\in\mathcal{I}_t$ and calculated according to:
$$\overline{T}_i[t]=(1-\beta)\cdot\overline{T}_i[t-1]+\beta\cdot T_i[t] \quad (1)$$
where β is the forgetting factor. When β takes low values, then the impact of new value throughput $T_i[t]$ is very low when computing its average value. On the other side, when β increases, then high oscillations are introduced when the scheduling performance is analyzed.
- MMF based average user throughput: the instantaneous user throughputs are stored for a given time window W . Then the computation of average user throughput $\overline{\overline{T}}_i[t]$ is determined each TTI according to the following formula:

$$\overline{\overline{T}}_i[t] = \frac{1}{W}\sum_{x=0}^{W} T_i[t-x] \quad (2)$$

When the time window W is high, the average user throughput is calculated based on the large number of instantaneous throughputs. In this case, the convergence of average user throughput $\overline{\overline{T}}_i[t]$ to its requirement $\underline{T_i}$ may take longer time to be achieved and the RRM outcome in terms of GBR satisfaction depends on very large number of observations. For very restrictive lengths of time windows, only some previous instantaneous throughputs are considered and then, the process can get significant oscillations and the RRM outcome in terms of GBR satisfaction is noisy. For this reasons, W must be carefully chosen in order to quantify best the scheduling rule applied on each state. In this sense, we propose to calculate the median moving window W as a function that depends on the number of active users I_t and the maximum number of schedulable users I_{max} such as:

$$W = \rho \cdot [I_t / I_{\max}] \quad (3)$$

where $I_{\max}$ represents the maximum number of users that can be scheduled at each TTI based on the signaling overhead constraints and system bandwidth and $\rho \in \mathbb{R}^+$ is the windowing factor.

According to some studies conducted by Comşa (2014a), the average user rates $\overline{T}_i[t]$ computed based on the exponential filter are used to compensate the channel fluctuations in OFDMA downlink scheduling, whereas the MMF based average user throughputs are used to measure the impact of each applied scheduling rule from the perspective of GBR objective. Thus, the setting of the windowing factor ρ plays a crucial role. When ρ is large, the probability of satisfying the GBR objective for each user increases since this performance is measured on the long term purpose by improving the system throughput and degrading the user fairness. But the long term GBR satisfaction does not guarantee the required amount of data in the short term downlink scheduling. Therefore, one purpose of this chapter is to find the optimal windowing factor which can permit to deliver the requested data rate on a short term purpose.

Let us define $\overline{\overline{\mathbb{T}}}[t] = \left[\overline{\overline{T}}_1, \overline{\overline{T}}_2, ..., \overline{\overline{T}}_I\right]$ the momentary average throughput vector calculated based on MMF filter and $\underline{\mathbb{T}} = \left[\underline{T_1}, \underline{T_2}, ..., \underline{T_I}\right]$ the corresponding vector of GBR requirements. It is said that, the GBR objective is satisfied if the instantaneous vector $\overline{\overline{\mathbb{T}}}[t]$ satisfies the requirement vector for all active users. Furthermore, let us define $\mathcal{D} = \{1, 2, ..., D\}$ the set of scheduling rules oriented on GBR objective. At each TTI, each scheduling rule $d \in \mathcal{D}$ impacts differently in the GBR satisfaction of the average throughput vector $\overline{\overline{\mathbb{T}}}[t]$. The aim is to apply at each momentary scheduler state the best scheduling rule $d^* \in \mathcal{D}$ in order to maximize the number of TTIs when the vector $\overline{\overline{\mathbb{T}}}[t]$ satisfies the requirement vector $\underline{\mathbb{T}}$. If the overall GBR satisfaction is not possible due to certain network conditions (i.e. poor channel conditions, high number of users), the aim would be to maximize the percentage of user throughputs from $\overline{\overline{\mathbb{T}}}[t]$ that respects their GBR requirements from $\underline{\mathbb{T}}$.

The scheduler should be aware about the benefit of allocating each RB $j \in \mathcal{B}$ to UE $i \in \mathcal{I}_t$ from the viewpoint of the GBR objective. This cost value is determined for each user by a concave and monotone

utility function, defined as: $U_i : \mathbb{R} \to \mathbb{R}$, $U_i\left(\overline{T}_i\right) = F_i\left(\overline{T}_i\right) \cdot G\left(\overline{\overline{T}}_i\right)$, where F_i is a function that depends on the average user rate calculated with exponential moving filter in order to compensate the channel oscillations and $G\left(\overline{\overline{T}}_i\right)$ is the utility weight function designed to take as input the average user rate $\overline{\overline{T}}_i$ and output the priority of user $i \in \mathcal{I}_t$ to be scheduled on a given RB at TTI *t*. In fact, a scheduling rule defines the type of utility function with the condition that the same utility is assigned for all active users within one TTI. Then, the utility function of user $i \in \mathcal{I}_t$ can be defined as: $U_{d,i}\left(\overline{T}_i\right) = F_i\left(\overline{T}_i\right) \cdot G_{d,i}\left(\overline{\overline{T}}_i\right)$ in which we keep the same function F_i for all scheduling rules while the weight function $G_{d,i}$ depends on the selected scheduling rule $d \in \mathcal{D}$ at each TTI.

3.1 Optimization Problem

By using the Taylor's expansion (Comşa, 2014a), the proposed short-term optimization problem aims to allocate the radio resources as well as to decide the scheduling rule to be applied at each TTI, such as:

$$
\begin{aligned}
&\max_{b,c} \sum_{d=1}^{D} \sum_{i=1}^{I_t} \sum_{j=1}^{B} b_{d,i}[t] \cdot c_{i,j}[t] \cdot F_i'\left(\overline{T}_i\right) \cdot G_{d,i}\left(\overline{\overline{T}}_i\right) \cdot r_{i,j}[t] \\
&\quad \sum\nolimits_d b_{d,i}[t] = 1, \; i = 1,...,I_t, && (a) \\
&\quad \sum\nolimits_i b_{d^*,i}[t] = I_t, \; d^* \in \mathcal{D}, && (b) \\
&s.t. \quad \sum\nolimits_i b_{d^{\otimes},i}[t] = 0, \; d^{\otimes} \in \mathcal{D}, \forall d^{\otimes} \neq d^*, && (c) \\
&\quad \sum\nolimits_i c_{i,j}[t] \leq 1, \; j = 1,...,B, && (d) \\
&\quad b_{d,i}[t] \in \{0,1\}, \; \forall d \in \mathcal{D}, \forall i \in \mathcal{I}_t, && (e) \\
&\quad c_{i,j}[t] \in \{0,1\}, \; \forall i \in \mathcal{I}_t, \forall j \in \mathcal{B}, && (f)
\end{aligned}
\tag{4}
$$

where, the optimization problem uses the first derivative of utility function $U_{d,i}'\left(\overline{T}_i\right) = F_i'\left(\overline{T}_i\right) \cdot G_{d,i}\left(\overline{\overline{T}}_i\right)$. The function F_i must be chosen to attenuate the channel variations such as: $F_i\left(\overline{T}_i\right) = \log\left(\overline{T}_i\right)$, and then, $F_i'\left(\overline{T}_i\right) = 1/\overline{T}_i$. In (4), $r_{i,j}[t]$ is the achievable rate of user $i \in \mathcal{I}_t$ for RB $j \in \mathcal{B}$ at TTI t, being calculated as $r_{i,j}[t] = N_{i,j}^{bits}[t]/0.001$ (Comsa, 2014a), where $N_{i,j}^{bits}[t]$ is the maximum number of bits that can be transmitted if RB $j \in \mathcal{B}$ would be allocated to user $i \in \mathcal{I}_t$. Basically, $N_{i,j}^{bits}[t]$ is determined based on the Channel Quality Indicator (CQI) which is transmitted by each UE to the base station at certain number of TTIs. However, if $r_{i,j}[t] \cdot F_i'(\mathrm{x}_i) \approx 1$, then the optimization problem is focused more on the GBR objective since those users with higher values of $G_{d,i}\left(\overline{\overline{T}}_i\right)$ are preferred to be scheduled. Also, if $G_{d,i}\left(\overline{\overline{T}}_i\right) = 1$, then the optimization problem is focused more on scheduling those users with higher $r_{i,j}[t] \cdot F_i'\left(\overline{T}_i\right)$ values, and thus, the fairness objective is addressed.

The optimization problem from (4) is non-linear programming model and the set of constraints is convex. In the optimization problem, $b_{d,i}[t] \in \{0,1\}$ is the scheduling rule assignation variable: $b_{d,i}[t]=1$ when scheduling rule $d \in \mathcal{D}$ is assigned to user $i \in \mathcal{I}_t$ and $b_{d,i}[t]=0$, otherwise. The time-based RB allocation variable is $c_{i,j}[t]=1$ when the RB $j \in \mathcal{B}$ is allocated to user $i \in \mathcal{I}_t$, and $c_{i,j}[t]=0$, otherwise. Constraints (a) denote that the only one scheduling rule or utility function can be associated for each user $i \in \mathcal{I}_t$. The selected scheduling rule $d^* \in \mathcal{D}$ must be same for each user $i \in \mathcal{I}_t$, fact which is denoted by constraints (b) and (c). Constraints (d), allocates at most one user (if the corresponding data queues are not empty) per each RB $j \in \mathcal{B}$. Moreover, constraints (e) and (f) make the problem combinatorial. If the type of utility function is a priori decided, then the allocation metric becomes:

$$m_j[t] = \underset{i \in \mathcal{I}_t}{arg\,max} \left\{ G_{d,i}\left(\overline{\overline{T}}_i\right) \cdot r_{i,j}[t] / \overline{T}_i \right\} \tag{5}$$

where $m_j[t]$ indicates that RB $j \in \mathcal{B}$ is allocated to user $m \in \mathcal{I}_t, \forall m \neq i$ at TTI t. In this case, $c_{m,j}[t]=1$ and the rest of variables are $c_{i,j}[t]=0$, $\forall \{m,i\} \in \mathcal{I}_t, \forall i \neq m$. The same computation (5) is repeated for each RB until the entire spectrum is allocated. Once the resource allocation is performed, then the Modulation and Coding Scheme (MCS) is determined for each allocated user according to their CQIs. Also, the Transport Block (TB) is calculated in order to define the number of bits to be sent to each allocated user.

3.2 Conventional Utility Functions

According to (5), the performance of the scheduling process depends on the exploited rule $d \in \mathcal{D}$ that assigns the unique weight function $G_{d,i}$. In literature, three scheduling rules are proposed to optimize the GBR objective, such as:

- Barier Function (BF) ($d=1$): proposed by Lundevall et al. (2004) with the utility weight defined as follows:

$$G_{1,i}\left(\overline{\overline{T}}_i\right) = 1 + \omega_1 \cdot exp\left[-\omega_2 \cdot \left(\overline{\overline{T}}_i - \underline{T_i}\right)\right] \tag{6}$$

  where ω_1 and ω_2 are BF parameters. Basically, if user $i \in \mathcal{I}_t$ has its average throughput $\overline{\overline{T}}_i$ much below of its GBR requirement, then the allocation probability on RB $j \in \mathcal{B}$ increases exponentially as defined by (6).

