QoS-Driven Scheduling in 5G Radio Access Networks - A Reinforcement Learning Approach

Ioan-Sorin Comșa, Antonio De-Domenico and Dimitri Ktenas
CEA-Leti, Minatec Campus, 17 Avenue des Martyrs, 38054 Grenoble Cedex 9, France
E-mails: {ioan-sorin.comsa, antonio.de-domenico, dimitri.ktenas}@cea.fr

***Abstract*—The expected diversity of services and the variety of use cases in 5G networks will require a flexible Radio Resource Management able to satisfy the heterogeneous Quality of Service (QoS) requirements. Classical scheduling strategies have been designed to deal mainly with some particular QoS requirements for specific traffic types. To improve the scheduling performance, this paper proposes an innovative scheduler framework, that selects at each transmission time interval, the appropriate scheduling strategy capable to maximize the users′ satisfaction measure in terms of distinct QoS requirements. Neural networks and the Reinforcement Learning paradigm are jointly used to learn the best scheduling decision based on the past experiences. The simulation results show very good convergence properties for the proposed policies, and notable QoS improvements with the respect to the baseline scheduling solutions.**

***Index Terms*—5G, radio resource management, scheduling algorithms, reinforcement learning, neural networks.**

I. Introduction

Driven by the proliferation of services and applications, the 5G mobile network will be characterized by heterogeneous demands, in terms of e.g., latency, throughput, and reliability [1]. To handle these challenges, some important features will be adopted, such as: new waveforms, network densification, higher frequency bands, and dynamic Radio Resource Management (RRM) [2]. This RRM flexibility will enable more intelligent scheduling and resource allocation schemes, adaptive energy saving techniques, and smart mobility management. The key driver for such a paradigm shift is the data-driven network intelligence, where types of user-centric (e.g. channel conditions) and network-centric (e.g. traffic routing) data are stored in the cloud to learn intelligent RRM strategies [3].

In this study, we focus on the packet scheduler, a RRM entity which is a responsible of dynamically sharing the available spectrum between the active users at each Transmission Time Interval (TTI). Among the well-known schedulers, we can cite the Proportional Fair based EXPonential (PF-EXP) function, the PF based Barrier Function (PF-BF), and the PF based Opportunistic Packet Loss Fair (PF-OPLF), focusing on the packet delay reduction, Guaranteed Bit Rate (GBR) satisfaction, and Packet Loss Rates (PLRs) minimization, respectively [4], [5]. Other scheduling strategies prioritize users with real-time traffic over users without stringent latency requirements [6]. However, due to the multitude of services and requirements, none of the state-of-the-art packet schedulers can be considered as *an appropriate option* for all network conditions. Our idea is to apply each time the scheduling strategy that maximizes the satisfaction of QoS objectives. To achieve this goal in realistic environment (e.g. with traffic and channel characteristics), we develop a solution based on the Reinforcement Learning (RL) framework [7].

In the RRM context, RL algorithms have been proposed to optimize the parameters of the Generalized Proportional Fair scheduler [8], to control the discontinuous transmissions in small cells [9], to manage the parameters of self-organizing network functions [10] and to adapt the power control [11]. The main concern of the RL applicability is related to the algorithm complexity. To learn an effective strategy in realistic scenarios, where storing the reward values for each state-action pair is not feasible, linear approximations [10], fuzzy logic [9], and neural networks (NNs) [8] are implemented.

This paper proposes an innovative downlink scheduling framework that improves the QoS satisfaction in terms of delay, PLR and GBR requirements by selecting at each TTI, the best strategy based on the momentary scheduler state. To summarize, the main contributions of this paper are two-folds:
1) QoS-driven Scheduler Framework: We model and solve the problem of dynamically selecting the appropriate scheduling rule by taking into account the momentary system conditions and the heterogeneous requirements of the active users. In contrast with classic RRM schemes, which typically focus on the sum rate maximization, we investigate a multi-objective optimization problem, which is more challenging to solve. Our results show large gains in terms of QoS satisfaction.
2) Actor-Critic scheme based on NNs: We use an actor-critic (AC) RL scheme to learn the appropriate scheduler mechanism to be implemented at each TTI to maximize the QoS satisfaction. The AC scheme enables a higher stability in the learning process with respect to more classic methods (such as Q-Learning) [12]. To handle the complexity of this framework, given by the continuous and multi-dimensional state space, we approximate the AC solutions through NNs.

The rest of this paper is organized as follows: Section II presents the system model and the problem under investigation, Section III proposes the RL-based solution for the QoS-driven scheduling. The insights of AC scheme based on NNs are discussed in Section IV. Section V presents the simulation results and finally, the paper concludes with Section VI.

II. System Model and Problem Statement

A. System Model

We consider downlink communications in an OFDMA system, where the disposable bandwidth is divided in Resource Blocks (RBs), which represent the smallest amount of

This work has been performed in the framework of the Horizon 2020 project FANTASTIC-5G (ICT-671660) receiving funds from the European Union. The authors would like to acknowledge the contributions of their colleagues in the project, although the views expressed in this contribution are those of the authors and do not necessarily represent the project.

978-1-5090-3019-2/17/$31.00 ©2017 IEEE

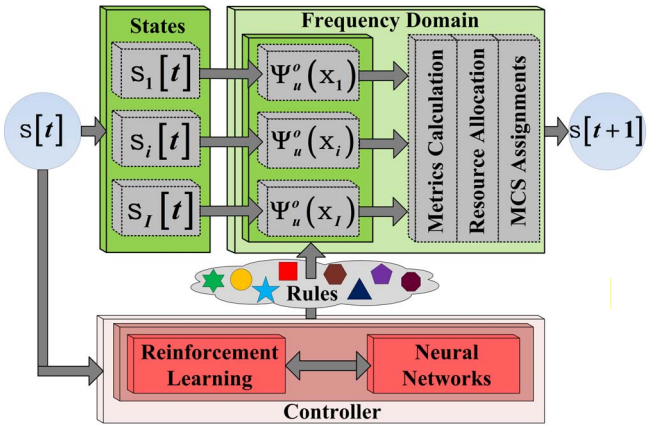

Fig. 1 Packet Scheduler Based on Proposed RL Framework.

frequency resources that can be allocated by a Base Station (BS). A User Equipment (UE) is characterized by traffics with different QoS requirements in terms of PLR, GBR, and delay. At the scheduler level, we consider various scheduling rules, each one mainly focusing on a specific QoS objective.

