Towards 5G: A Reinforcement Learning-Based Scheduling Solution for Data Traffic Management

Ioan-Sorin Comşa, Sijing Zhang, Mehmet Emin Aydin, *Senior Member, IEEE*, Pierre Kuonen, Yao Lu, Ramona Trestian, *Member, IEEE*, and Gheorghiţă Ghinea, *Member, IEEE*

***Abstract*—Dominated by delay-sensitive and massive data applications, radio resource management in 5G access networks is expected to satisfy very stringent delay and packet loss requirements. In this context, the packet scheduler plays a central role by allocating user data packets in the frequency domain at each predefined time interval. Standard scheduling rules are known limited in satisfying higher quality of service (QoS) demands when facing unpredictable network conditions and dynamic traffic circumstances. This paper proposes an innovative scheduling framework able to select different scheduling rules according to instantaneous scheduler states in order to minimize the packet delays and packet drop rates for strict QoS requirements applications. To deal with real-time scheduling, the reinforcement learning (RL) principles are used to map the scheduling rules to each state and to learn when to apply each. Additionally, neural networks are used as function approximation to cope with the RL complexity and very large representations of the scheduler state space. Simulation results demonstrate that the proposed framework outperforms the conventional scheduling strategies in terms of delay and packet drop rate requirements.**

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

I. Introduction

THE ENVISIONED applications in the Fifth Generation (5G) of Mobile Technologies (e.g., traffic safety, control of critical infrastructure and industry processes, 50+ Mbps everywhere [1]) impose more stringent QoS requirements like very low end-to-end latency, ultra high data rates, and consequently, very low packet loss rates [2]. To cope with these challenges, access networks should be able to support advanced waveform technologies, mass-scale antennas and flexible Radio Resource Management (RRM) [3]. Alongside standard RRM functions (i.e., power control, interference management, mobility control, resource allocation, packet scheduling [4]), a flexible RRM involves more dynamic functionalities able to adapt to unpredictable network conditions. Some studies have shown an increased interest of integrating Machine Learning (ML) methodologies to *learn* the optimal RRM strategies based on some centralized user-centric (i.e., channel conditions, QoS parameters) and network-centric (traffic routing) data [5].

In the context of RRM, the packet scheduler is responsible for sharing the disposable spectrum of radio resources at each Transmission Time Interval (TTI) between active users with heterogeneous applications and QoS requirements [6]. The prioritized set of active users to be served at each TTI depends on the type of scheduling rule that is implemented. Different rules may perform differently in terms of packet delay and Packet Drop Rate (PDR) requirements according to various scheduling conditions. For example, the scheduling strategy in [7] minimizes the drop rates at the cost of system throughput degradation. The scheduling rules proposed in [8] improve the packet delays with no guarantees on the PDR performance. Another rule in [9] minimizes packet drops at the expense of higher packet delays when compared with other scheduling strategies. However, most of the proposed rules provide unsatisfactory performance when both delay and PDR objectives are considered concomitantly.

Being motivated by this fundamental drawback of conventional scheduling strategies and considering the requirements of 5G networks that need to cater for applications with strict QoS constraints, we propose a flexible RRM packet scheduler able to adapt based on dynamic scheduling conditions. Instead of using a single scheduling rule across the entire transmission, the proposed framework combines multiple scheduling rules TTI-by-TTI in order to improve the satisfaction of stringent QoS requirements in terms of both packet delay and PDR objectives. To make this solution tractable in real time scheduling, our approach must decide the strategy to be applied at each TTI based on momentary conditions, such as: dynamic traffic load, QoS parameters, and application requirements.

One solution is to use the Reinforcement Learning principle as part of the ML domain to learn the scheduling rule selection

Manuscript received December 15, 2017; revised April 11, 2018 and June 24, 2018; accepted July 22, 2018. Date of publication August 6, 2018; date of current version December 10, 2018. This work was supported by the Framework of the Horizon 2020 Project NEWTON (ICT-688503) with funds from the European Union. The associate editor coordinating the review of this paper and approving it for publication was W. Kellerer. *(Corresponding author: Ioan-Sorin Comşa.)*

I.-S. Comşa and G. Ghinea are with the Department of Computer Science, Brunel University London, Uxbridge UB8 3PH, U.K. (e-mail: ioan-sorin.comsa@brunel.ac.uk; george.ghinea@brunel.ac.uk).