- Minimum/Maximum (mM) ($d=2$): proposed by Andrews et al. (2005) stores an additional token counter for each user and the weight function is defined as follows:

$$G_{2,i}\left(\overline{\overline{T}}_i\right) = exp\left(\omega_3 \cdot TC_i[t]\right), TC_i[t] = max\left\{0, TC_i[t-1] + \underline{T_i} - T_i\right\} \tag{7}$$

  where ω_3 is the corresponding parameter and $T_i = \overline{\overline{T}}_i\left(W=1, x=0\right)$ is the instantaneous user throughput. Actually, mM rule introduces a timer when defining the weight function in the sense that, if the user instantaneous throughput is higher than its requirement, then that user can be still scheduled for certain period until the cooldown period starts and $TC_i[t] < 0$.

- Required Activity Detection (RAD) ($d=3$): Kolding (2006) proposes a simple weight function defined as:

$$G_{3,i}\left(\overline{\overline{T}}_i\right) = \underline{T}_i \Big/ \overline{\overline{T}}_i \tag{8}$$

where, the user with the lowest average rate compared to its requirement has the greatest chance to be scheduled on that resource block.

Studies conducted by Comsa (2014) reveal different behavior of these scheduling rules when the network conditions are variable in terms of: downlink channel type, number of active users, arrival bit rates and GBR requirements.

3.3 Proposed Utility Function

The scheduling rule proposed in this section is entitled Lagrange Multiplier (LM) since the proposed weight function take into account the following multiplier $\lambda_i[t]$ which is determined according to the following equation:

$$\lambda_i[t] = \lambda_i[t-1] + \omega_4 \cdot \left(\underline{T}_i[t] - T_i[t]\right) \tag{9}$$

where ω_4 is the forgetting factor which corresponds to the Lagrange multiplier computation for the GBR objective and $T_i = \overline{\overline{T}}_i\left(W=1, x=0\right)$ is the instantaneous user throughput at TTI *t*. Then, the proposed weight function is:

$$G_{4,i}\left(\overline{\overline{T}}_i\right) = log\left(\omega_5 + \lambda_i\right) \tag{10}$$

By setting proper values for ω_5, the user with the lowest multiplier has much less chances to be scheduled for the upcoming scheduling instants, whereas users with higher multipliers are much likely to be served by using the logarithmic scale. Even when users are experiencing very high fading oscillations $\left(r_{i,j}[t] \cdot F_i'\left(\overline{T}_i\right) >> 1\right)$, the optimization problem can be still focused on the GBR requirement satisfaction due to the logarithmic form of the weight function.

4. REINFORCEMENT LEARNING FRAMEWORK

The solution to the optimization problem exposed in (4) aims to find at each TTI *t*, the best rule assignation and resource allocation variables $\left\{b_{d,i}[t], c_{i,j}[t]\right\} \in \{0,1\}$ such that the GBR satisfaction condition $\overline{\overline{T}}_i \geq \underline{T}_i$ is met for each active user $i \in \mathcal{I}_t$. To decide jointly both decision variables would be practically impossible since the overall problem is combinatorial and each solution must be provided within one TTI. In order to implement the proposed optimization framework in real-time schedulers, we propose a sub-optimal solution that can be defined as follows: in the first stage, the scheduling rule assignation variable must be decided whereas, in the second stage, the resource allocation can be performed according to the metrics' calculation provided in (5). In this sense, we propose a reinforcement learning framework able to learn over time the most appropriate scheduling rule to be applied on each scheduler state.

Figure 1 presents the proposed RL framework, where an intelligent controller learns to take proper scheduling decisions based on the momentary scheduler states. By $\mathcal{S}$ we define the measurable, continuous and multi-dimensional scheduler state space, and $\mathrm{S}[t] \in \mathcal{S}$ is a vector that defines the momentary scheduler state at TTI *t*. The OFDMA packet scheduler (Fig. 1, right side) takes as input at each TTI two types of information: *a)* the momentary state $\mathrm{S} \in \mathcal{S}$ and *b)* the utility function decided by the controller. Then, the utility weights are determined based on one of the equations (6)-(10) followed by

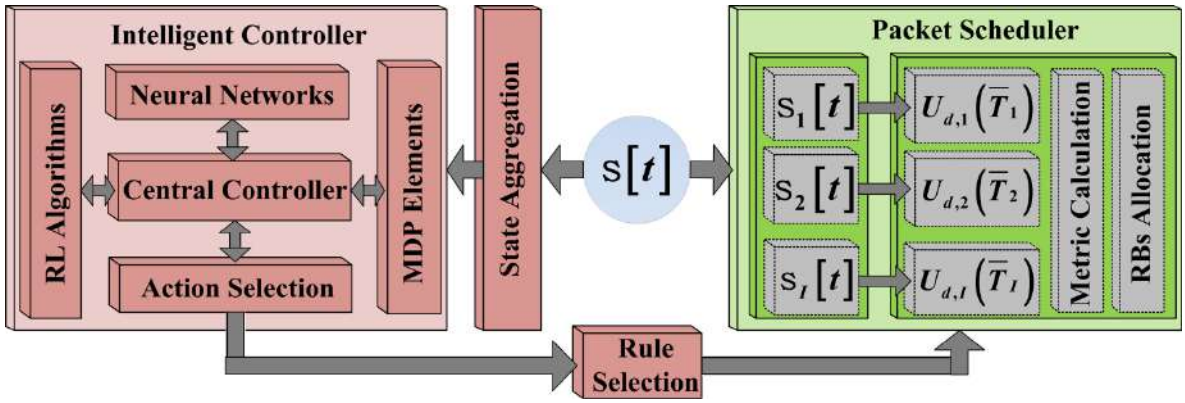


Figure 1 Proposed System Model

the metrics calculations and resource allocations that are performed according to (5). In contrast to the packet scheduler, the controller behaves as a *black box* that needs to decide the rules to be followed by the entire scheduling procedure based only on the momentary scheduler states.

At TTI t, the scheduler controller observes the momentary state $\mathrm{s}[t] \in \mathcal{S}$ and takes a decision regarding the type of scheduling rule that must be used by the scheduler. At TTI t+1, a new state $\mathrm{s}[t+1] = \mathrm{s}' \in \mathcal{S}$ is observed and a reward value is obtained from the RRM entity in order to evaluate the performance of the applied scheduling rule in the previous state $\mathrm{s}[t] \in \mathcal{S}$. Here, the reward value gives a general measure for the GBR satisfaction including all active users. The controller needs to experience high number of states' transitions in order to learn over time which is the best scheduling rule to be applied on each momentary state. This is basically the way how the RL principle works (Sutton and Barto, 2017): for many visits of the state-action pairs, the reward values are stored and discounted; then, the controller just follows the action that has the highest accumulated reward on that state. Due to the actual circumstances of the scheduler state space (continuous and multi-dimensional), the state-action values cannot be enumerated exhaustively. In this sense, we propose to learn only the approximation of the best scheduling rule to be applied on each momentary scheduler state.

According to Figure 1, the dimension of scheduler state space has to be reduced to a fixed dimension in order to enhance the learning performance and reduce the framework complexity. Then, we propose the use of neural networks to approximate the scheduling rule decision for each momentary state. A given neural network takes as input the momentary scheduler state and outputs the preference value of selecting its corresponding rule on that state. The previous state, action, reward and current state are stored in the Markov Decision Process (MDP) at every iteration. For each GBR oriented scheduling rule, we use one neural network for its approximation. According to the selected rule in the previous state, the corresponding neural network is updated in the next state once the reward value is given. The neural network weights corresponding to the selected scheduling rule are updated according to the information provided by the MDP entity and the type of RL algorithm which is used. This stage when the neural networks are updated is entitled *learning*. During the learning stage, the action selection block may choose to improve or to exploit what the neural networks have learnt so far. If the exploitation step is chosen, then the rule with the highest neural network output is selected to perform the scheduling procedure. If the improvement step is decided, then a different scheduling rule may be selected according to some probabilities. The action selection has the role of enhancing the learning outcome by enlarging the state space exploration. The learning stage continues until some convergence criteria are met. When the exploitation stage is performed, only the scheduling rule with the highest neural network output is selected on each momentary state.

4.1 State Space

The scheduler state space is divided in two disjoint sub-spaces: *a)* controllable sub-space $\mathcal{S}_C$ that evolves based on the selected scheduling rules and *b)* uncontrollable sub-space $\mathcal{S}_U$ which is rather stochastic and it cannot be predicted in general. Then, the obtained scheduler state space is $\mathcal{S} = \mathcal{S}_U \bigcup \mathcal{S}_C$. We define by $\mathrm{c}[t] \in \mathcal{S}_C$ the momentary controllable state and by $z[t] \in \mathcal{S}_U$ the momentary uncontrollable state at TTI *t*. Consequently, the momentary scheduler state is defined as: $\mathrm{s} = [\mathrm{c}[t], z[t]] \in \mathcal{S}$. By momentary uncontrollable state we mean the CQI reports from all users and the arrival rates in data queues. To improve the learning performance, the vector of GBR requirements $\underline{\mathrm{T}} = [\underline{T}_1[t], \underline{T}_2[t], ..., \underline{T}_I[t]]$ is randomly changed at different time periods. The number of active user is dynamically changed in the learning stage in order to increase the generality of the neural network approximations. Then, these parameters are also included in the uncontrollable scheduler state. The controllable state comprises the following elements: $\mathrm{c} = [\overline{\mathrm{T}}, \overline{\overline{\mathrm{T}}}, \phi, \mathrm{q}]$, where $\overline{\mathrm{T}} = [\overline{T}_1[t], \overline{T}_2[t], ..., \overline{T}_I[t]]$ is the instantaneous vector with average user rates calculated based on the EMF filter, $\overline{\overline{\mathrm{T}}} = [\overline{\overline{T}}_1[t], \overline{\overline{T}}_2[t], ..., \overline{\overline{T}}_I[t]]$ is the vector of average user rates determined based on the MMF filter, $\underline{\mathrm{T}} = [\underline{T}_1[t], \underline{T}_2[t], ..., \underline{T}_I[t]]$, and $\phi = [\phi_1[t], \phi_2[t], ..., \phi_I[t]]$ is the vector of differences between user rates and GBR requirements calculated for each use with the following formula: $\phi_i = \max\left\{0, \underline{T}_i[t] - \overline{\overline{T}}_i[t]\right\}$. Finally, $\mathrm{q} = [\mathrm{q}_1[t], \mathrm{q}_2[t], ..., \mathrm{q}_I[t]]$ is the vector with queue lengths for each active user. We consider for this study, that each active user has one active data queue.