Let $\mathcal{J} = \{1, 2, ..., J\}$ denote the set of available RBs. Additionally, let $\mathcal{I} = \{1, 2, ..., I\}$ be the set of active users. The packet scheduler aims to share between the mobile users the available resources at each TTI t such that the QoS satisfaction is maximized. Let $\mathcal{O} = \{o_1, o_2, ..., o_O\}$ be the set of QoS objectives with O number of objectives. Furthermore, let $x_{i,o}[t]$ be the performance indicator of user $i \in \mathcal{I}$ in terms of objective $o \in \mathcal{O}$, and $\bar{x}_{i,o}$ the associated requirement. Specifically, the objective o is satisfied for the user i at TTI t, if the performance indicator $x_{i,o}[t]$ respects the requirement $\bar{x}_{i,o}$. Let $\mathcal{U} = \{u_1, u_2, ..., u_U\}$ be the set of scheduling rules with U number of rules. Moreover, let us consider the vectors $\mathbf{x}_i[t] = [x_{i,o_1}[t], x_{i,o_2}[t], ..., x_{i,o_O}[t]]$ and $\bar{\mathbf{x}}_i = [\bar{x}_{i,o_1}, \bar{x}_{i,o_2}, ..., \bar{x}_{i,o_O}]$ as the QoS indicators of UE $i \in \mathcal{I}$ and their requirements, respectively. Each rule u is associated with a utility function Ψ_u^o that, for each user, takes as input the performance indicator $\mathbf{x}_i[t]$ and outputs its priority when allocating the radio resources from $\mathcal{J}$ at each TTI t.

Once the appropriate scheduling rule is selected, the packet scheduling encompasses three stages as shown in Fig. 1: a) the function Ψ_u^o related to the selected scheduler is used to determine $I \times J$ ranking values, where each value provides a measure of how necessary is the allocation of a given RB $j \in \mathcal{J}$ to user i from the perspective of objective o; b) the resource allocation selects the user with the highest metrics for each RB; c) for each served UE, a Modulation and Coding Scheme (MCS) is assigned to its transmission, based on the Channel Quality Indicators (CQIs) of the allocated RBs.

B. Problem Statement

Our objective is to apply at each TTI t the most appropriated scheduling rule such that the user satisfaction in terms of the QoS objectives is maximized. This problem turns the classical resource allocation, where only one objective is considered (typically the network sum rate maximization), into an utility maximization problem, which depends on the resource allocation and scheduling rule selection. This is a multi-objective optimization problem, as shown in Eq. 1:

$$\max_{\mathbf{b},\mathbf{c}} \sum_{u\in\mathcal{U}} b_u[t] \sum_{i\in\mathcal{I}} \Psi_u^o(\mathbf{x}_i[t]) \sum_{j\in\mathcal{J}} c_{i,j}[t] \cdot \delta_{i,j}[t], \tag{1a}$$

$$\text{s.t.} \sum\nolimits_{i\in\mathcal{I}} c_{i,j}[t] \leq 1, \quad \forall j \in \mathcal{J}, \tag{1b}$$

$$\sum\nolimits_{u\in\mathcal{U}} b_u[t] = 1, \tag{1c}$$

$$b_u[t] \in \{0, 1\}, \quad \forall u \in \mathcal{U}, \tag{1d}$$

$$c_{i,j}[t] \in \{0, 1\}, \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}. \tag{1e}$$

In the objective function, the binary variable $b_u[t]$ indicates the scheduling rule selection (see the constraint (1d)), i.e., $b_u[t] = 1$ when the rule u is selected and $b_u[t] = 0$, otherwise. The selected function Ψ_u^o prioritizes the most urgent users to be scheduled at TTI t from the perspective of objective o. The binary variable $c_{i,j}[t]$ represents the RB selection (see the constraint (1e)), i.e., $c_{i,j}[t] = 1$ when the RB j is allocated to the user i and $c_{i,j}[t] = 0$, otherwise. The relation $c_{i,j}[t] = 1$ is valid when $i = argmax\,[\Psi_u^o(\mathbf{x}_k[t]) \cdot \delta_{k,j}[t]]$, where $k \in \mathcal{I}$ and $\delta_{k,j}[t]$ is the expected data rate calculated for each RB j and user k based on the reported CQIs from BS. In addition, the constraints (1b) indicate that a RB can be assigned to at most one UE, and the constraint (1c) forces the controller to select the same scheduling rule within one TTI for all users.

The constraints (1d) and (1e) make the problem combinatorial; moreover, the scheduling rule selection should be optimized jointly with the resource allocation, since the resource allocation depends on the scheduling rule and the performance of a scheduling rule is a function of the resource allocation. To find a solution to this complex problem, we design a RL framework, which is used to learn the appropriate RRM strategy that maximizes the QoS satisfaction. This framework is based on the AC scheme combined with NNs, which makes it reliable even in realistic environments with a large state space. Section III will highlight the preliminaries on the RL framework and in Section IV, we will discuss how the NNs are used in conjunction with the AC-RL scheme.

III. Preliminaries on RL Framework

RL is a stochastic optimization framework that enables a controller to learn the optimal policy, which maximizes a long term reward function. More specifically, at each TTI t, the controller observes its *state* and takes an *action*, accordingly. Then, at TTI $t+1$, it perceives a new state and an associated *reward* value, which evaluates the previous selected action. The controller goes through a series of transitions (i.e., it iterates from a state to another), and explores the possible state-action pairs until it learns the optimal policy. More specifically, the AC schemes uses two functions, the *actor*, which provides a preference value for each state-action pair, and the *critic*, which maintains a value function that evaluates each state. The critic is used to update the actors function in such a way that the controller strategy is improved.

A. System States and Actions

Let $\mathcal{S} = \mathcal{S}^U \cup \mathcal{S}^C$ be a finite set denoted as the controller state space, where $\mathcal{S}^U$ and $\mathcal{S}^C$ represent the uncontrollable and controllable state sets, respectively. More specific, $\mathbf{s}^U \in \mathcal{S}^U$ is the state vector that indicates the number of active users, the arrival rates, and the user CQIs, which

do not depend on the controller behavior. Moreover, $\mathbf{s}^C = \left[\mathbf{s}_1^C, \ldots, \mathbf{s}_I^C\right] \in \mathcal{S}^C$ describes the performance in terms of the QoS objectives, i.e., $\forall\, i \in \mathcal{I}$, and $\mathbf{s}_i^C = [\mathbf{x}_i[t], \bar{\mathbf{x}}_i - \mathbf{x}_i[t]]$.

The considered action set $\mathcal{A} = \{a_1, a_2, ..., a_U\}$ has a size of U actions. If $a[t] = u$, then the controller action at TTI t is the scheduling rule $u \in \mathcal{U}$. Now, let us consider that the controller state at TTI t is $\mathbf{s}[t] = \mathbf{s} \in \mathcal{S}$, per definition of the controllable state, we can compute the controllable state at TTI $t+1$, $\mathbf{s}^C[t+1] = \mathbf{x}' \in \mathcal{S}^C$, as a function of the previous controller state and selected action. Specifically, $\mathbf{x}'$ can take one of the following possibilities $[\mathbf{x}'_{u_1}, ..., \mathbf{x}'_{u_U}]$, where $\mathbf{x}'_u = [\mathbf{x}'_{u,1}, ..., \mathbf{x}'_{u,I}]$ is the achievable controllable set when applying action $a[t] = u$. Moreover, each of these sets is calculated based on the following transition function:

$$\mathbf{x}'_u = f(\mathbf{s}, u). \tag{2}$$

If the action $a[t] = u$ is decided to be the best one to be taken at TTI t, then the controllable set becomes: $\mathbf{x}' = \mathbf{x}'_u$.