S. Zhang is with the School of Computer Science and Technology, University of Bedfordshire, Luton LU1 3JU, U.K. (e-mail: sijing.zhang@beds.ac.uk).

M. E. Aydin is with the Department of Computer Science and Creative Technologies, University of the West of England, Bristol BS16 1QY, U.K. (e-mail: mehmet.aydin@uwe.ac.uk).

P. Kuonen is with the Department of Communications and Information Technology, University of Applied Sciences of Western Switzerland, CH-1700 Fribourg, Switzerland (e-mail: pierre.kuonen@hefr.ch).

Y. Lu is with the Department of Informatics, University of Fribourg, CH-1700 Fribourg, Switzerland (e-mail: yao.lu@unifr.ch).

R. Trestian is with Design Engineering and Mathematics Department, Middlesex University, London NW4 4BT, U.K. (e-mail: r.trestian@mdx.ac.uk).

Digital Object Identifier 10.1109/TNSM.2018.2863563

in each instantaneous scheduler state in order to improve the users' performance measure for delay and PDR objectives when compared to other state-of-the-art scheduling strategies. The RL framework aims to learn the best action to be applied at each state of an unknown environment by keeping track of some state-action values that are updated accordingly at every state-to-state iteration [10]. If these state-action values cannot be enumerated exhaustively, then the optimality of such decisions is not guaranteed [11]. In this paper, we aim to select at each state a scheduling rule from the finite and discrete space of actions. Even so, the selection of the *best* rule is not guaranteed since the scheduler state space (i.e., channel conditions, QoS parameters) is infinite, continuous, multidimensional and stochastic, and the scheduling problems cannot be enumerated exhaustively. Thus, we can only approximate the most convenient rule to be applied at each scheduler state.

RL decisions can be approximated by using parameterized functions [12]. In our approach, each scheduling rule is represented by an individual function and the RL algorithm is used to update these functions TTI-by-TTI until learning criteria are fulfilled. Parameterized functions are used to rank the scheduling rule to be applied in each instantaneous state. The scheduler state space needs to be pre-processed in order to reduce the complexity of the proposed RL framework.

A. Paper Contributions

The 5G networks bring the promise of very high data rates and extremely low latencies to enable the support for advanced applications with stringent QoS requirements. However, this is not possible to achieve through the classical methods of RRM, and the integration of ML is seen as a promising solution. In this context, we propose a RL-based optimization mechanism for RRM to enable efficient resource allocation and strict QoS provisioning, bringing us a step closer to 5G. The approach makes use of dynamic scheduling rule selection at each TTI for OFDMA-based downlink access systems. The choice of OFDMA is due to its simplicity and efficiency as well as its wide deployment placing it among the candidate multiple access schemes to be considered in 5G networks [13].

The contributions of this paper are divided in four parts:

1) Flexible RRM Packet Scheduler: We propose a dynamic RRM scheduler able to select, at each TTI, appropriate scheduling rules according to the momentary network conditions and QoS requirements. The obtained results show significant gains in terms of both delay and PDR satisfaction.

2) RL-based Framework: Here, a RL algorithm is used to learn non-linear functions that approximate the scheduling rule decision at each TTI based on the instantaneous scheduler state. To evaluate the performance of different RL algorithms, five RL algorithms were selected and implemented. Their performance was tested in terms of variable window size, traffic type, objective and dynamic network conditions.

3) Neural Networks (NNs) Based Rule Selection: NNs are non-linear functions that take as input the instantaneous scheduler state and output the preference values of selecting each scheduling rule on that state. Neural networks are used to deal with the continuous and multidimensional scheduler state space.

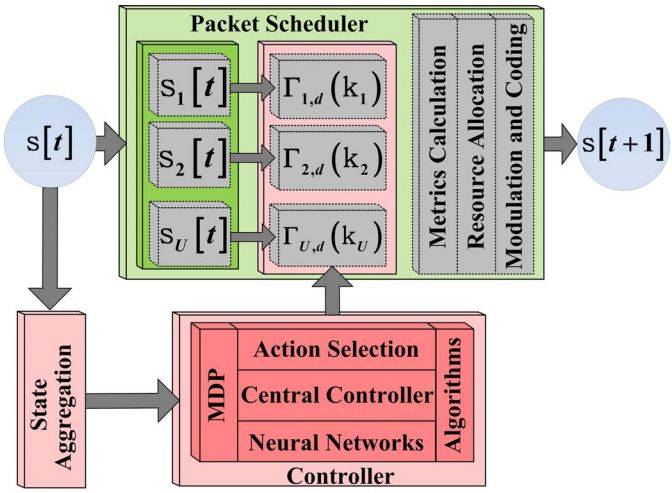


Fig. 1. Proposed System Model.