4.2 Action Space

It is defined by $\mathcal{A} = \{a_1, a_2, ..., a_D\}$ the finite set of controller actions. If the selected action at TTI *t* is $a[t] = d$, then the scheduling rule $d \in \mathcal{D}$ is selected to conduct the scheduling procedure on that TTI.

4.3 Reward Function

In the reward computation, the average user throughput with MMF filter is considered only for each active user. From the user perspective, the reward is calculated as a normalized value of the difference between the average user throughput with MMF and its GBR requirement as expressed by the following equation:

$$r_{1,i}\left(\overline{\overline{T}}_i[t]\right) = \left(\overline{\overline{T}}_i[t] - \underline{T}_i[t]\right) \Big/ \underline{T}_i[t], \quad \forall i \in \mathcal{I}_t \tag{11}$$

If the reward is $r_{1,i}\left(\overline{\overline{T}}_i[t]\right) > 0$, then user $i \in \mathcal{I}_t$ is considered to be satisfied from the viewpoint of the GBR constraint. For reasons related more to the convergence of neural networks, the reward function for each active user is modeled by using the following equation:

$$r_{2,i}\left(\overline{\overline{T}}_i[t]\right) = \begin{cases} r_{1,i}\left(\overline{\overline{T}}_i[t]\right), \ if \ r_{1,i}\left(\overline{\overline{T}}_i[t]\right) < 0 \\ 1, \ if \ r_{1,i}\left(\overline{\overline{T}}_i[t]\right) \geq 0 \end{cases} \tag{12}$$

The intrinsic scheduler reward value is obtained by summing the rewards from (12) at each TTI such that:

$$r_{3,i}\left(\overline{\overline{\mathrm{T}}}[t]\right) = \sum_{i=1}^{I_t} r_{2,i}\left(\overline{\overline{T}}_i[t]\right) \tag{13}$$

The proposed reward function evaluates the performance of applying action $a[t] \in \mathcal{A}$ in state $\mathrm{s}[t] \in \mathcal{S}$. The reward value for pair (s, a) is received in state $\mathrm{s}'[t+1] \in \mathcal{S}$. It would be interesting to measure if there is any improvement of the intrinsic reward value $r_{3,i}\left(\overline{\overline{\mathrm{T}}}[t+1]\right)$ received at TTI t+1 compared to the previous intrinsic reward value $r_{3,i}\left(\overline{\overline{\mathrm{T}}}[t]\right)$ at TTI t. If we further consider now that $\mathrm{s}'[t+1] \in \mathcal{S}$ is the current state and $\mathrm{s}[t] \in \mathcal{S}$ is the previous one, then the global reward function is calculated as follows:

$$r\left(\overline{\overline{\mathrm{T}}}[t+1], \overline{\overline{\mathrm{T}}}[t]\right) = \begin{cases} 1, & \text{if } r_{3,i}\left(\overline{\overline{\mathrm{T}}}[t+1]\right) = 1 \\ r_{3,i}\left(\overline{\overline{\mathrm{T}}}[t+1]\right) - r_{3,i}\left(\overline{\overline{\mathrm{T}}}[t]\right), & \text{otherwise} \end{cases} \tag{14}$$

When the reward is $r = 1$, then the reward is maximized and all active users satisfy the GBR requirements. On the other side, when $r \geq 0$, then the reward is moderate and the scheduling rule applied in the previous state was a good action. However, when $r < 0$, then the controller gets a punishment and the applied action may not be a good option. For instance, in some networking conditions (when the traffic load is high and stringent GBR requirements) whatever the applied rule is, the reward is still negative. This makes part of noisy data which cannot be avoided when running real time RL frameworks. However, the scheduling rule that provides the minimum punishment value should be selected. The commitment of the proposed solution is to maximize the number of TTIs when the reward is maximized and minimize as much as possible the amount of punishment rewards.

4.4 Value Functions

The action selection block from Figure 1 implements the scheduling policy that maps momentary states over the action space. In learning stage, the scheduling policy is determined according to some probability functions in order to enhance the exploration capability of RL framework. In the exploitation stage, the output values provided by each neural network are used as distributions to rank the best scheduling rule to be applied every state. However, the scheduling policy is defined as a function $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$ and determined based on the following formula (Sutton and Barto, 2017):

$$\pi(d \mid \mathrm{s}) = \mathbb{P}\left(a[t] = d \mid \mathrm{s}[t] = \mathrm{s}\right) \tag{15}$$

where $\mathbb{P}(\cdot)$ is the probability of selecting action $a[t] = d$ by π when the actual state is $\mathrm{s}[t] = \mathrm{s}$.

The interest would be to measure the value of the initial state from where the whole learning stage starts. This means that we need a sum of discounted rewards from state to state starting from, let us say, $\mathrm{s}[0] = \mathrm{s}$ when given the policy π. By using notations from machine learning domain, this measurement is obtained by the value function defined as $V^{\pi} : \mathcal{S} \to \mathbb{R}$ and calculated as expressed below (Sutton and Barto, 2017):

$$V^{\pi}(\mathrm{s}) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \cdot R_{t+1} \mid \mathrm{s}[0] = \mathrm{s}\right] \tag{16}$$

where $\left(\gamma^t \cdot R_{t+1}; t \geq 0\right)$ is the accumulated reward value from state to state being discounted by the $\gamma \in [0,1]$ and $\mathrm{s}[0]$ is considered a random state such that $\mathbb{P}(\mathrm{s}[0] = \mathrm{s}) > 0$. If we also assume that the first action of the learning process is randomly chosen and $\mathbb{P}(a[0] = d) > 0$ keeps valid for all scheduling rules $d \in \mathcal{D}$, then we get the action-value function $Q^{\pi} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ determined as follows (Sutton and Barto, 2017):

$$Q^{\pi}(\mathrm{s},d)=\mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}\cdot R_{t+1}\,|\,\mathrm{s}[0]=\mathrm{s},a[0]=d\right] \tag{17}$$

and the rest of scheduling rules that follows state $\mathrm{s}[0]=\mathrm{s}$ are chosen according to policy π.

The purpose is to be able to access the value of these functions in between consecutive states in order to update them based on the received rewards. According to Comşa et al. (2018b), the following recursions are obtained for equations (16) and (17), respectively:

$$V^{\pi}(\mathrm{s})=r\left(\overline{\overline{\mathrm{T}}}[t+1],\overline{\overline{\mathrm{T}}}[t]\right)+\gamma\cdot V^{\pi}(\mathrm{s}') \tag{18.a}$$

$$Q^{\pi}(\mathrm{s},d)=r\left(\overline{\overline{\mathrm{T}}}[t+1],\overline{\overline{\mathrm{T}}}[t]\right)+\gamma\cdot V^{\pi}(\mathrm{s}') \tag{18.b}$$

The target of the proposed RL framework is to train the neural networks in order to obtain near-optimal value and action-value functions. However, the value function $V^{*}(\mathrm{s})$ is optimal when the values provided in each state have the highest expectable return such as $V^{*}(\mathrm{s})=\max_{\pi}V^{\pi}(\mathrm{s})$. Similarly, the optimal action-value function $Q^{*}(\mathrm{s},d)$ is the highest expected return when the scheduling process starts from $\mathrm{s}\in\mathcal{S}$ and the applied scheduling rule is $d\in\mathcal{D}$. When both functions are optimal, then the policy itself is optimal and the scheduling rule to be applied in momentary state $\mathrm{s}\in\mathcal{S}$ can be extracted by using the following formula:

$$d^{*}=\arg\max_{d'\in\mathcal{D}}\pi(d'\,|\,\mathrm{s}) \tag{19}$$

According to the optimality conditions, the equations expressed in (18.a) and (18.b) respectively, become:

$$V^{*}(\mathrm{s})=r\left(\overline{\overline{\mathrm{T}}}[t+1],\overline{\overline{\mathrm{T}}}[t]\right)+\gamma\cdot V^{*}(\mathrm{s}') \tag{20.a}$$

$$Q^{*}(\mathrm{s},d)=r\left(\overline{\overline{\mathrm{T}}}[t+1],\overline{\overline{\mathrm{T}}}[t]\right)+\gamma\cdot V^{*}(\mathrm{s}') \tag{20.b}$$

If we also consider that the optimal value function in state $\mathrm{s}'\in\mathcal{S}$ is $V^{*}(\mathrm{s}')=\max_{d'\in\mathcal{D}}Q^{*}(\mathrm{s}',d')$ (according to Comşa et al., 2018b), then the above equations can be rewritten as follows:

$$V^{*}(\mathrm{s})=r\left(\overline{\overline{\mathrm{T}}}[t+1],\overline{\overline{\mathrm{T}}}[t]\right)+\gamma\cdot\max_{d'\in\mathcal{D}}Q^{*}(\mathrm{s}',d') \tag{21}$$

For the purpose of the scheduling process oriented on GBR satisfaction, the optimality of these functions is not guaranteed since the state-action pairs are not exhaustively stored and enumerated. The aim is to find the best approximations of these functions by implementing the neural networks. During the learning stage, the value and action-value non-linear functions must be updated in each state according to the received rewards. The way of how thee weights of these functions are updated gives in fact the type of RL algorithms which is used. Actually, based on equations (20.a), (20.b) and (21), each RL algorithm has its own strategy to perform the learning stage.

4.5 State Space Aggregation

Under its original form, the momentary scheduler state is variable due to the dependence of controllable parameters on the number of active users. Also, the CQI state depends as well on the number of RBs and implicitly on the system bandwidth. For these reasons, the scheduler state space needs to be aggregated in order to reduce the state dimension to fixed value regardless the number of active users and system

bandwidth. The aggregation scheme aims to reduce the complexity of neural networks and speed-up the learning process. Comşa (2014) proposes an aggregation technique of CQI state space under three layers: *1)* the pre-processing layer is responsible of eliminating the system bandwidth dependence; *2)* classification layer performs the Radial Basis Function (RBF) classification for each user CQI to a given cluster of preprocessed CQI and *3)* regression layer applies statistical models to extract the most relevant features. Therefore, the CQI state space dimension is reduced from $I_t \times B$ to 4. Also, we use the proposal given by Comşa et al. (2018b) to reduce the dependency on I_t of the controllable elements $\mathrm{c}[t] \in \mathcal{S}_C$ and user arrival rates. We use the mean and standard deviation function based on likelihood estimators for each of these vectors that are depending on the number of users. This way, we obtain a reduction of this space dimension from $I_t \times 5$ to 2×5. The vector of GBR requirements does not need to be processed, since the same requirement is set for all active users. Thus, by encompassing the compression techniques expressed above, the obtained state is $\mathrm{v}[t] \in \overline{\mathcal{S}}$ that is an aggregate version of $\mathrm{s}[t] \in \mathcal{S}$ with a much lower dimension.