B. Reward Functions

The reward function measures the goodness of the applied action in a given state. By considering the definition in [13], the reward function is determined as follows:

$$r(\mathbf{s}, u) \overset{(\mathbf{def})}{=} \mathbb{E}\Big[\mathcal{R}_{t+1} | \mathbf{s}[t] = \mathbf{s}, a[t] = u\Big], \tag{3}$$

where $\mathcal{R}_{t+1}$ is the reward value calculated at TTI $t+1$.

Theorem 1: For any selected action $a[t] = u$ in state $\mathbf{s}[t] = \mathbf{s}$, the reward function depends only on the controllable elements $\mathbf{x}'_u$ of the selected rule such as: $r(\mathbf{s}, u) = r(\mathbf{x}'_u, u)$.

Proof: By following the transition function of the scheduling procedure from Eq. (2) and Eq. (3), we have:

$$\begin{aligned} r(\mathbf{s}, u) &\overset{(3,2)}{=} \mathbb{E}\Big[\mathcal{R}_{t+1} | \mathbf{x}' = f(\mathbf{s}, u), \mathbf{s}[t] = \mathbf{s}, a[t] = u\Big] \\ &= \mathbb{E}\Big[\mathcal{R}_{t+1} | \mathbf{x}' = f(\mathbf{s}, u), a[t] = u\Big] \\ &= \mathbb{E}\Big[\mathcal{R}_{t+1} | \mathbf{x}' = \mathbf{x}'_u, a[t] = u\Big] = r(\mathbf{x}'_u, u). \quad \blacksquare \end{aligned} \tag{4}$$

We consider that: *a)* $o_1 = GBR$ and x'_{i,o_1} denotes the average rate; *b)* $o_2 = Delay$ and x'_{i,o_2} indicates the head-of-line packet delay; *c)* $o_3 = PLR$ and $x'_{i,3}$ is the average packet loss. Let us further consider that we have subsets $\mathcal{U}_o \subset \mathcal{U}$ in which each rule $u \in \mathcal{U}_o$ is focusing mainly on one objective o. Then, the proposed reward function is calculated as follows:

$$r(\mathbf{x}'_u, u) = \frac{1}{I} \cdot \left\{ \sum_{i=1}^{I} \left[R(\mathbf{x}'_{u,i}, u) + \frac{1}{2} \cdot \sum_{\substack{n=1 \\ o_n \neq o}}^{2} r_{o_n}(\mathbf{x}'_{u,i}) \right] \right\}, \tag{5}$$

where the sub-reward functions

$$r_{o_n}(\mathbf{x}'_{u,i}) = \begin{cases} 1 - \frac{\bar{x}_{i,o_n} - x'_{u,i,o_n}}{\bar{x}_{i,o_n}}, \bar{x}_{i,o_n} \geq x'_{u,i,o_n}, n = 1 \\ 1 + \frac{\bar{x}_{i,o_n} - x'_{u,i,o_n}}{x'_{i,o_n}}, x'_{u,i,o_n} \geq \bar{x}_{i,o_n}, n = \{2,3\} \\ 1, \qquad\qquad otherwise, n = \{1,2,3\}, \end{cases} \tag{6}$$

determine the costs of applying rule u from the perspective of other objectives $\mathcal{O} \setminus \{o\}$, where the set of QoS requirements $\{\bar{x}_{i,o_1}, \bar{x}_{i,o_2}, \bar{x}_{i,o_3}\}$ is fixed [14]. The function $R(\mathbf{x}'_u, u)$ determines the impact of rule u in the addressed objective o:

$$R(\mathbf{x}'_{u,i}, u) = r_{o_n}(\mathbf{x}'_{u,i}),\ if\ u \in \mathcal{U}_{o_n}. \tag{7}$$

C. State and Action Functions

We define a policy π as a probability of taking action $a[t] = u$ in state $\mathbf{s}[t] = \mathbf{s}$, such as:

$$\pi(\mathbf{s}, u) = \mathbb{E}[a[t] = u | \mathbf{s}[t] = \mathbf{s}]. \tag{8}$$

Moreover, let $\boldsymbol{\pi}(\mathbf{s}) = [\pi(\mathbf{s}, u_1), \ldots, \pi(\mathbf{s}, u_U)]$ denote the policy vector for each state $\mathbf{s} \in \mathcal{S}$.

We define $V^{\boldsymbol{\pi}} : \mathbb{R}^{D(\mathcal{S})} \rightarrow \mathbb{R}$ as a state-value function that measures the value of policy $\boldsymbol{\pi}$ starting from any initial state $\mathbf{s}[0] = \mathbf{s}$, where $D(\mathcal{S})$ is the state dimension. In addition, the state-action functions $Q^{\pi} : \mathbb{R}^{D(\mathcal{S})} \times |\mathcal{A}| \rightarrow \mathbb{R}$ measure the performance of each action $a[0] = u$, $\forall u \in \mathcal{U}$ starting from any initial state $\mathbf{s}[0] = \mathbf{s}$. These functions represent the discounted sums of RRM rewards, such that [13]:

$$V^{\boldsymbol{\pi}}(\mathbf{s}) \overset{(\mathbf{def})}{=} \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} | \mathbf{s}[0] = \mathbf{s} \Big], \tag{9}$$

$$Q^{\boldsymbol{\pi}}(\mathbf{s}, u) \overset{(\mathbf{def})}{=} \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} | \mathbf{s}[0] = \mathbf{s}, a[0] = u \Big], \tag{10}$$

where $\gamma \in [0, 1]$ is the discount factor of calculated rewards. We aim to find the optimal state value $V^*(\mathbf{s})$ which is the highest expected discounted reward value when the scheduling starts from any state $\mathbf{s} \in \mathcal{S}$ such as: $V^*(\mathbf{s}) \geq V^{\boldsymbol{\pi}}(\mathbf{s})$ [13].