4) Scheduler State Space Compression Technique: This technique aims to reduce the scheduler state space and speed-up the learning procedure when refining the NNs' weights. In this paper, the focus of the compression procedure is on packet delay and PDR as Key Performance Indicators (KPIs).

The objective of the proposed RL framework is to improve the satisfaction of heterogeneous delay and PDR requirements when scheduling Constant Bit Rate (CBR) and Variable Bit Rate (VBR) traffic types. The CBR and VBR traffic characteristics were chosen in such a way as to cover a wide range of applications (e.g., video, VoIP, FTP, Web browsing) and create a more realistic environment with dynamic channel conditions and traffic loads. By building this dynamic and realistic environment, we can evaluate the stability of the learned policies with different RL algorithms.

B. Paper Organization

The remainder of this paper is organized as follows: Section II introduces the system model. Section III highlights the preliminaries for the RL framework. Section IV details the implemented RL algorithms and the NN function representation. The performance of the obtained RL framework is evaluated in Section V, and Section VI presents the related work. Finally, Section VII concludes the paper.

II. System Model

In the proposed system model presented in Fig. 1, an intelligent controller decides the rule to be applied by the packet scheduler at each TTI. We consider an OFDMA downlink transmission, where the available bandwidth is divided in Resource Blocks (RBs). Let us consider the set of RBs for a given bandwidth as $\mathcal{B} = \{1, 2, \ldots, B\}$, where B is the total number of RBs. Additionally, we consider an User Equipment (UE) being characterized by VBR and CBR traffic types with heterogeneous delay and PDR requirements. Also, at each predefined number of TTIs, an UE is able to change its status (idle/active), data rates and QoS requirements. Let us decide that $\mathcal{U}_t = \{1, 2, \ldots, U_t\}$ is the set of active users at TTI t, where U_t is the number of active users.

The packet scheduler (Fig. 1) aims to allocate a set of RBs $b \in \mathcal{B}$ to user $u \in \mathcal{U}_t$ in such a way that delay and PDR satisfaction is maximized. We consider the set of objectives $\mathcal{O} = \{o_1, o_2\}$ to be satisfied at each TTI *t*, where $o_1 = DELAY$ and $o_2 = PDR$. For each user $u \in \mathcal{U}_t$, we define the online KPI $k_{o,u}[t]$ corresponding to objective $o \in \mathcal{O}$ and its corresponding requirement as $\bar{k}_{o,u}[t]$. For user $u \in \mathcal{U}_t$ the objective $o \in \mathcal{O}$ is satisfied at TTI *t* if and only if the KPI $k_{o,u}[t]$ respects its requirement $\bar{k}_{o,u}[t]$. Both, delay and PDR objectives are satisfied when $\mathbf{k}_u[t] = [k_{o_1,u}, k_{o_2,u}]$ respects the requirement vector $\bar{\mathbf{k}}_u[t] = [\bar{k}_{o_1,u}, \bar{k}_{o_2,u}]$. Globally, the proposed solution aims to increase the percentages of TTIs for all active users when the KPI vector $\mathbf{k}[t] = [\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_{U_t}]$ respects the requirement vector $\bar{\mathbf{k}}[t] = [\bar{\mathbf{k}}_1, \bar{\mathbf{k}}_2, \ldots, \bar{\mathbf{k}}_{U_t}]$, and consequently, both network objectives are satisfied.