4.6 Function Approximations

The proposed RL framework makes use of $D+1$ neural networks: the neural network used to approximate the optimal value function is used as a critic regarding the selected actions on each states whereas other D neural networks are used to approximate the optimal action value functions for each considered scheduling rule. Then, let $\overline{V^*}: \overline{\mathcal{S}} \to \mathbb{R}$ be the value neural network and $\overline{Q^*}: \overline{\mathcal{S}} \times \mathcal{A} \to \mathbb{R}$ action-value neural networks represented based on the following equations:

$$\overline{V^*}(\mathrm{v}) = N\left[\theta_t, \psi(\mathrm{v})\right] \tag{22.a}$$

$$\overline{Q^*}(\mathrm{v}, d) = N_d\left[\theta_t^d, \psi(\mathrm{v})\right] \tag{22.b}$$

where $\{N, N_1, ..., N_D\}$ are the neural networks that model the non-linear functions, $\{\theta_t, \theta_t^1, ..., \theta_t^D\}$ are the weights at TTI t for each neural network, and $\psi(\mathrm{v})$ is the feature vector with non-linear transformations. Figure 2 illustrates the insights of the proposed RL framework when the learning stage is conducted based on five types of RL algorithms. At each TTI *t*, the momentary state $\mathrm{v}[t] \in \overline{\mathcal{S}}$ is propagated through all neural networks. If the action selection block decides to exploit the non-linear functions, then the scheduling rule corresponding to the neural network with the highest output is selected. Otherwise, a random action is chosen to perform the scheduling task. This state and action are saved in the MDP entity. At TTI *t*+1 when the reward is received the neural networks are updated according to the performed RL algorithm that uses the temporal difference values given by one of the equations (20.a), (20.b), and (21). After the learning stage is finished, the exploitation stage uses only the action value neural networks that rank the best scheduling rule to be applied in each compressed scheduler state.

Each neural network needs to define its structure before the learning stage is performed. A neural network is defined by L number of layers and M_l number of hidden nodes for each layer $l \in \{1, 2, ..., L\}$. The number of hidden nodes for input layer M_1 is the dimension of the compressed state $\mathrm{v}[t] \in \overline{\mathcal{S}}$ and

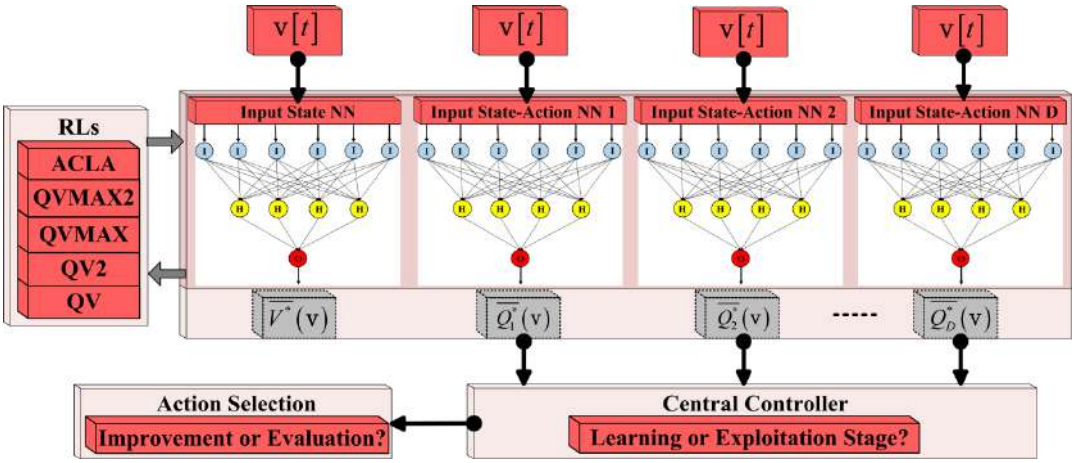


Figure 2 Proposed RL Framework

$M_L = 1$ since a single output value is provided by each neural network. A given vector of weights consists a number of $L-1$ matrices of weights to be updated such as: $\theta_t = \left[\mathrm{w}_{1,2}, \mathrm{w}_{2,3}, ..., \mathrm{w}_{L-1,L} \right]$, where the matrix between layers $l-1$ and l is: $\mathrm{w}_{l-1,l} = \left\{ w_{j,k}; j = 1,...,M_{l-1}, k = 1,...,M_l \right\}$. So, when running the learning stage, these weights for each neural network are updated according to the data provided by the MDP entity in three main stages: *a)* forward the momentary states $\left\{ \mathrm{v}[t], \mathrm{v}'[t+1] \right\} \in \overline{\mathcal{S}}$ through the neural networks; *b)* determine the error to be reinforced in each neural network according to the type of RL algorithm; *c)* back-propagate the errors to update the weights for each involved neural network.

4.6.1 Forward Propagation

We denote by $\mathrm{v}_+^{(l-1)}$ the compressed scheduler state being propagated through $l-1$ layers including the biased points and by $\mathrm{v}^{(l)}$ the scheduler state at the output of layer l. Then, the non-linear transformation between layers $l-1$ and l becomes (Comşa et al., 2018b):

$$\mathrm{v}^{(l)} = \psi_l \left(\mathrm{w}_{l-1,l}^T \times \mathrm{v}_+^{(l-1)} \right) \tag{23}$$

where $\mathrm{v}_+^{(l-1)}$ denotes the input values of $\mathrm{v}^{(l-1)}$ plus the bias point. The result of propagating the input state $\mathrm{v}[t] \in \overline{\mathcal{S}}$ through the entire structure looks like (Comşa et al., 2018b):

$$\mathrm{v}^{(L)} = \overline{V^*}(\mathrm{v}) = \psi_L \left(\mathrm{w}_{L-1,L}^T \times ... \times \psi_l \left(\mathrm{w}_{l-1,l}^T \times ... \times \psi_1 \left(\mathrm{v}^{(1)} \right) \right) \right) \tag{24}$$

Actually in the learning stage, the momentary state is propagated through a number of $D+1$ neural networks and the obtained values are: $\left\{ \mathrm{v}^{(L)}, \mathrm{v}_1^{(L)}, ..., \mathrm{v}_D^{(L)} \right\}$.

4.6.2 Error Calculation based on RL Algorithm

Over the learning stage, we aim to learn a set of $D+1$ neural networks. But at each TTI, only the value neural network and only one action value are updated at each TTI, where the updated action-value neural network corresponds to the selected rule in the previous state. Let us consider that the framework is in the

current state $\mathrm{v}' \in \overline{\mathcal{S}}$ at TTI t+1 and the selected action in the previous state $\mathrm{v} \in \overline{\mathcal{S}}$ is $a[t]=d$. Then, at TTI t+1 only the weights $\{\theta_t, \theta_t^d\}$ are updated. This is done by reinforcing the corresponding errors that are able to evaluate the quality of scheduling decisions at each TTI. We define by $e_{t+1}(\theta_t, \mathrm{v}', \mathrm{v})$ the value error calculated at TTI t+1 by using the value neural network with weights θ_t. Similar, $e_{t+1}^d(\theta_t^d, \mathrm{v}', \mathrm{v})$ is the action value error corresponding to the neural network with weights θ_t^d. These errors measure the difference between the target values obtained by using equations (20.a), (20.b) and (21) reported to propagated values $\{\mathrm{v}^{(L)}, \mathrm{v}_1^{(L)}, ..., \mathrm{v}_D^{(L)}\}$ in state $\mathrm{v} \in \overline{\mathcal{S}}$. In this sense, different RL algorithms are used to learn the best non-linear functions able to take proper scheduling decisions for different traffic types and network settings:

a) QV (Wiering and van Hasselt, 2009): the target functions for both value and action-value functions are determined based on equations (20.a) and (20.b), and the errors are calculated as follows:

$$e_t(\theta_t, \mathrm{v}', \mathrm{v}) = \left[r\left(\overline{\overline{\mathrm{T}}}[t+1], \overline{\overline{\mathrm{T}}}[t]\right) + \gamma \cdot \overline{V^*}(\mathrm{v}') \right] - \overline{V^*}(\mathrm{v}) \tag{25.a}$$

$$e_{t+1}^d(\theta_t^d, \mathrm{v}', \mathrm{v}) = \left[r\left(\overline{\overline{\mathrm{T}}}[t+1], \overline{\overline{\mathrm{T}}}[t]\right) + \gamma \cdot \overline{V^*}(\mathrm{v}') \right] - \overline{Q^*}(\mathrm{v}, d) \tag{25.b}$$

b) QV2 (Wiering and van Hasselt, 2009): the action value error is similar to (25.b), the target value functions is determined based on (20.a), and the value error is calculated as follows:

$$e_t(\theta_t, \mathrm{v}', \mathrm{v}) = \left[r\left(\overline{\overline{\mathrm{T}}}[t+1], \overline{\overline{\mathrm{T}}}[t]\right) + \gamma \cdot \overline{V^*}(\mathrm{v}') \right] - \overline{Q^*}(\mathrm{v}, d) \tag{26}$$

c) QVMAX (Wiering and van Hasselt, 2009): the action value error is similar to (25.b), the target for the value function is determined based on (21) and the corresponding error is determined as:

$$e_t(\theta_t, \mathrm{v}', \mathrm{v}) = \left[r\left(\overline{\overline{\mathrm{T}}}[t+1], \overline{\overline{\mathrm{T}}}[t]\right) + \gamma \cdot \max_{d' \in \mathcal{D}} \overline{Q^*}(\mathrm{v}', d') \right] - \overline{V^*}(\mathrm{v}) \tag{27}$$

d) QVMAX2 (Wiering and van Hasselt, 2009): the action value error is similar to (25.b), the target value function is determined based on (21) and the corresponding error is calculated as follows:

$$e_t(\theta_t, \mathrm{v}', \mathrm{v}) = \left[r\left(\overline{\overline{\mathrm{T}}}[t+1], \overline{\overline{\mathrm{T}}}[t]\right) + \gamma \cdot \max_{d' \in \mathcal{D}} \overline{Q^*}(\mathrm{v}', d') \right] - \overline{Q^*}(\mathrm{v}, d) \tag{28}$$

e) ACLA (Van Hasselt and Wiering, 2009): the value error is determined similar to (25.a); if this error is $e_t(\theta_t, \mathrm{v}', \mathrm{v}) \geq 0$, then the action $a[t]=d$ applied in state $\mathrm{v} \in \overline{\mathcal{S}}$ is a good option. Otherwise, the probability of selecting this action must be decreased. The action-value error is determined as follows:

$$e_{t+1}^d(\theta_t^d, \mathrm{v}', \mathrm{v}) = \begin{cases} 1 - \overline{Q^*}(\mathrm{v}, d), & \textit{if } e_{t+1}(\theta_t, \mathrm{v}', \mathrm{v}) \geq 0 \\ -1 - \overline{Q^*}(\mathrm{v}, d), & \textit{otherwise} \end{cases} \tag{29}$$

Regardless the chosen RL algorithm to conduct the learning stage, both value and action-value errors must decrease over time. At each TTI, the instantaneous errors must be reinforced in the corresponding neural networks in order to refine the weights $\left\{\theta_t, \theta_t^d\right\}$, process entitled back propagation.