Theorem 2: For any policy $\boldsymbol{\pi}$, there is a function $F^{\boldsymbol{\pi}} : \mathbb{R}^{D(\mathcal{S})} \rightarrow \mathbb{R}$ for each action $a = u, \forall u \in \mathcal{U}$, such that:

$$Q^{\boldsymbol{\pi}}(\mathbf{s}, u) = F^{\boldsymbol{\pi}}(\mathbf{x}'_u, u) \tag{11}$$

and there is a function $G^{\boldsymbol{\pi}} : \mathbb{R}^{D(\mathcal{S})} \rightarrow \mathbb{R}$ for which:

$$V^{\boldsymbol{\pi}}(\mathbf{s}) = G^{\boldsymbol{\pi}}[F^{\boldsymbol{\pi}}(\mathbf{x}'_{u_1}, u_1), ..., F^{\boldsymbol{\pi}}(\mathbf{x}'_{u_U}, u_U)]. \tag{12}$$

Proof: First, we change the state action function:

$$\begin{aligned} Q^{\boldsymbol{\pi}}(\mathbf{s}, u) &\overset{(10)}{=} \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} | \mathbf{s}[0] = \mathbf{s}, a[0] = u \Big] \\ &\overset{(2)}{=} \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} | \mathbf{s}^C[1] = f(\mathbf{s}, u), \mathbf{s}[0] = \mathbf{s}, a[0] = u \Big] \\ &= \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} | \mathbf{s}^C[1] = \mathbf{x}'_u, a[0] = u \Big] \\ &= F^{\boldsymbol{\pi}}(\mathbf{x}'_u, u). \end{aligned} \tag{13}$$

Then, the state function can be further developed as follows:

$$\begin{aligned} V^{\boldsymbol{\pi}}(\mathbf{s}) &\overset{(9)}{=} \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} | \mathbf{s}[0] = \mathbf{s} \Big] \\ &\overset{(8)}{=} \sum_{u \in \mathcal{U}} \Big\{ \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} | \mathbf{s}[0] = \mathbf{s}, a[0] = u \Big] \cdot \pi(\mathbf{s}, u) \Big\} \\ &\overset{(2)}{=} \sum_{u \in \mathcal{U}} \Big\{ \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} | \mathbf{s}^C[1] = \mathbf{x}'_u, a[0] = u \Big] \cdot \pi(\mathbf{s}, u') \Big\} \\ &\overset{(13)}{=} \sum_{u \in \mathcal{U}} F^{\boldsymbol{\pi}}(\mathbf{x}'_u, u) \cdot \pi(\mathbf{s}, u) \\ &= G^{\boldsymbol{\pi}}[F^{\boldsymbol{\pi}}(\mathbf{x}'_{u_1}, u_1), ..., F^{\boldsymbol{\pi}}(\mathbf{x}'_{u_U}, u_U)]. \quad \blacksquare \end{aligned} \tag{14}$$

We proved in Eqs. (13) and (14) that the state-action and state functions are depending on the achievable controlla-

$$G^{\boldsymbol{\pi}}(\mathbf{z}) = \mathbb{E}\Big[\mathcal{R}_1|\mathbf{s}^C[1]=\mathbf{x}', a[0]=u\Big] + \gamma \sum_{\mathbf{y}'}\sum_{u'}\Big\{\mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=1}^{\infty}\gamma^t\mathcal{R}_{t+1}|\mathbf{s}^C[2]=f[(\mathbf{y}',\mathbf{x}'),u'], a[1]=u', a[0]=u\Big]\pi[(\mathbf{y}',\mathbf{x}'),u']\Big\}$$

$$\stackrel{*}{=} r(\mathbf{x}',u) + \gamma\cdot\sum_{u'}\Big\{\mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=1}^{\infty}\gamma^t\mathcal{R}_{t+1}|\mathbf{s}^C[2]=\mathbf{x}''_{u'}, a[1]=u'\Big]\cdot\pi(\mathbf{s}',u')\Big\} = r(\mathbf{x}',u) + \gamma\cdot G^{\boldsymbol{\pi}}(z'_{u_1},...,z'_{u_U}). \quad (15)$$

ble parameters. For simplicity, we consider the following notations: $z_u = F^{\boldsymbol{\pi}}(\mathbf{x}'_u, u)$, $\mathbf{z} = [z_{u_1},...,z_{u_U}]$ and $z'_u = F^{\boldsymbol{\pi}}(\mathbf{x}''_u, u)$, where $\mathbf{x}''_u = f(\mathbf{s}', u)$. Then, the third term of Eq. (14) can be decomposed as shown in Eq. (15), where the property $(*)$ indicates the fact that the randomness of $\mathbf{y}' = \mathbf{s}^U[t+1]$ is eliminated since we are in state $\mathbf{s}'$ and we want to update the state value of the previous state $\mathbf{s}$. By using Eq. (15), the optimal state function is obtained when:

$$G^*(\mathbf{z}) = r(\mathbf{x}',u) + \gamma\cdot G^*(z'^*_{u_1},..z'^*_{u^*},..,z'^*_{u_U}). \quad (16)$$

where, $z'^*_u = F^*(\mathbf{x}''_u, u)$ is the optimal state-action function and $z'^*_{u^*} = F^*(\mathbf{x}''_{u^*}, u^*)$ is the optimal state-action value corresponding to the selected action u^*, where the optimal action is determined as follows: $u^* = argmax_{u'\in\mathcal{U}}F^*(\mathbf{x}''_{u'}, u')$.

IV. RL Framework

The analytical model exposed in Sub-section III.C cannot be used in practice under its original form due to two reasons. First, the scheduler state $\mathbf{s}\in\mathcal{S}$ is multi-dimensional and continuous, and it becomes automatically unfeasible to keep values for each state-action pairs. Secondly, in real-time scheduling systems, it is impossible to obtain the achievable vectors of controllable elements $[\mathbf{x}'_{u_1},...,\mathbf{x}'_{u_U}]$. For the first problem, the state and state-action functions can be represented by using some non-linear function approximations. For the second one, the idea is to use the time difference learning based on RL algorithm to train the non-linear functions.

A. Non-Linear Function Approximations

In non-linear function approximations, we aim to approximate the optimal state function by using some predetermined parameterized functions such as:

$$\widetilde{G^*_t}(\mathbf{z}) = h[\theta_t, \varphi(\mathbf{s})], \quad (17)$$

where h is the neural network, θ_t is a vector of weights and $\varphi(\mathbf{s})$ is the feature vector. Similar, the optimal scheduling rule functions can be approximated as follows:

$$\widetilde{F^*_t}(\mathbf{x}'_u, u) = h^u[\theta^u_t, \varphi(\mathbf{s})], u = u_1,...,u_U, \quad (18)$$

Note that $(h, h^{u_1},...,h^{u_U})$ and $\varphi(s)$ are fixed, and the vectors of weights $(\theta_t, \theta^{u_1}_t,...,\theta^{u_U}_t)$ must be tuned or learned.