Let us consider the discrete set of scheduling rules as $\mathcal{D} = \{1, 2, \ldots, D\}$, where each rule $d \in \mathcal{D}$ has a particular impact in satisfying objectives $\{o_1, o_2\} \in \mathcal{O}$ for certain network and traffic conditions. We define for each rule $d \in \mathcal{D}$ its associated concave and monotone utility function $\Gamma_{d,u}(\mathbf{k}_u) : \mathbb{R}^2 \rightarrow \mathbb{R}$ [14]. The selected utility function $\Gamma_{d,u}$ takes as input the KPI vector $\mathbf{k}_u$ for user $u \in \mathcal{U}_t$ and claims its priority to be served at TTI *t*. Globally, if the same utility Γ_d is used for all active users at each TTI *t*, the KPI vector $\mathbf{k}$ satisfies its requirements $\bar{\mathbf{k}}$ in a certain measure. The proposed flexible RRM scheduler is able to properly choose the utility function Γ_d at each state in order to increase the number of TTIs when $\mathbf{k}$ satisfies $\bar{\mathbf{k}}$.

At the packet scheduler level, $B * U_t$ ranking values are calculated at each TTI in the metrics calculation block, as indicated in Fig. 1. The allocation of RBs involves the selection for each RB $b \in \mathcal{B}$, the user with the highest priority calculated according to the selected utility function $\Gamma_{d,u}$. Then, a proper Modulation and Coding Scheme (MCS) is assigned for the set of allocated RBs of each selected user at each TTI.

A. Problem Formulation

The aim of the proposed solution is to apply at each TTI, the best scheduling rule $d \in \mathcal{D}$ so that as many as possible KPI parameters $\mathbf{k}[t]$ will respect their requirements $\bar{\mathbf{k}}[t]$. Alongside the simple resource allocation problem, the proposed optimization problem to be solved is more challenging since the rule assignment is required at each TTI, such as:

$$\max_{x,y} \sum_{d\in\mathcal{D}} \sum_{u\in\mathcal{U}_t} \sum_{b\in\mathcal{B}} x_{d,u}[t] \cdot y_{u,b}[t] \cdot \Gamma_{d,u}(\mathbf{k}_u[t]) \cdot \gamma_{u,b}[t], \quad s.t. \tag{1}$$

$$\sum\nolimits_u y_{u,b}[t] \leq 1, \quad b = 1, \ldots, B, \tag{1.a}$$

$$\sum\nolimits_d x_{d,u}[t] = 1, \quad u = 1, \ldots, U_t, \tag{1.b}$$

$$\sum\nolimits_u x_{d^*,u}[t] = 1, \quad d^* \in \mathcal{D}, \tag{1.c}$$

$$\sum\nolimits_u x_{d^{\otimes},u}[t] = 0, \quad \forall d^{\otimes} \in \mathcal{D} \backslash d^*, \tag{1.d}$$

$$x_{d,u}[t] \in \{0, 1\}, \quad \forall d \in \mathcal{D}, \forall u \in \mathcal{U}_t, \tag{1.e}$$

$$y_{u,b}[t] \in \{0, 1\}, \quad \forall u \in \mathcal{U}_t, \forall b \in \mathcal{B}, \tag{1.f}$$

where, $\gamma_{u,b}[t]$ is the achievable rate of user $u \in \mathcal{U}_t$ for RB $b \in \mathcal{B}$ at TTI *t*, being calculated as: $\gamma_{u,b}[t] = N^{bits}_{u,b}[t]/0.001$ [15], where, $N^{bits}_{u,b}[t]$ is the maximum number of bits that could be sent if RB $b \in \mathcal{B}$ would be allocated to user $u \in \mathcal{U}_t$. According to [15], $N^{bits}_{u,b}[t]$ is determined as follows: a) at each TTI, the Channel Quality Indicator (CQI) is received for each RB $b \in \mathcal{B}$ and user $u \in \mathcal{U}_t$; b) a MCS scheme for each RB $b \in \mathcal{B}$ and UE $u \in \mathcal{U}_t$ is associated based on CQI; c) using a mapping table $N^{bits}_{u,b}[t]$ is determined based on MCS.