4.6.3 Back Propagation

Both errors $\left\{e_{t+1}, e_{t+1}^d\right\}$ are back-propagated layer by layer at TTI t+1. For simplicity, we explain the back-propagation concept for the value error while the action-value error follows the same computations. Let us define the vector of errors $\mathrm{e}_{t+1}^{(l-1)} = \left[e_1^{(l-1)}, e_2^{(l-1)}, ..., e_{M_{l-1}}^{(l-1)}\right]$ that is propagated to output of layer $l-1$. These errors are back-propagated from layer l by using the following formula (Van Hasselt, 2011):

$$\mathrm{e}_{t+1}^{(l-1)} = \mathrm{W}_l^T \times \Delta^T\left(\Psi_l^{'}, \mathrm{e}_{t+1}^{(l)}\right) \tag{30}$$

where $\Psi_l^{'} = \left[\psi_{l,1}^{'}, \psi_{l,2}^{'}, ..., \psi_{l,N_l}^{'}\right]$ and $\Delta\left(\Psi_l^{'}, \mathrm{e}_{t+1}^{(l)}\right) = \left[\psi_{l,1}^{'} \cdot e_1^{(l)}, \psi_{l,2}^{'} \cdot e_2^{(l)}, ..., \psi_{l,M_l}^{'} \cdot e_{1,M_l}^{(l)}\right]$. By following equation (30), the errors are back-propagated to the input layer. We proceed in this way to update each particular matrix of weights between all layers by using the gradient descent principle. We denote by $\mathrm{W}_{l-1,l}^t = \left\{w_{l-1,l}^t; j = 1, ..., M_{l-1}, k = 1, ..., M_l\right\}$ the matrix of weights between layers l and $l-1$ that needs to be updated. For these computations, we need the propagated vector $\mathrm{v}^{(l-1)}$ to the output of layer $l-1$ and the error at the input of layer l $\Delta\left(\Psi_l^{'}, \mathrm{e}_{t+1}^{(l)}\right)$ (if we browse the neural network from input to output layer). Then, the updating process of weight $w_{l-1,l}^t$ takes the following form (Van Hasselt, 2011):

$$w_{j,k}^{t+1} = w_{j,k}^t + \eta_t \cdot v_j^{(l)} \cdot \psi_{l,k}^{'} \cdot e_k^{(l)} \tag{31}$$

where $\eta_t \in [0,1]$ is the learning rate and $v_j^{(l)}$ is the output value of node j of layer l.

4.7 Action Selection in Learning Stage

In the learning stage, after updating the neural weights, an action has to be decided for state $\mathrm{v}'[t+1] \in \overline{\mathcal{S}}$. The action selection block from Figure 2 aims to enhance the exploration of the scheduler state space by selecting the improvements or exploitation steps based on some probabilities. Two types of probability distributions are studied: ε - greedy and Boltzmann. The former one aims to select the action $a[t+1] \in \mathcal{A}$ based on the following policy function (Van Hasselt, 2011):

$$\pi\left(d' \mid \mathrm{v}'\right) = \begin{cases} \varepsilon_{t+1}(d'), & if \quad \varepsilon \le \varepsilon_{t+1} \\ \mathrm{v}_d^{'(L)}, & otherwise \end{cases} \tag{32}$$

where $\varepsilon_{t+1}(d') \in [0,1]$ is the random variable for action $a[t+1] = d'$, ε_{t+1} is the probability that decides between improvement and exploitation steps, and $\mathrm{v}_d^{'(L)}$ is the output value of action-value neural network in state $\mathrm{v}'[t+1] \in \overline{\mathcal{S}}$. Parameter ε can be fixed or can evolve according to some functions. Lower ε

values imply more improvements steps since random actions are likely to be selected. On the other hand, higher ε values imply more exploitation steps. The Boltzmann distribution considers only the output of the action-value neural networks. The potentially good actions for momentary state $\mathrm{v}' \in \overline{\mathcal{S}}$ are determined by using the following policy function (Van Hasselt, 2011):

$$\pi\left(d' \mid \mathrm{v}'\right) = \frac{\exp\left(\mathrm{v}_{d'}^{\prime(L)} / \tau\right)}{\sum_d \exp\left(\mathrm{v}_d^{\prime(L)} / \tau\right)} \tag{33}$$

where $\tau \in [0, \infty]$ is the temperature factor that sets how greedy the action should be. Lower τ values imply a more greedy scheduling policy that aim to select the scheduling rule corresponding to the neural network with the highest output. When τ is very large, the selection is more random.

5. SIMULATION RESULTS

The purpose of the obtained simulation results can be divided in two categories: (a) to study the performance of QV, QV2, QVMAX, QVMAX2 and ACLA RL algorithms for different traffic types and network settings and b) to evaluate the performance of learnt scheduling policies compared to state-of-the art schedulers. Three types of traffic are considered: infinite buffer, CBR and VBR. For each traffic type, the learnt scheduling policies are studied in terms of different settings of the windowing factor ρ in order to find the optimal filter length when computing the average user rates that maximizes the percentages of TTIs when all active users are 100% satisfied from the viewpoint of the GBR objective. The pool of scheduling rules consist the following techniques: BF, mM, RAD and LM as expressed by equations (6)-(10). For each configuration (traffic type and windowing factor), the learning stage is launched only once with the same time period. Ten simulations with different random user positions are launched for the exploitation stage. The results are averaged and the standard deviations are presented for each RL algorithm.

5.1 Parameter Settings

The considered system bandwidth is 100 MHz with the maximum number of $B = 100$ RBs. The retransmission scheme is an ARQ algorithm with maximum 5 retransmissions. Data packets failing to be retransmitted within this limit are declared lost. During learning and exploitation stages, the number of active users is randomly changed at each 1000 TTIs in the domain of $I_t \in [15;120]$. All users are moving in the macro-cell scenarios with 120kmph by using the random direction mobility model for the learning and exploitation stages in order to experience as many CQI observations as possible. For the interference model, we consider a cluster with 7 cells, and the simulation model runs only on the central cell, with others being used to provide the interference levels. The training stage runs for 500s by using the same user-network-application conditions for all five RL algorithms. The exploitation stages are launched in 10 different simulations of 95s each, and the results are averaged

The full buffer traffic model will consider full data queues at any time. However, those data packets which are waiting in the queue for more 300ms are dropped and implicitly are declared lost. The rates of CBR traffic that are also changed in the same interval of 1000 TTIs with the same values for all active users belonging to the following set $\{32;64;128;256;512;1024\}$. The VBR traffic generates the packet sizes and arrival rates according to Pareto and Geometric distributions (Comsa 2014a), respectively. The GBR requirements for each traffic type are switched for each active user randomly in the same interval of 1000 TTIs from the set of rate requirements $\underline{T}_i[t] \in \{32;64;128;256;512;1024\}$ kbps. In this way, the envi-

Table 1. OFDMA RL Framework Parameters

Parameters Name	**Description/Values**
System Bandwidth/Cell Radius	20 MHz/1000m
User Speed/Mobility Model	120kmph/Random Direction
Channel Model	Jakes Model
Path Loss / Penetration Loss	Macro Cell Model / 10 dB
Interfered Cells/Shadowing	0/8dB
Carrier Frequency/DL Power	2GHz/43dBm
Frame Structure	FDD
CQI Reporting Mode	Full-band, periodic at each TTI
PUCCH Model	Errorless
Scheduler Type	BF, mM, RAD, LM
$\{\omega_1;\omega_2;\omega_3;\omega_4\}$	$\{1.25;13.1\cdot10^{-5};10.1;2\}$
Traffic Type	Homogeneous: Infinite Buffer, CBR, VBR
Max. Number of schedulable users	10
RLC ARQ	Acknowledged Mode (Max. 5 retransmissions)
AMC Levels	QPSK, 16-QAM , 64-QAM
Target BLER	10%
Number of Users	Variable: 15-120
RL Algorithms	QV, QV2, QVMAX, QVMAX2, ACLA
Controller Timescale	1 TTI
Number of layers for neural networks	3
Number of Hidden Nodes	100
Exploration/Exploitation Periods	500s/95s
Windowing Factor	$\{2.5;4.0;5.5\}$
Dynamic GBR Constraints	$\underline{T_i} = \{32;64;128;256;512;1024\}\ kbps$
Maximum HoL Delay	300ms
CBR Traffic Type	Data Rates based on GBR Constraints $\lambda = \{32;64;128;256;512;1024\}\ kbps$
VBR Traffic Type	Packet size: Pareto Distrib. $(x = 35.5;\alpha = 1.1)$ Arv. Rate: Geometric Distrib. $(\mu = 1.5;\sigma = 1.93)$

ronment is very dynamic and the proposed RL framework is able to experience a variety of network conditions.

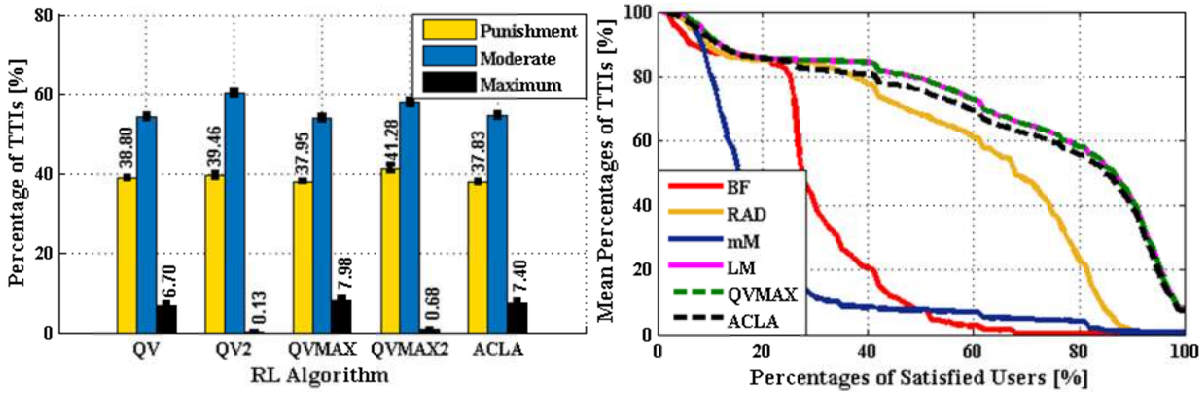
Based on some extensive simulation results, the optimum number of maximum schedulable uses at each TTI is set to $I_{\max} = 10$. It is also observed that for very low windowing factors, the GBR requirements cannot be satisfied by any involved scheduling rule. Very high windowing factors will increase the time when the GBR requirements are satisfied by the proposed scheduling policies and they are not able to react and take appropriate decisions for such large horizons of user throughputs. However, we would like to find the most suitable value for the windowing factor for each traffic class such that the GBR satisfaction is maximized. The rest of scheduler parameters are provided in Table I.