A neural network consists a set of W_l weight matrices for each layer $l\in\{1,2,...,L\}$, where $L+1$ is the number of layers. Each matrix W_l has $(N_l+1)\cdot N_{l+1}$ number of weights to be tuned, where N_1 is the number of nodes in the first layer and N_{L+1} is the number of nodes of the output layer. In general, the functional form of a neural network aims to forward propagate the scheduler state $\mathbf{s}\in\mathcal{S}$ in desired output:

$$ForwardPropagation(\mathbf{s}) = \overrightarrow{\mathfrak{L}_L}...\overrightarrow{\mathfrak{L}_l}...\overrightarrow{\mathfrak{L}_1}\mathbf{s}, \quad (19)$$

where $\mathfrak{L}_l$ is the layer operator that takes a vector $\mathbf{s}^{(l)}$ of size N_l+1 (including the bias point for layer l) as input and provides a vector $\mathbf{s}^{(l+1)}$ of size N_{l+1} as an output, and

$$\mathfrak{L}_l\mathbf{s}^{(l)} = h_{l+1}(W_l^T\cdot\mathbf{s}^{(l)}+), \quad (20)$$

where $\mathbf{s}^{(l)}+ = (\mathbf{s}^{(l)}_1,...,\mathbf{s}^{(l)}_{N_l},1)$ is the biased input vector and $h_l : \mathbb{R}^{N_l}\rightarrow\mathbb{R}^{N_l}$ is the activation function. We consider that the activation function operates elementwise such that $h_l = (h_{l,1},..,h_{l,N_l})$ and the same activation function is used by all nodes within one layer.

B. Error Back-Propagation with Gradient Descent

The general idea is to learn $U+1$ vectors of weights and to update at each TTI t only two vectors $(\theta_{t-1}, \theta^{u*}_{t-1})$, where θ^{u*}_{t-1} is the set of weights for the neural network corresponding to the applied rule at TTI $t-1$. In this sense, the state function error is defined as a difference between the optimal value (Eq. 16) and the approximated one (Eq. 17):

$$\mathbf{e}_t(\theta_{t-1}, \mathbf{s}, \mathbf{s}') = G^*_t(\mathbf{z}) - \widetilde{G^*_t}(\mathbf{z}) \quad (21)$$
$$= r(\mathbf{x}',u) + \gamma\cdot h[\theta_{t-1}, \varphi(\mathbf{s}')] - h[\theta_{t-1}, \varphi(\mathbf{s})].$$

So, at each TTI, the state function error is computed based on the current state $\mathbf{s}'$, the received reward value and the previous state $\mathbf{s}$. In order to update the vector of weights θ_{t-1}, the current error $\mathbf{e}_t(\theta_{t-1}, \mathbf{s}, \mathbf{s}')$ must be back-propagated from layer to layer. The functional form of the back-propagation procedure is highlighted by the following chain of operators:

$$BackPropagation(\mathbf{e}) = \overleftarrow{\mathfrak{L}_1}...\overleftarrow{\mathfrak{L}_l}...\overleftarrow{\mathfrak{L}_L}\mathbf{e}, \quad (22)$$

where, the back-propagation operator of layer l acts as:

$$\overleftarrow{\mathfrak{L}_l}\mathbf{e}^{(l)} = W^T_{l,t}\cdot\mathbf{e}^{(l+1)}_t\cdot\nabla h_{l+1}(W_l^T\cdot\mathbf{s}^{(l)}+), \quad (23)$$

and $\nabla h_l = (h'_{l,1},...,h'_{l,N_L})$ are the derivative activation functions. Before the back-propagation operator is applied to layer l, the matrix of weights $W^T_{l,t-1}$ is updated based on the gradient descent principle. If $p = 1,...,(N_l+1)\cdot N_{l+1}$ is the weight index in layer $l\in\{1,2,...,L\}$ that links the output node $m = 1,...,N_l$ to input node $m+ = 1,...,N_{l+1}$, then the elementwise updating process is based on:

$$\theta_{p,t} = \theta_{p,t-1} + \eta_t\cdot\mathbf{s}^{(l)}_{m,t}\cdot\mathbf{e}^{(l+1)}_{m+,t}\cdot h'_{l+1,m+}, \quad (24)$$

where $\eta_t\in[0,1]$ is the learning rate.

C. Actor-Critic RL Algorithm

The AC RL scheme from [8] is used in this paper with the difference that the action space is discrete. This means that at most one neural network can be updated at each TTI t that corresponds to the applied scheduling rule at TTI $t-1$. The critic follows at each TTI the error of Eq. (21), such that:

$$F^*_t(\mathbf{x}'_u, u) = 1, \quad if \quad \mathbf{e}_t(\theta_{t-1}, \mathbf{s}, \mathbf{s}') \geq 0. \quad (25)$$

If the selected rule at TTI $t-1$ $u^*\in\mathcal{U}$ satisfies the condition from Eq. (25), then the set of weights θ^{u*}_{t-1} is updated by

Algorithm I. 5G Scheduler based on Actor-Critic RL Algorithm

for each TTI t
  observe state $\mathbf{s}'$, **recall** previous state-action $(\mathbf{s}, u^*)$, $u^* \in \mathcal{U}$
  calculate reward $r(\mathbf{x}')$ based on Eqs. (5), (6) and (7)
  forward propagate states $(\mathbf{s}, \mathbf{s}')$ on $h[\theta_{t-1}, \varphi(\mathbf{s})]$ - Eq. (19)
  calculate state error $\mathbf{e}_t(\theta_{t-1}, \mathbf{s}, \mathbf{s}')$ - Eq. (21)
  back propagate error $\mathbf{e}_t(\theta_{t-1}, \mathbf{s}, \mathbf{s}')$ - Eqs. (22), (23)
  update weights θ_{t-1} - Eq. (24)
  // criticize previous action $u^* \in \mathcal{U}$
  if $\mathbf{e}_t(\theta_{t-1}, \mathbf{s}, \mathbf{s}') \geq 0$
    forward propagate $(\mathbf{s}, \mathbf{s}')$ on $h^{u*}[\theta^{u*}_{t-1}, \varphi(\mathbf{s})]$ - Eq. (19)
    calculate error $\mathbf{e}^{u*}_t(\theta^{u*}_{t-1}, \mathbf{s}, \mathbf{s}')$ - Eq. (26)
    back propagate error $\mathbf{e}^{u*}_t(\theta^{u*}_{t-1}, \mathbf{s}, \mathbf{s}')$ - Eqs. (22), (23)
    update weights θ^{u*}_{t-1} - Eq. (24)
  end if
  // act based on the learned policy
  determine action $u'^* \in \mathcal{U}$ in state $\mathbf{s}'$ based on Eq. (27)
end for

back-propagating the error $\mathbf{e}^{u*}_t(\theta^{u*}_{t-1}, \mathbf{s}, \mathbf{s}')$ according to Eqs. (22), (23), and (24):

$$\mathbf{e}^{u*}_t(\theta^{u*}_{t-1}, \mathbf{s}, \mathbf{s}') = 1 - h^{u*}[\theta^{u*}_{t-1}, \varphi(\mathbf{s})]. \tag{26}$$

Otherwise, the structure of state-action NNs stays unchanged.