In the maximization problem, $x_{d,u}[t]$ is the scheduling rule assignation variable (i.e., $x_{d,u}[t] = 1$ when the scheduling rule $d \in \mathcal{D}$ is assigned to user $u \in \mathcal{U}_t$, and $x_{d,u}[t] = 0$, otherwise). The RB allocation variable is $y_{u,b}[t] = 1$ when the RB $b \in \mathcal{B}$ is allocated to user $u \in \mathcal{B}$, and $y_{u,b}[t] = 0$, otherwise. When $y_{u,b}[t] = 1$, user $u \in \mathcal{U}_t$ is selected such that $u = argmax[\Gamma_{d,i}(\mathbf{k}_i[t]) \cdot \gamma_{i,b}[t]]$, where $i \in \mathcal{U}_t$. The same procedure is repeated for all RBs from $\mathcal{B}$, until the RBs allocation is finished at TTI *t*. The constraints in (1.a) indicate that at most one user is allocated to resource RB $b \in \mathcal{B}$ (if the data queue is empty and the scheduling rule does not consider this aspect). One RB cannot be allocated to more than one user, but one user can get more than one allocated RB. Constraints in (1.b) associate for each user a single scheduling rule, and the constraints in (1.c) and (1.d) indicate that the same rule $d^* \in \mathcal{D}$ is selected for the entire set of active users at TTI *t*.

The solution to the optimization problem in (1) aims to find at each TTI the best scheduling decision $x_{d,u}[t]$ and resource variable $y_{u,b}[t]$ for all users $u \in \mathcal{U}_t$ and RBs $b \in \mathcal{B}$ such that the utilization of resources $\mathcal{B}$ is fully exploited and the satisfaction of objectives $\mathcal{O}$ is maximized. Although the PDR objective is correlated with the packet delay, we propose a novel strategy in such a way that: (a) *Delay-based Non-congested Case:* the delay requirements can be satisfied for most of the active users if proper rules are applied at each TTI; thus, delay minimization represents the primary objective; (b) *Delay-based Congested Case:* may appear when the KPI vector $\mathbf{k}_{o_1}[t] = \{k_{o_1,1}[t], k_{o_1,2}[t], \ldots, k_{o_1,U_t}[t]\}$ is not able to reach the delay requirements anymore and then, we aim to minimize the PDR KPI vector $\mathbf{k}_{o_2}[t] = \{k_{o_2,1}[t], k_{o_2,2}[t], \ldots, k_{o_2,U_t}[t]\}$; in this case, PDR minimization is the primary objective.

The proposed solution is able to detect both cases by considering the multi-objective performance measure or the reward value which is reported at each TTI by the RRM entity.

B. Problem Solution

The constraints in (1.e) and (1.f) make the optimization problem combinatorial. The rule assignment and RBs allocation must be jointly performed in order to keep the linearity of the problem. Moreover, the scheduler has to disseminate which objective to follow according to the delay-based congested and non-congested cases. To solve such a complex aggregate problem, we propose the use of RL framework that is able to interact with the RRM scheduler as indicated in Fig. 1. The RL controller learns to take proper scheduling decisions based on momentary network conditions. This stage is entitled *learning*. Then, the *exploitation* stage evaluates what the

controller has learned. Both learning and exploitation stages are managed by a central controller. In order to deal with the optimization problem complexity and large input state, the RL controller engine requires the use of neural networks. In the learning stage, the neural networks are adapted to output better scheduling decisions. The RL algorithms indicate here different ways of updating the NN weights. In Section III, the preliminary elements of RL framework are presented and Section IV elaborates the insights of the RL controller.

III. Preliminaries on RL Framework

The RL framework is used to solve the stochastic and multi-objective optimization problem by learning the approximation of policy of rules that can be applied in real time scheduling to improve the multi-objective satisfaction measure. At TTI t, the RL controller observes the current state and takes an action. At TTI $t+1$, a new scheduler state is observed and the reward value is calculated to evaluate the performance of the action performed in the previous state. The reward function together with the scheduler state enhance the decision of the RL controller on the delay-based congestion or non-congestion phases. The previous state, previous action, reward, current state, current action are stored in the Markov Decision Process (MDP) module at the level of RL controller, as shown in Fig. 1. The RL controller explores many state-to-state iterations to optimize the approximation of the best scheduling decisions.

We use neural networks to approximate the scheduling rule selection at every momentary state. With each scheduling rule, we associate a neural network for its approximation. Instead of using a single neural network to represent all scheduling rules at once, we propose a distributed architecture of NNs in order to reduce the framework complexity. At each state, the set of NNs provides D output values. In the learning stage, the action selection block may choose to improve or to evaluate the NNs outputs according to some probabilities. If the evaluation step is chosen, then the scheduling rule with the highest NN output is selected. Otherwise, the improvement step selects a random rule. The NN weights are updated at each TTI based on the tuple stored in MDP and the type of RL algorithm. During the exploitation, only the evaluation steps are applied.