Table 2. RL Algorithms' Parameters

RL Algorithm	Learning Rates for Action Value Functions	Learning Rates for Value Functions	Discount Factor (γ)	Exploration Type (ε,τ)
QV	0.001	0.00001	0.99	Boltzmann $(\tau = 10)$
QV2	0.001	0.00001	0.95	Boltzmann $(\tau = 10)$
QVMAX	0.001	0.00001	0.99	Boltzmann $(\tau = 10)$
QVMAX2	0.001	0.00001	0.95	Boltzmann $(\tau = 10)$
ACLA	0.0001	0.0001	0.99	Greedy $\left(\varepsilon = 5 \cdot 10^{-4}\right)$

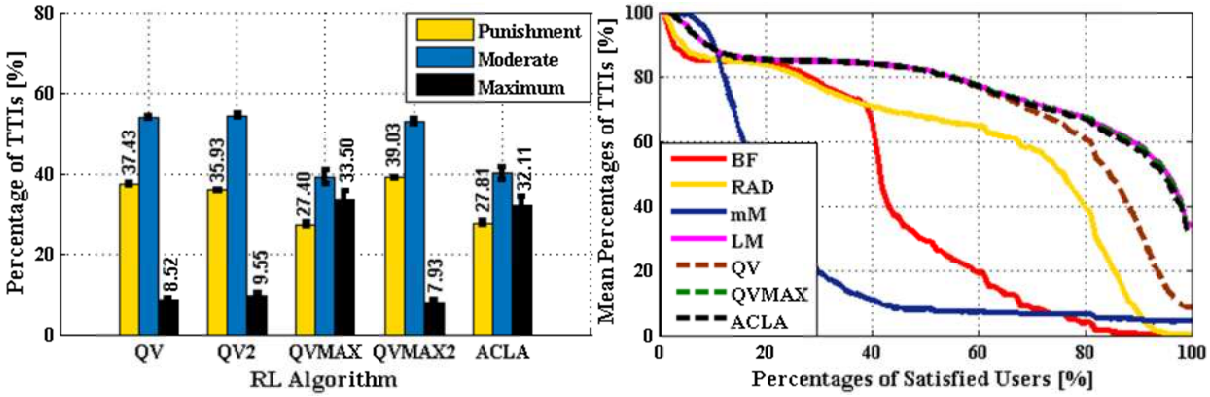
The controller parameters are represented by learning rates, discount factors, exploration parameters (ε,τ) and the configuration parameters of neural networks. For the given time period of learning, these parameters need to be properly set in order to minimize the set of errors $\left\{e,e^1,e^2,...,e^d\right\}$. Table 2 illustrates the most suitable parameters for each RL algorithm. Also, the neural networks need to be parameterized before launching the learning stage in terms of $\left\{L,M_1,...,M_L\right\}$. When the neural network is too flexible (involving large number of layers and hidden nodes), the RL framework complexity is higher, the learning is slower and the neural networks may overfit the input data. On the other side, reducing the number of hidden layers and nodes will underfit the input data in the sense that some parts from the scheduler state space will not be very well represented by the neural network functions. In both cases, the value error starts to increase starting at certain points in the learning stage. Different configurations of neural networks are tested by using extensive simulation results and considering the under-fitting, over-fitting and system complexity trade-off, the optimal setting for each neural network when addressing the GBR objective is: $L = 3$ and $M_2 = 100$. The same configuration is used for value and action-value neural networks. Also, the activation function for input and output layers $\left\{\Psi_1,\Psi_3\right\}$ is linear and for the hidden layer Ψ_2 is tangent hyperbolic.

The learning performance is studied for each RL algorithm which is used to update the neural network weights by measuring the percentage of TTIs when the reward is punishment, moderate and maximized, respectively. Let $p\left(-1 < r < 0\right)$ be the mean percentage of TTIs when the reward is punishment, $p\left(0 \le r < 1\right)$ is the mean percentage of TTIs when the reward is moderate and finally, we denote by $p\left(r = 1\right)$ the mean percentage of TTIs when the reward is maximized and consequently, the GBR requirements are satisfied by all active users. The aim is to find those scheduling policies that are able to provide the highest amount of $p\left(r = 1\right)$ and the lowest percentage of $p\left(-1 < r < 0\right)$. In the exploitation stage we also measure the mean percentage of TTIs when certain percent of users satisfies the GBR requirements. Let $q\%$ be the percentage of users that satisfy the GBR requirements and $p\left(q\%\right)$ the mean percentage of TTIs when the GBR objective is met for $q\%$ of active users. The exploitation stage is running in the same conditions as the learning stage and the simulation conditions are similar to all involved scheduling and RL algorithms.



(a) *(b)*

Figure 3 Mean Percentages of TTIs for (a) Punishment, Moderate and Maximum Rewards and (b) Percentages of Satisfied Users when the Windowing Factor is $\rho = 2.5$

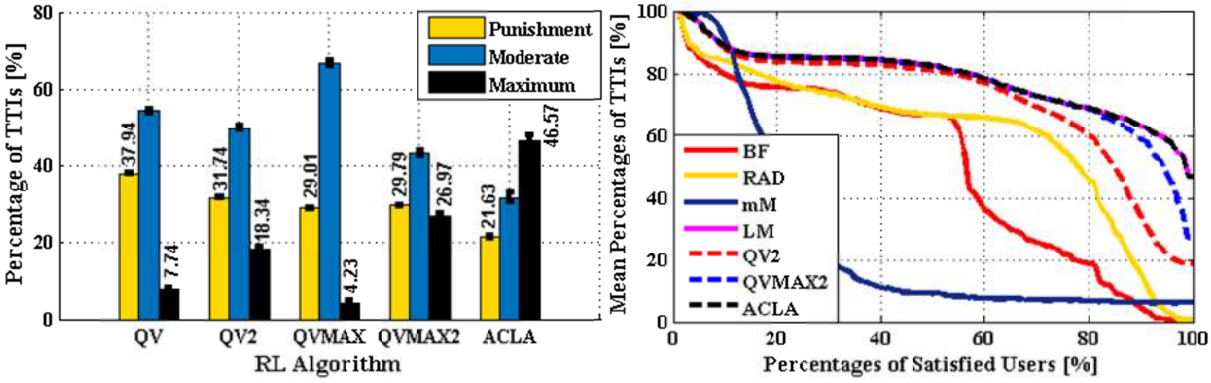


(a) *(b)*

Figure 4 Mean Percentages of TTIs for (a) Punishment, Moderate and Maximum Rewards and (b) Percentages of Satisfied Users when the Windowing Factor is $\rho = 4$

5.2 Full Buffer Traffic

For the full buffer traffic model, the impact of the learnt policies is studied only in terms of dynamic GBR requirements and number of users since the data queues are always full. When $\rho = 2.5$, QV, QVMAX and ACLA show the best performance from the viewpoint of $p(r = 1)$ maximization (as seen in Fig. 3.a). From the viewpoint of $p(-1 < r < 0)$, ACLA and QVMAX are the best options. In Figure 3.b, the performance of classical scheduling rules, ACLA and QVMAX policies is presented when measuring the mean percentage of TTIs when users are satisfied in the range percentage of $q \in [0,100]\%$. The proposed LM scheduling rule outperforms other rules for the entire domain of q. Also, the scheduling policies learnt with ACLA and QVMAX algorithms provide similar results when compared to LM scheduling rule. However, a small performance degradation can be observed for ACLA when the range between $p(40\%)$ and $p(80\%)$ is considered. Figure 4 presents the scheduling performance when $\rho = 4$. ACLA



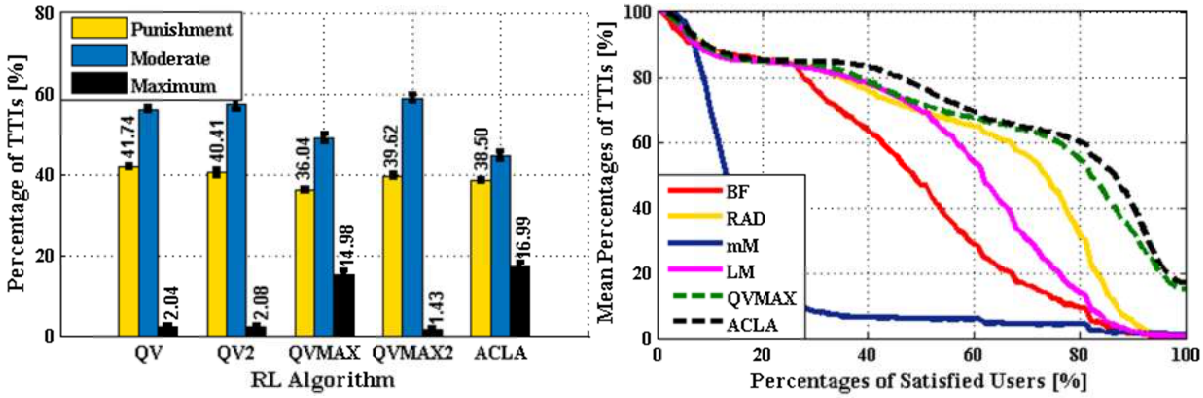
(a) *(b)*

Figure 5 Mean Percentages of TTIs for (a) Punishment, Moderate and Maximum Rewards and (b) Percentages of Satisfied Users when the Windowing Factor is $\rho = 5.5$

and QVMAX assures the best percentages of maximized rewards as presented in Figure 4.a. When comparing with other schedulers, ACLA and QVMAX policies together with the LM scheduling rule perform the best. When the windowing factor increases to $\rho = 5.5$, only ACLA algorithm is able to assure the best percentage of maximized rewards (Fig. 5.a). From the perspective of the mean percentage of TTIs with satisfied users, LM scheduling rule and ACLA policy are the best options. From Figures (3.a), (4.a) and (5.a), it can be observed that the mean $p(100\%)$ gradually increases according to the windowing factor. We conclude that $\rho = 5.5$ is the optimum windowing factor for the full buffer traffic type and the best choice is ACLA policy by providing a gain higher than $p(100\%) > 40\%$.