The actor impacts greatly in the learning stage, when the sets of weights are trained. The actor decides to follow the action value given by the NN output or to follow a random selection of actions in order to better tune the NN weights being updated so far. This is achieved by using the action selection based on $\epsilon-$ greedy rule such as:

$$\pi(\mathbf{s}') = \begin{cases} [\epsilon^{(u_1)}_t, ..., \epsilon^{(u_U)}_t] & \epsilon \geq 1 - \epsilon_t \\ [h^{(u_1)}, ..., h^{(u_U)}] & \epsilon < 1 - \epsilon_t, \end{cases} \tag{27}$$

where, $\epsilon^{(u)}_t \in [0, 1]$ is the random variable for each action $u \in \mathcal{U}$. At each TTI, the rule selection is based on: $u^* = argmax_{u' \in \mathcal{U}}[\pi(\mathbf{s}', u')]$. If the first condition of Eq. (27) is fulfilled, it means that we explore other actions for state $\mathbf{s}'$; otherwise, we exploit what the neural networks know so far. Summarized, the AC principle is presented by Algorithm I.

V. Simulation Results

As we mentioned in Sub-section III.B, objectives o_1 = GBR(G), o_2 = Delay(D) and o_3 = PLR(P) are considered when computing the proposed reward function. When refining the policies, the following rules are combined: u_1 = PF-BF $\in \mathcal{U}_{o_1}$, u_2 = PF-EXP $\in \mathcal{U}_{o_2}$ and u_3 = PF-OPLF $\in \mathcal{U}_{o_3}$. The purpose is to construct such scheduling policies that are able to call the best rule at each TTI based on the scheduler states.

If one scheduling policy would be trained for all traffic types, then the neural networks will require higher number of hidden layers and nodes for good generalizations. This becomes immediately unfeasible in real time scheduling. In order to simplify as much as possible the architecture of the neural networks, we train a different policy for each considered traffic type: Constant Bit Rate (CBR) ($\bar{x}_{i,o_1} = 256kbps, \bar{x}_{i,o_2} = 300ms, \bar{x}_{i,o_3} = 10^{-6}$), Variable Bit Rate (VBR) ($\bar{x}_{i,o_1} = 256kbps, \bar{x}_{i,o_2} = 300ms, \bar{x}_{i,o_3} = 10^{-6}$), non-conversational video traffic (buffered streaming) with the arrival rate of $\lambda = 138kbps$ ($\bar{x}_{i,o_1} = 128kbps, \bar{x}_{i,o_2} = 300ms, \bar{x}_{i,o_3} = 10^{-6}$) and conversational video (live streaming) at $\lambda = 242kbps$ ($\bar{x}_{i,o_1} = 180kbps, \bar{x}_{i,o_2} = 150ms, \bar{x}_{i,o_3} = 10^{-3}$) [14].

Table I Description of Simulation Parameters

Parameter	Value
System Bandwidth/Cell Radius	20 MHz/1000m
Speed/Mobility (Learning)	30 Kmph/Random Direction
Speed/Mobility (Exploitation)	Static/Uniform Distribution
Channel Model	Jakes Model
Path Loss/Penetration Loss	Macro Cell Model/10dB
Interfered Cells/Shadowing STD	6/8dB
Carrier Frequency/DL Power	2GHz/43dBm(equal on each RB)
Frame Structure	Frequency Division Duplexing
CQI Reporting Mode	Full-band, periodic at each TTI
PUCCH Model	Errorless
PDSCH Model	Wideband Error Model based on Effective SINR [15]
Scheduling Rules	PF-BF, PF-EXP, PF-OPLF
Traffic Types	CBR($640k$), VBR($512 - 1024k$), Video($138kbps$,$242kbps$)
Max. No. of Schedulable Users	32 each TTI
RLC ARQ	Acknowledged Mode (5 retransmissions)
AMC Levels	QPSK, 16-QAM, 64-QAM
Target BLER	10%
Number of Users (I_t)	Variable at each 1s (Learning)
RL Algorithm	Actor-Critic [8]
Number of NN layers	3 - based on apriori simulations
Activation Functions	input layer: linear, hidden layer: tangent hyperbolic, output layer: linear
Number of Hidden Nodes	80 - based on a priori simulations
Learning Duration	1000s for each traffic
Exploitation Duration	500s for each traffic load setting
Averaging Window (rate, PLR)	1000TTIs

Each scheduling policy is trained for 10^7ms and the number of users is changing at each 10^3ms by switching the equipments from IDLE to ACTIVE mode and vice-versa. The user speed is 30kmph, the mobility model is random direction and the handover procedure is considered to be set on OFF mode. The cell cluster size is 7, but only the the central cell is considered in the training process since other cells are used only to provide the interference levels from other BSs. For the exploitation stage, the learned policies are exploited for different traffic load settings with static positions and uniform distributed in the central cell. The exploitation stage is divided in ten simulations of 50s duration each, and the results are averaged. The rest of parameters are exposed in Table I. The software used is an extended version of LTE-Sim [15].

A. Learning Stage

The parametrization of the neural networks $(L, (N_l), l = 1, .., L)$ is one of the most important task that has to be decided before launching the learning stage. A neural network with a large number of layers will provide a better generalization of state and action-state functions, but at the price of an increased system complexity. In order to avoid this drawback, we found out based on some a priori simulations, that a minimum number of layers $(L = 3)$ for all functions $(h, h^{u_1}, .., h^{u_U})$ is enough to learn the scheduler model for each traffic type. Higher number of nodes N_2 for the hidden layer will provide a better generalization while a lower N_2 will make the NN inflexible. If N_2 is too large, then the NNs will diverge in the sense that the errors $(\mathbf{e}, \mathbf{e}^{u_1}, .., \mathbf{e}^{u_U})$ will increase significantly starting from a given point in the learning stage. Based on some a priori simulations, we found that $N_2 = 80$ is an optimum value to approximate these functions. The input and output activation functions are linear and the hidden activation function is tangent hyperbolic [8].