A. Scheduler State Space

Let us define the finite and measurable set for the scheduler state space as $\mathcal{S} = \mathcal{S}^U \cup \mathcal{S}^C$, where $\mathcal{S}^U$ and $\mathcal{S}^C$ are the uncontrollable and controllable sub-spaces, respectively. The uncontrollable sub-space $\mathcal{S}^U$ cannot be predicted whereas $\mathcal{S}^C$ evolves according to the selected rules at every TTI. Let us further define the instantaneous scheduler state at TTI t as a vector: $\mathbf{s}[t] = [\mathbf{c}[t], \mathbf{z}[t]]$, where $\mathbf{s}[t] \in \mathcal{S}$, $\mathbf{c}[t] \in \mathcal{S}^C$ and $\mathbf{z}[t] \in \mathcal{S}^U$. The uncontrollable elements at TTI t, $\mathbf{z} \in \mathcal{S}^U$ are: CQI reports, number of active users at TTI t, the arrival rates in data queues and the vector of KPI requirements $\bar{\mathbf{k}}[t]$. The controllable sub-state at TTI t, $\mathbf{c} \in \mathcal{S}^C$ is denoted by $\mathbf{c} = [\mathbf{k}, \lambda, \underline{\mathbf{k}}, \mathbf{q}]$, where $\lambda[t] = [\lambda_1, \lambda_2, \ldots, \lambda_{U_t}]$ is the vector of user data rates being scheduled, $\underline{\mathbf{k}}[t] = [\underline{\mathbf{k}}_1, \underline{\mathbf{k}}_2, \ldots, \underline{\mathbf{k}}_{U_t}]$ comprises the differences between the momentary KPI values $k_{o,u}$ and their requirements $\bar{k}_{o,u}$, and $\mathbf{q}[t] = [q_1, q_2, \ldots, q_{U_t}]$ is the vector of queue lengths. For each user $u \in \mathcal{U}_t$, the controllable elements $\underline{\mathbf{k}}_u[t] = [k_{o,u} - \bar{k}_{o,u}]$, enable the RL controller to notify when objectives $\{o_1, o_2\} \in \mathcal{O}$ are satisfied.

B. Action Space

We define the finite action set as $\mathcal{A} = \{a_1, a_2, \ldots, a_D\}$, where D is the number of scheduling rules. When the RL controller selects action $a[t] = d$ at TTI t, the RBs allocation is performed and the system moves into the next state $\mathbf{s}' = \mathbf{s}[t+1] \in \mathcal{S}$ according to the following transition function:

$$\mathbf{c}'_{(d)} = f(\mathbf{s}, d), \tag{2}$$

where $\mathbf{c}'_{(d)} = [\mathbf{k}'_{(d)}, \lambda'_{(d)}, \underline{\mathbf{k}}'_{(d)}, \mathbf{q}'_{(d)}] \in \mathcal{S}^C$ is the expected controllable set at TTI $t+1$ when applying the scheduling rule $a[t] = d$ in state $\mathbf{s}[t] \in \mathcal{S}$. The new state $\mathbf{s}' \in \mathcal{S}$ is obtained based on the uncontrollable elements $\mathbf{z}' = \mathbf{z}[t+1] \in \mathcal{S}^U$.

C. Reward Function

As per the original definition [10], the reward represents the expected goodness of applying action $a[t] = d$ in state $\mathbf{s} \in \mathcal{S}$:

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

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

Theorem 1: For any action $a[t] = d$ applied in state $\mathbf{s}[t] = \mathbf{s}$, the reward function will depend on controllable elements from the current and next states, such as: $r(\mathbf{s}, d) = r(\mathbf{c}'_{(d)}, \mathbf{c}, d)$.

Proof 1: The proof is provided in Appendix A. ■

The role of *Theorem 1* within the RL framework aims to simplify the reward function calculation and to eliminate the dependency on uncontrollable CQIs. In the absence of *Theorem 1*, additional pre-processing steps are necessary to compress the CQI sub-space, which in fact, increases the complexity of the entire RL framework.