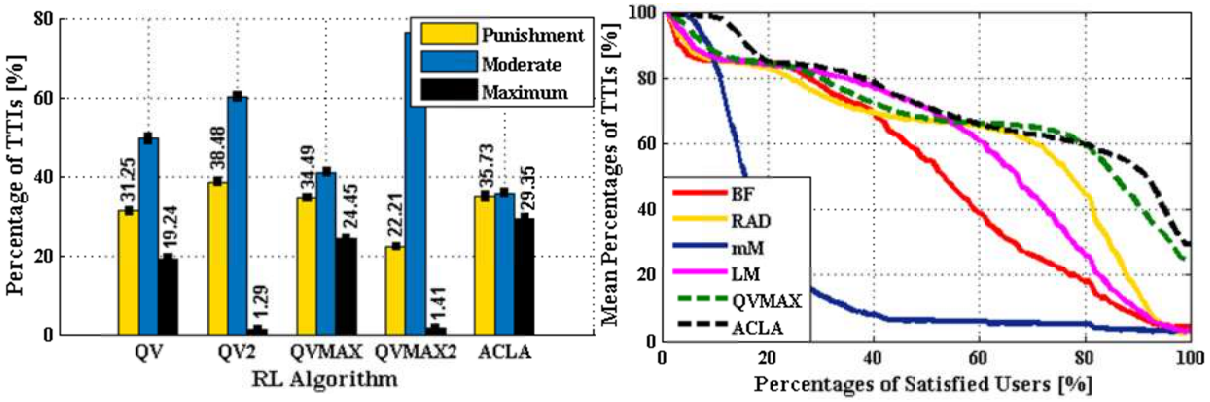
5.3 CBR Traffic

The GBR constraints are set based on the arrival rates that are switched at each 1000 TTIs for all active users. In Figure 6.a, ACLA and QVMAX obtain the best amount of maximized rewards $p(r = 1)$. QVMAX provides a slightly lower amount of punishments but a higher percentage of TTIs for $p(0 \le r < 1)$. These differences are quantified in Figure 6.b where it can be observed a slight degradation of provided services when exploiting the neural network trained by QXMAX. However, these policies are able to outperform the conventional scheduling rules when measuring the mean percentage of TTIs when users are satisfied in the interval of $[p(60\%); p(100\%)]$. If the windowing factor increases to $\rho = 4$, the same policies are able to provide the best performance when $p(r = 1)$ is measured (Fig. 7.a), while QV gets the lowest amount of punishments but at the price of much higher percentage of TTIs with moderate rewards. Figure 7.b show the advantage of using ACLA policy that is able to follow the distribution of percentages with satisfied users as follows: between $p(20\%)$ and $p(50\%)$, ACLA follows the LM rule that is the best choice for the considered interval; for $[p(50\%); p(100\%)]$, the policy provided by ACLA is able to outperform other candidates. As seen in Fig. 7.a, QVMAX obtains higher percentage of TTIs with punishment and moderate rewards when compared to ACLA policy. This difference is



(a) *(b)*

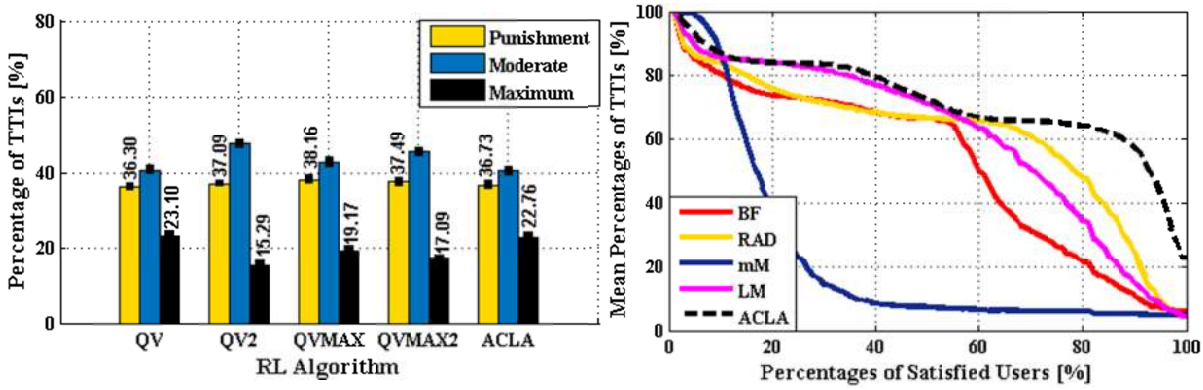
Figure 6 Mean Percentages of TTIs for (a) Punishment, Moderate and Maximum Rewards and (b) Percentages of Satisfied Users when the Windowing Factor is $\rho = 2.5$



(a) *(b)*

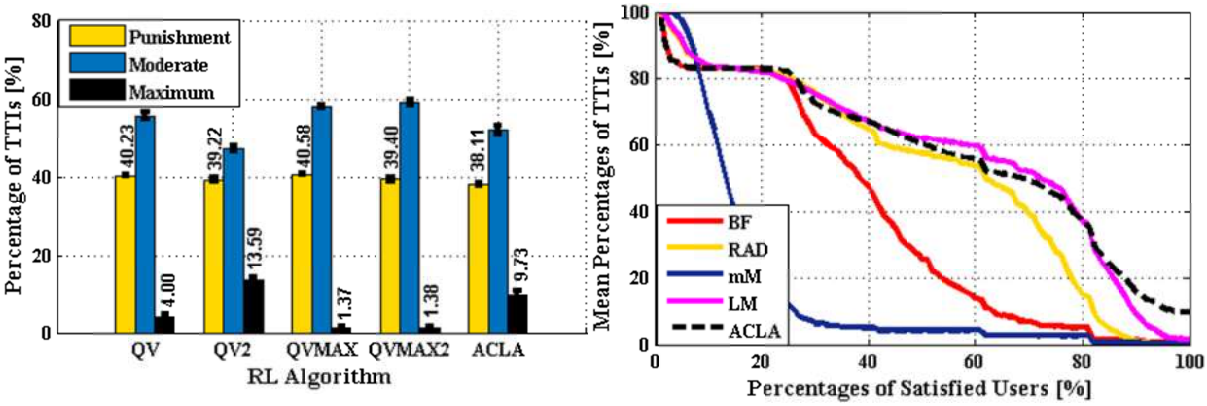
Figure 7 Mean Percentages of TTIs for (a) Punishment, Moderate and Maximum Rewards and (b) Percentages of Satisfied Users when the Windowing Factor is $\rho = 4$

quantified in Fig. 7.b where ACLA policy outperforms QVMAX. When $\rho = 5.5$, ACLA and QV policies obtain the highest amount of maximized rewards and the lowest percentage for the punishments rewards (Fig. 8.a). From Fig. 8.b. it can be observed that ACLA performs the best when measuring the percentages of TTIs when more than $q > 60\%$ of users satisfy the dynamic GBR requirements. When comparing to Fig. 7.b, ACLA policy with $\rho = 5.5$ is able to obtain higher percentage of TTIs when the considered interval is $q \in [70;90]\%$, but with lower amount of TTIs when 100% of active users are satisfied from the viewpoint of the GBR objective. Then, for CBR traffic type, the optimum value for the windowing factor is $\rho = 4$ and ACLA algorithm learns the best.



(a) *(b)*

Figure 8 Mean Percentages of TTIs for (a) Punishment, Moderate and Maximum Rewards and (b) Percentages of Satisfied Users when the Windowing Factor is $\rho = 5.5$

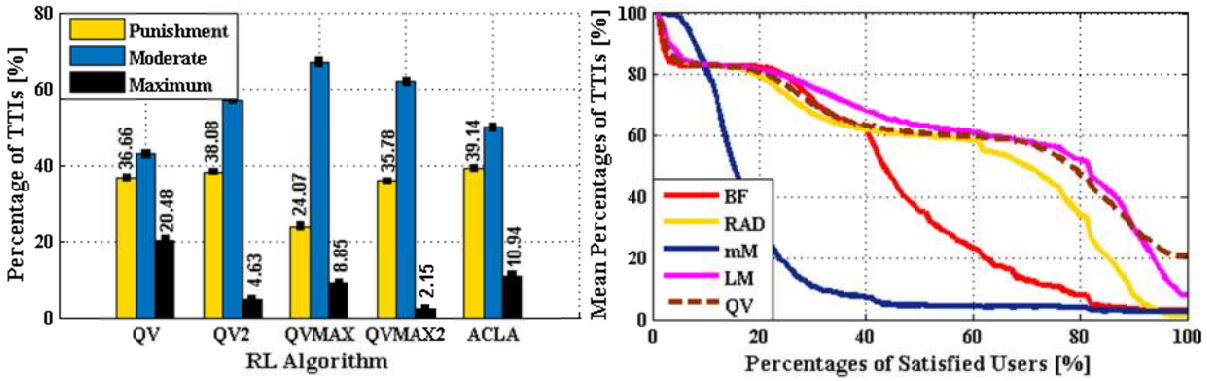


(a) *(b)*

Figure 9 Mean Percentages of TTIs for (a) Punishment, Moderate and Maximum Rewards and (b) Percentages of Satisfied Users when the Windowing Factor is $\rho = 2.5$

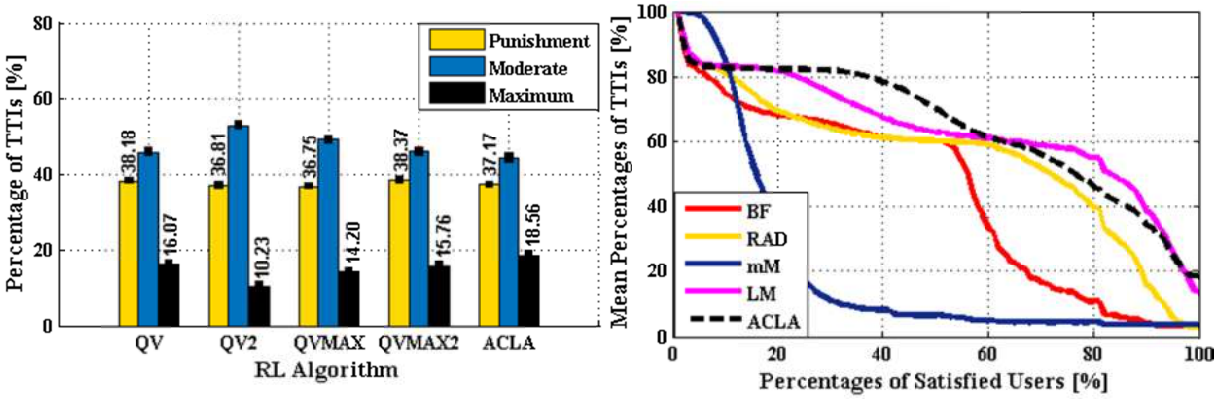
5.4 VBR Traffic

The VBR traffic is generated by considering variable arrival rates for each user at each TTI. Basically, the variability of networking conditions is higher than in other scenarios considered for full buffer and CBR traffic types. For lower windowing factors like $\rho = 2.5$, QV2 policy provides the highest percentage of TTIs when the reward is maximized while ACLA is getting the lowest percentage of punishments (as seen in Fig. 9.a). When comparing to other scheduling rules, ACLA achieves a gain greater than 10% when considering the range of $\left[p(90\%); p(100\%) \right]$. When increasing the windowing factor to $\rho = 4.0$ (Fig. 10.a), the QV policy is the best option from both viewpoints of $p(r = 1)$ and $p(-1 < r < 0)$, respectively. ACLA is the second best policy since QVMAX has the highest percentage of $p(0 \le r < 1)$ which means that, this policy is not able to provide satisfactory results for larger percentages' range of user satisfaction. Figure 10.b shows the performance evaluation of QV and other conventional rules. QV