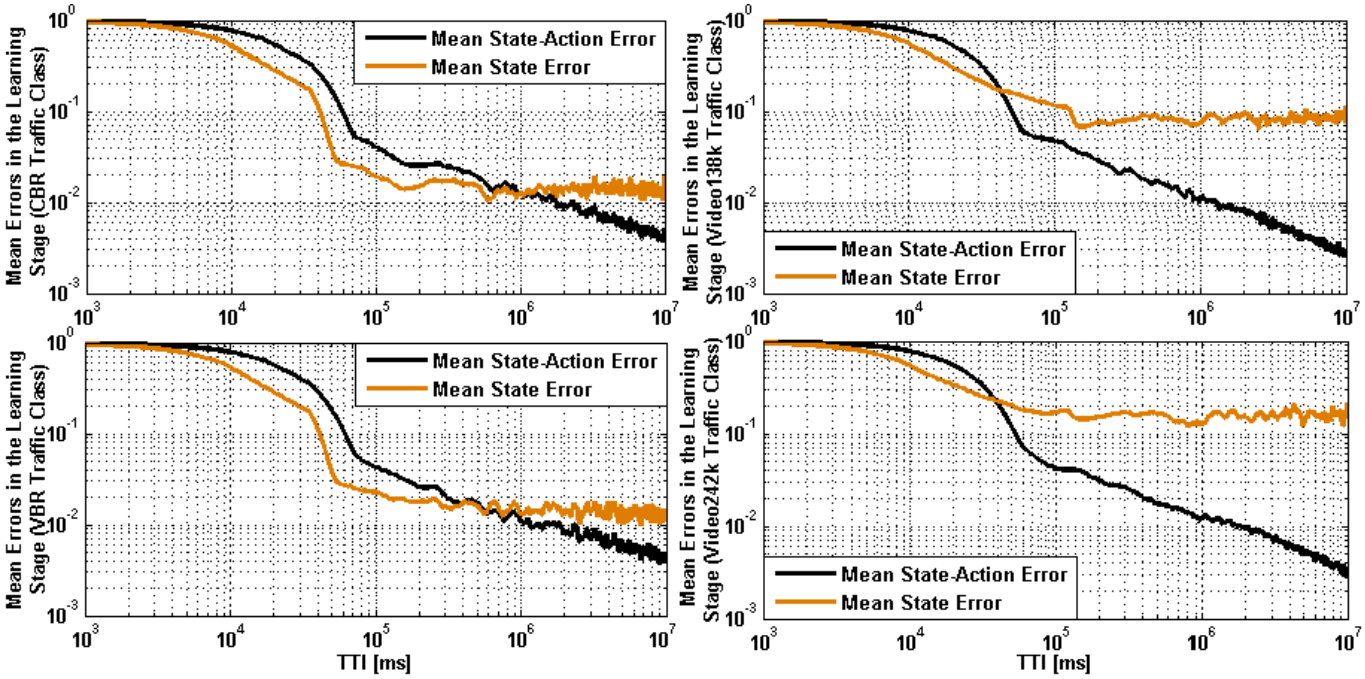

Fig. 2 Mean State and State-Action Errors in the Learning Stage

Another important aspect is denoted by the moment of time in the learning stage when the sets of weights should be saved to avoid the local minimum problems [13]. For this, we aim to keep track of average errors $\bar{\mathbf{e}}_t = 1/W \cdot \sum_{k=t-W}^{t} \mathbf{e}_k$ and $\mathbf{e}_t^{\bar{u}*} = 1/W \cdot \sum_{k=t-W}^{t} \mathbf{e}_k^{u*}, \forall u^* \in \mathcal{U}$. Each time when a new minimum is found for the mean state error ($\bar{\mathbf{e}}_t < min(\bar{\mathbf{e}})$), the entire set of NN weights $(\theta_t, \theta_t^{u_1}, ..., \theta_t^{u_U})$ is saved. Then, we explore more scheduler states until the instantaneous errors are getting stabilized. To avoid the local minimum, the mean state error is calculated by using a large horizon of $W = 20s$. Also, the learning rate η_t is decreased with the fix step when the NN weights are saved ($\eta_t = \eta_{old} - \Delta\eta_{fix}$). We aim to learn more at the beginning of the learning stage. On the other side, parameter ϵ_t from Eq. (27) is increased with a fixed step each TTI such as: $\epsilon_t = t/10^7$. We aim to explore more options in rule selection at the beginning of the learning stage, whereas at the end of the learning phase, we are exploiting much more the outputs of the neural networks.

Figure 2 plots the mean errors for both state and state-action functions and for all traffics. Three ideas are extracted. First, the state-action error decreases slower than the state error. The error $\mathbf{e}_t$ is reinforced each TTI in function $h(\theta, \phi(\mathbf{s}))$, whereas the error $\mathbf{e}_t^{u*}$ is back-propagated only if the critic will permit. Second, the state-action error is much lower than the state error at the end of the learning stage. The reason is that the state value needs to adapt to the instantaneous reward which is noisy due to the interference and channel errors, whereas the state-action error is adapting on the fixed target value as shown in Eq. (25). Third, the state errors for video traffics are higher than in other cases of CBR or VBR traffic. This is explicable since for lower number of video users, all data queues are empty. Consequently, the controller continues to evaluate the actions even if there is no impact in the scheduling performance. So, the state error is increasing when compared to other cases of CBR and VBR traffics where the load of queues is higher. We conclude that, the state error becomes stable after 1000s, the moment when the training stage can be stopped for all four traffic types.

B. Exploitation Stage

In the exploitation stage, we measure the normalized number of TTIs when objectives $(o_1 o_2 o_3) =$ (GDP) are satisfied for all four types of traffic. For the CBR traffic (Fig. 3.a), we observe the perfect tracking capabilities of our learned AC policy. For instance, when $I = [18, 25]$ users, the AC policy follows the best rule PF-OPLF. For a number of users higher than 27, the AC policy tracks the best rule PF-EXP. There is a considerable gain on our policy when the number of users is $I = \{26, 27, 28\}$. The multi-objective target is met for higher number of TTIs when selecting proper rules in this inflexion region. A maximum gain of 30% for I=26 users when compared to PF-EXP is achieved by our policy for the CBR traffic. The same gain is obtained for the VBR traffic type (Fig. 3.b). We proved that, the AC policy can select the appropriate rules even when the arrival rates are variable. For video traffic types, the impact of selecting the best scheduling strategy each TTI is more obvious. For the video traffic with $\lambda = 138kbps$ (Fig. 3.c), the maximum gains are: 66% when compared to PF-EXP and PF-OPLF for I=65 users, and 100% w.r.t. PF-BF for I=75 users. For $\lambda = 242kbps$, the AC policy shows a gain of 30% for I=37 users when compared to PF-EXP and a gain of 60% for I=43 users when matched against PF-BF (Fig. 3.d). So, regardless the traffic type, the obtained policies show considerable benefits by increasing the time when the GBR, Delay and PLR objectives are satisfied.