The reward function can be further simplified if we consider that, at TTI $t+1$, the future controllable elements are already known, and consequently, we can say that, $\mathbf{c}' = \mathbf{c}'_{(d)}$. Then, the proposed reward function is calculated as follows:

$$r(\mathbf{c}', \mathbf{c}) = \sum\nolimits_{n=1}^{2} \delta_{o_n} \cdot r_{o_n}(\mathbf{c}'_{o_n}, \mathbf{c}_{o_n}), \tag{4}$$

where $\delta_{o_n} \in \mathbb{R}_{[0,1]}$ represents the reward weights, where $\delta_{o_1} + \delta_{o_2} = 1$, $\mathbf{c}_{o_n} = [\mathbf{k}_{o_n}, \lambda, \underline{\mathbf{k}}_{o_n}, \mathbf{q}]$ and $\underline{\mathbf{k}}_{o_n} = [k_{o_n,u} - \bar{k}_{o_n,u}]$. The weights δ_{o_n} must stay constant during the entire learning stage to ensure the convergence of the learned policies [10]. Each sub-reward function in (4) is calculated based on:

$$r_{o_n}(\mathbf{c}'_{o_n}, \mathbf{c}_{o_n}) = \begin{cases} 1, & \{r^+_{o_n}(\mathbf{c}'_{o_n}), r^+_{o_n}(\mathbf{c}_{o_n})\} = 1 \\ r^+_{o_n}(\mathbf{c}'_{o_n}) - r^+_{o_n}(\mathbf{c}_{o_n}), & otherwise. \end{cases} \tag{5}$$

The reward expressed above shows the temporal difference in performance for delay and PDR objectives. The proposed sub-reward functions $r^+_{o_n} : \mathbb{R}^{4 \cdot U_t} \rightarrow \mathbb{R}$ are determined according to: $r^+_{o_n}(\mathbf{c}_{o_n}) = 1/U_t \cdot \sum_{u=1}^{U_t} r^-_{o_n}(\mathbf{c}_{o_n,u})$,

where the controllable vector for user $u \in \mathcal{U}_t$ is $\mathbf{c}_{o_n,u} = [k_{o_n,u}, \lambda_u, \underline{k}_{o_n,u}, q_u]$ and,

$$r^-_{o_n}(\mathbf{c}_{o_n,u}) = \begin{cases} 1 - \frac{\underline{k}_{o_n,u}}{k_{o_n,u}}, & \underline{k}_{o_n,u} \geq 0, \{q_u, \lambda_u\} \neq 0, \\ 1, & otherwise. \end{cases} \quad (6)$$

Basically, when both QoS requirements of all users are satisfied, the reward is 1. Otherwise, the rewards are moderate ($r \geq 0$) or punishments ($r < 0$). We set the delay requirements $\bar{k}_{o_1}$ at lower values than initially proposed by 3GPP [16]. In this way, the scheduler is able to provide much lower packet delays than the conventional scheduling approaches.

D. Value and Action-Value Functions

Let us define $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ the policy function that maps states to distributions over the action space [17]. In the context of our scheduling problem, we denote the stochastic policy $\pi(d \mid \mathbf{s})$ as the probability of action $a[t] = d$ being selected by π in state $\mathbf{s}[t] = \mathbf{s}$ [17], that is defined as follows:

$$\pi(d \mid \mathbf{s}) = \mathbb{P}[a[t] = d | \mathbf{s}[t] = \mathbf{s}]. \quad (7)$$

Additionally, we define the value function $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$ that measures the value of state $\mathbf{s}$ underlying π and defined as [17]:

$$V^\pi(\mathbf{s}) \overset{(\mathbf{def})}{=} \mathbb{E}_\pi\Big[\sum\nolimits_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} | \mathbf{s}[0] = \mathbf{s}\Big], \quad (8)$$

where, (1) the process $(\gamma^t \mathcal{R}_{t+1}; t \geq 0)$ is the accumulated reward value being averaged from state to state by the discount factor $\gamma \in [0, 1]$; (2) $\mathbf{s}[0]$ is considered as random such that $\mathbb{P}(\mathbf{s}[0] = \mathbf{s}) > 0$ holds for every $\mathbf{s} \in \mathcal{S}$. The second condition makes the expectation in (8) defined for all states in $\mathcal{S}$. If we also assume that the first action $a[0]$ of the whole process is randomly chosen such that $\mathbb{P}(a[0] = d) > 0$ holds for all rules $d \in \mathcal{D}$ while the following action decisions follow π, then the action-value function $Q^\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is defined as [17]:

$$Q^\pi(\mathbf{s}, d) \overset{(\mathbf{def})}{=} \mathbb{E}_\pi\Big[\sum\nolimits_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} | \mathbf{s}[0] = \mathbf{s}, a[0] = d\Big]. \quad (9)$$

Both value and action-value functions defined in (8) and (9), respectively, consider as argument the general state representation as defined in Section III-A. These functions need to be redefined and adapted to our scope since the reward in (4) takes as input the consecutive controllable sub-states.