(a) *(b)*

Figure 10 Mean Percentages of TTIs for (a) Punishment, Moderate and Maximum Rewards and (b) Percentages of Satisfied Users when the Windowing Factor is $\rho = 4$



(a) *(b)*

Figure 11 Mean Percentages of TTIs for (a) Punishment, Moderate and Maximum Rewards and (b) Percentages of Satisfied Users when the Windowing Factor is $\rho = 5.5$

policy is able to outperform the proposed LM scheduling rule when $p(100\%)$ is measured, but this policy provides a slight degradation when certain satisfaction intervals are considered. When measuring the performance of the obtained scheduling policies (Fig. 11.a), only ACLA and QV achieve satisfactory results when compared to other policies. However, as seen from Figure 11.b, ACLA is able to outperform the LM rule from the viewpoint of $p(100\%)$, but at the price of a degradation for the percentages of TTIs when the considered range of satisfied users is $q \in [70;90]\%$. This is in contradiction with the CBR traffic type, where ACLA is able to provide much better results. So, for windowing factors larger than $\rho = 4$, the scheduling policies are not able to follow the trajectory imposed by other scheduling rules. This is explainable since for larger horizons of time when computing the average rates, the scheduling policies are not able to predict the best scheduling rule decisions when the variable arrival rates are considered. Then, the best performance for the GBR satisfaction with VBR traffic is obtained when the windowing factor is $\rho \leq 4$ and the best RL algorithms must be chosen between QV and ACLA.

5.5 General Remarks

Alongside the windowing factor parameterization, the type of simulated traffic has a great influence on the performance of the exploited policies. In this sense, the proposed LM scheduling rule outperforms other candidate schemes from the viewpoint of the percentages of TTIs when the active users are satisfied in proportion of 100% for the full buffer traffic model. Only the policy trained by ACLA algorithm can follow the trajectory imposed by LM by indicating a gain of 5% to 40% when measuring $p(100\%)$ performance metric. For this traffic type, the optimum windowing factor is $\rho = 5.5$. With the CBR traffic type, the combination of different scheduling rules improves the GBR satisfaction of all active users by about 15% to 20% when ACLA policy is exploited. But the optimum windowing factor is reduced to $\rho = 4$. When the VBR traffic type is scheduled, the proposed scheduling policies are able to provide the best results only if the windowing factor is $\rho \le 4$. For larger windowing factor values, the exploited scheduling policies are able to provide the best percent of TTIs when all active users are satisfied but at the price of degrading the performance when the interval is $\left[p(65\%); p(95\%) \right]$.

6. CONCLUSIONS

This chapter proposes an innovative RL framework able to improve the GBR satisfaction for different types of traffic such as: full buffer, CBR and VBR. These traffic characteristics were chosen in order to cover a wide range of applications with various arrival rates and GBR requirements (e.g. video, VoIP, FTP, web browsing). By taking into account the dynamic channel conditions and traffic loads within the learning stages, the stability of the learnt policies are evaluated in terms of various RL algorithms. Each RL algorithm aims to select in each momentary scheduler state the most suitable scheduling rule that can get the highest GBR satisfaction outcome. In order to reduce the complexity of the proposed RL framework due to the very high dimension of the scheduler state space, the scheduling decisions are approximated by using non-linear neural networks. The simulation results indicate that the proposed RL framework is able to outperform other conventional scheduling rules when setting optimum time windows of average user rates for each category of involved traffic.

REFERENCES

Cisco. (2017). *Cisco Visual Networking Index: Global Mobile Data Traffic Fore-cast Update, 2016-2021 White Paper* Retrieved from http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ visual-networking-index-vni/mobile-white-paper-c11-520862.html.

Trestian, R., Comşa, I.-S., & Tuysuz, M. F. (2018). Seamless Multimedia Delivery within a Heterogeneous Wireless Networks Environment: Are We There Yet?. *IEEE Communications Surveys and Tutorials, 20*(2), 945 – 977.

Ghinea, G., Timmerer, C., Lin, W., and Gulliver, S. R. (2014). Mulsemedia: State of the Art Perspectives and Challenges. *ACM Transactions on Multimedia Computing Communications and Applications (TOMM)*, 11(17), 1-23.

Elbamby, M. S., Perfecto, C., Bennis, M. and Doppler, K. (2018). Toward Low-Latency and Ultra-Reliable Virtual Reality. *IEEE Network*, 32(2), 78 – 84.

Li, Y., Pateromichelakis, E., Vucic, N., Luo, J., Xu, W., and Caire, G. (2017). Radio Resource Management Considerations for 5G Millimeter Wave Backhaul and Access Networks. *IEEE Communications Magazine*, 55(6), 86 – 92.

Comşa, I.-S. (2014a). *Sustainable Scheduling Policies for Radio Access Networks Based on LTE Technology*. University of Bedfordshire, Bedfordshire, U.K.

Lundevall, M., Olin, B., Olsson, J. , Wiberg, N., Wanstedt, S., Eriksson, J., and Eng, F. (2004). Streaming applications over HSDPA in Mixed Service Scenarios. In *Proceedings of IEEE 60th Vehicular Technology Conference VTC2004-Fall,* (pp. 841-845), Los Angeles, CA, USA.

Andrews, M., Qian, L. and Stolyar, A. (2005). Optimal Utility based Multi-User Throughput Allocation Subject to Throughput Constraints. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies,* (pp. 2415- 2424), Miami, FL, USA.

Kolding, T.E. (2006). QoS-Aware Proportional Fair Packet Scheduling with Required Activity Detection. In *Proceedings of IEEE Vehicular Technology Conference*, (pp. 1-5), Montreal, Quebec, Canada.

Hasselt, H. P. (2011). *Insights in Reinforcement Learning Formal Analysis and Empirical Evaluation of Temporal-Difference Learning Algorithms*. University of Utrecht, The Netherlands.

FANTASTIC-5G. (2016). *D4.1: Technical Results for Service Specific Multi-Node/MultiAntenna Solutions.* Retrieved from http://fantastic5g.com/wp-content/uploads/2016/06/FANTASTIC-5G_D4.1_Final.pdf.

Yuan, Z., Chen, S., Ghinea, G., and Muntean, G. M. (2014). User quality of experience of mulsemedia applications. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, *11*(1s), 15.

Ghinea, G., and Ademoye, O. (2012). The sweet smell of success: Enhancing multimedia applications with olfaction. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, *8*(1), 2.

Ademoye, O. A., and Ghinea, G. (2009). Synchronization of olfaction-enhanced multimedia. *IEEE Transactions on Multimedia*, *11*(3), 561-565.

Ademoye, O. A., Murray, N., Muntean, G. M., and Ghinea, G. (2016). Audio masking effect on inter-component skews in olfaction-enhanced multimedia presentations. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, *12*(4), 51.

Yuan, Z., Ghinea, G., and Muntean, G. M. (2015). Beyond multimedia adaptation: Quality of experience-aware multi-sensorial media delivery. *IEEE Transactions on Multimedia*, *17*(1), 104-117.

Comşa, I.-S., Trestian, R., and Ghinea, G. (2018a). 360° Mulsemedia Experience over Next Generation Wireless Networks - A Reinforcement Learning Approach. In *Proceedings of Tenth International Conference on Quality of Multimedia Experience (QoMEX)* (pp. 1-6). Cagliari, Italy.

Comşa, I. S., Aydin, M., Zhang, S., Kuonen, P., and Wagen, J.-F. (2011). Reinforcement learning based

radio resource scheduling in LTE-advanced. In *Proceedings of 17th International Conference on Automation and Computing* (pp. 219 - 224). Huddersfield, UK.

Comşa, I. S., Aydin, M., Zhang, S., Kuonen, P., and Wagen, J.-F. (2012). Multi Objective Resource Scheduling in LTE Networks Using Reinforcement Learning. *International Journal of Distributed Systems and Technologies, 3*(2), 39-57.

Comşa, I.-S., Zhang, S., Aydin, M., Kuonen, P., and Wagen, J. (2012). A Novel Dynamic Q-Learning-Based Scheduler Technique for LTE-advanced Technologies Using Neural Networks. In *Proceedings of IEEE Conference on Local Computer Networks (LCN)* (pp. 332 – 335). Clearwater, FL, USA.

Comşa, I.-S., Aydin, M., Zhang, S., Kuonen, P., Wagen, J.-F., and Lu, Y. (2014b). Scheduling Policies Based on Dynamic Throughput and Fairness Tradeoff Control in LTE-A Networks. In *Proceedings of IEEE Conference on Local Computer Networks (LCN)* (pp. 418-421). Edmonton, AB, Canada.

Comşa, I.-S., Zhang, S., Aydin, M., Chen, J., Kuonen, P., and Wagen, J.-F. (2014c). Adaptive Proportional Fair Parameterization Based LTE Scheduling Using Continuous Actor-Critic Reinforcement Learning. In *Proceedings of IEEE Global Communication Conference (GLOBECOM)* (pp. 4387 - 4393). Austin, TX, USA.

Comşa, I.-S., De Domenico, A., & Ktenas, D. (2017). QoS-Driven Scheduling in 5G Radio Access Networks - A Reinforcement Learning Approach. In *Proceedings of IEEE Global Communications Conference (GLOBECOM)* (pp. 1-7). Singapore, Singapore.

Comşa, I.-S., Zhang, S., Aydin, M. Kuonen, P., Lu, Y., Trestian, R., and Ghinea, G. (2018b). Towards 5G: A Reinforcement Learning-based Scheduling Solution for Data Traffic Management. *IEEE Transactions on Network and Service Management (Early Access)*, 1-15.

Sutton, R. S., and Barto, A. G. (2017). *Reinforcement Learning: An Introduction.* England/London: IMT Press Cambridge.

Wiering, M. A. and van Hasselt, H. (2009). The QV Family Compared to Other Reinforcement Learning Algorithms. In *Proceedings of IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning* (pp. 101–108). Nashville, TN, USA.

Van Hasselt, H. and Wiering, M. A. (2009). Using Continuous Action Spaces to Solve Discrete Problems. in *Proceedings of International Joint Conference on Neural Networks* (pp. 1149–1156). Atlanta, GA, USA.