C. General Remarks

The RL controller behaves as a black box and its performance cannot be checked analytically before simulating the scheduler. The efficiency of the proposed controller in real practice depends on: training data set, data pre-processing,

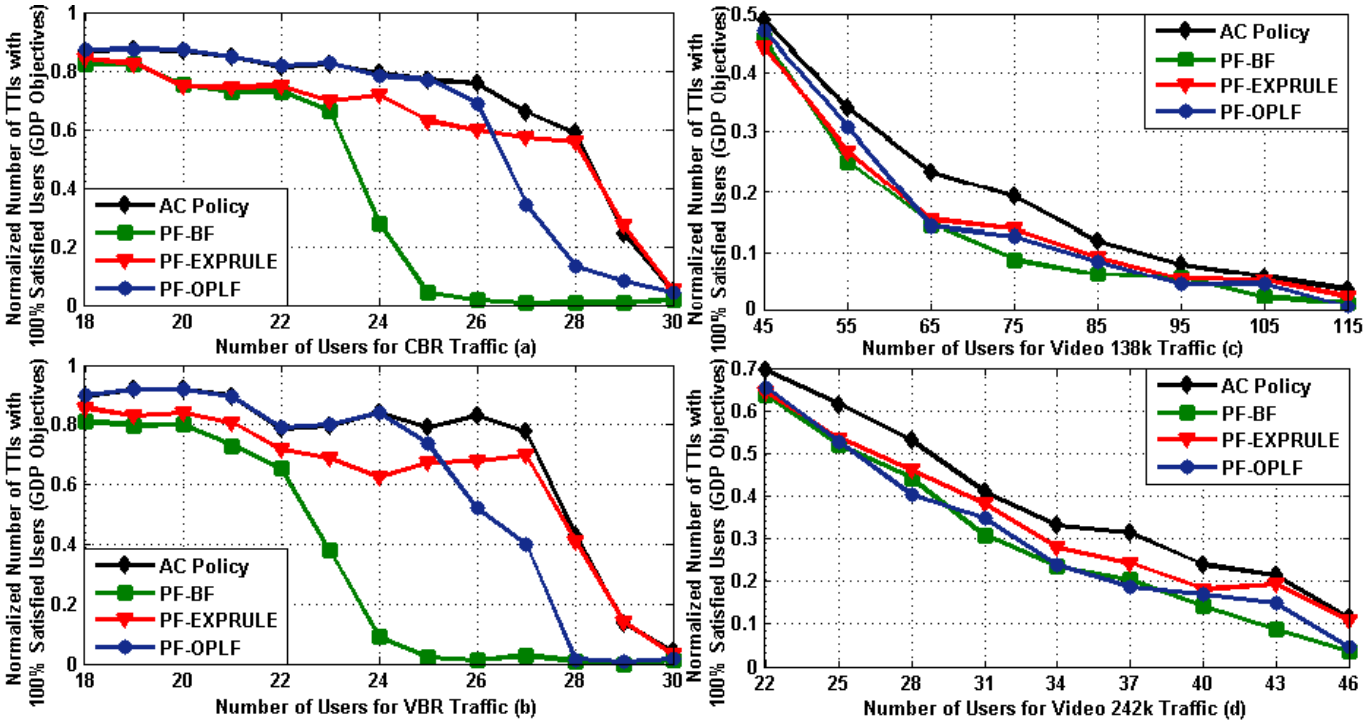

Fig. 3 Performance of AC Policies for Different Traffic Types

controller setting and learning termination. The training data set must be collected from different regions of the state space to avoid the local minimum problems when updating the NN weights. The input data should be pre-processed properly to reduce the system complexity. The optimal setting of RL controller is based on a priori simulations. The termination condition must monitor the NN errors in the training process. We shown that 10^7ms of training is enough when three rules approximate the scheduler state. If the number of rules increases, then additional time for training is required to minimize the NN errors. For other 5G technologies (i.e. mmWave, new waveform), the controller must be re-trained.

VI. Conclusions

In this paper, we propose an innovative RL framework for 5G scheduling that is able to select at each TTI the best scheduling rule to maximize the satisfaction of PLR, GBR and Delay objectives. We proved that, the state and state-action functions depend on the considered QoS indicators and these functions can be very well represented by using the non-linear approximations of neural networks. We proposed the use of the back-propagation method with the gradient descent principle in order to refine the sets of weights. In this sense, the actor-critic RL is used to calculate the necessary errors to be back-propagated for better generalizations. The proposed learning method needs only $10^6 ms$ to converge the policies for CBR, VBR and video traffic types. When exploiting the obtained policies, the proposed strategy is able to significantly increase the time when all QoS objectives are satisfied.

References

[1] Next Generation Mobile Networks (NGMN) Alliance, "5G White Paper,"February, 2015.

[2] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What Will 5G Be?"*IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065 – 1082, 2014.

[3] P. Agyapong, M. Iwamura, D. Staehle, W. Kiess, and A. Benjebbour, "Design Considerations For a 5G Network Architecture,"*in IEEE Communications Magazine*, vol. 52, no. 11, pp. 65 – 75, 2014.

[4] F. Capozzi, G. Piro, L. A. Grieco, G. Boggia, and P. Camarda, "Downlink Packet Scheduling in LTE Cellular Networks: Key Design Issues and a Survey,"*in IEEE Communications Surveys Tutorials*, vol. 15, no. 2, pp. 678 – 700, 2013.

[5] B. Sadiq, R. Madan, and A. Sampath, "Downlink Scheduling for Multiclass Traffic in LTE,"*in EURASIP Journal on Wireless Communications and Networking*, vol. 2009, no. 14, pp. 1–18, 2009.

[6] W. Chung, C. J. Chang, and L. Wang, "An Intelligent Priority Resource Allocation Scheme for LTE-A Downlink Systems,"*in IEEE Wireless Communications Letters*, vol. 1, no. 3, pp. 241 – 244, 2012.

[7] M. Bkassiny, Y. Li, and S. K. Jayaweera, "A Survey on Machine-Learning Techniques in Cognitive Radios,"*in IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1136 – 1159, 2013.

[8] I.-S. Comsa, S. Zhang, M. Aydin, J. Chen, P. Kuonen, and J.-F. Wagen, "Adaptive Proportional Fair Parameterization Based LTE Scheduling Using Continuous Actor-Critic Reinforcement Learning,"in *IEEE GLOBECOM*, Dec. 2014, pp. 4387–4393.

[9] A. De Domenico, V. Savin, D. Ktenas, and A. Maeder, "Backhaul-Aware Small Cell DTX based on Fuzzy Q-Learning in Heterogeneous Cellular Networks,"in *IEEE International Conference on Communications (ICC)*, 2016, pp. 1 – 6.

[10] O. C. Iacoboaiea, B. Sayrac, S. B. Jemaa, and P. Bianchi, "SON Coordination in Heterogeneous Networks: A Reinforcement Learning Framework,"*in IEEE Transactions on Wireless Communications*, vol. 15, no. 9, pp. 5835 – 5847, 2016.

[11] E. Ghadimi, F. D. Calabrese, G. Peters, and P. Soldati, "A Reinforcement Learning Approach to Power Control and Rate Adaptation in Cellular Networks,"in *IEEE International Conference on Communications (ICC)*, 2017.

[12] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients,"*in IEEE Trans. on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291 – 1307, 2012.

[13] C. Szepesvri, *Algorithms for Reinforcement Learning: Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan and Claypool Publishers, 2010.

[14] 3GPP, *Technical Specification Group Services and System Aspects; Policy and charging control architecture Release 12, v.12.2.0*, 2013.

[15] G. Piro, L. A. Grieco, G. Boggia, F. Capozzi, and P. Camarda, "Simulating LTE Cellular Systems: An Open-Source Framework,"*in IEEE Transactions on Vehicular Technology*, vol. 60, no. 2, pp. 498 – 513, 2011.