Theorem 2: For any policy π that optimizes (1), we have the new value function $K^\pi : \mathcal{S}^C \times \mathcal{S}^C \rightarrow \mathbb{R}$ determined as:

$$K^\pi(\mathbf{c}', \mathbf{c}) = \mathbb{E}_\pi\Big[\sum\nolimits_{t=1}^{\infty} \gamma^{t-1} \mathcal{R}_t | \mathbf{c}[1] = \mathbf{c}', \mathbf{c}[0] = \mathbf{c}\Big], \quad (10)$$

and $J^\pi : \mathcal{S}^C \times \mathcal{S}^C \times \mathcal{A} \rightarrow \mathbb{R}$ is the new action-value function:

$$J^\pi(\mathbf{c}', \mathbf{c}, d) = \mathbb{E}_\pi\Big[\sum\nolimits_{t=1}^{\infty} \gamma^{t-1} \mathcal{R}_t | \mathbf{c}[1] = \mathbf{c}', \mathbf{c}[0] = \mathbf{c}, a[1] = d\Big], \quad (11)$$

where the new policy $\pi[d \mid (\mathbf{c}', \mathbf{c})]$ states the probability of selecting rule $d \in \mathcal{D}$ when the current state is $(\mathbf{c}', \mathbf{c})$.

Proof 2: The proof can be found in Appendix B. ■

According to *Theorem 2*, the proposed RL framework learns based on the consecutive controllable states, while eliminating the dependency on other un-controllable elements. This is in fully congruence with *Theorem 1* with no need for additional steps to compress the CQI uncontrollable states.

By considering the relations in (8), (10) and (9), (11), respectively, the value and action-value functions can be decomposed according to the temporal difference principle:

$$K^\pi(\mathbf{c}', \mathbf{c}) = r(\mathbf{c}', \mathbf{c}, d) + \gamma \cdot K^\pi(\mathbf{c}'', \mathbf{c}'), \quad (12a)$$

$$J^\pi(\mathbf{c}', \mathbf{c}, d) = r(\mathbf{c}', \mathbf{c}, d) + \gamma \cdot K^\pi(\mathbf{c}'', \mathbf{c}'), \quad (12b)$$

where $\mathbf{c}'' = \mathbf{c}[t+2]$ and the reasonings behind above equations are given in Appendix C.

The optimal value $K^*(\mathbf{c}', \mathbf{c})$ of state $(\mathbf{c}', \mathbf{c}) \in \mathcal{S}^C \times \mathcal{S}^C$ is the highest expectable return when the entire scheduling process is started from state ($\mathbf{c}'$, $\mathbf{c}$). Then, function $K^* : \mathcal{S}^C \times \mathcal{S}^C \rightarrow \mathbb{R}$ is the optimal value function determined as: $K^*(\mathbf{c}', \mathbf{c}) = max_\pi K^\pi(\mathbf{c}', \mathbf{c})$ [17]. Similarly, the optimal action-value $J^*(\mathbf{c}', \mathbf{c}, d)$ of pair $(\mathbf{c}', \mathbf{c}, d)$ represents the highest expected return when the scheduling process starts from state ($\mathbf{c}'$, $\mathbf{c}$) and the first selected action is $a[1] = d$. Consequently, $J^*(\mathbf{c}', \mathbf{c}, d) : \mathcal{S}^C \times \mathcal{S}^C \times \mathcal{A} \rightarrow \mathbb{R}$ is the optimal action-value function. If we consider that the RL controller acts as optimal at each state, the selection of the best scheduling rule is achieved according to the following equation:

$$d^* = \underset{d' \in \mathcal{D}}{argmax}\big[\pi(d' \mid (\mathbf{c}', \mathbf{c})\big]. \quad (13)$$

In this case, both value and action-value functions are optimal, and relations (12a) and (12b), respectively, become:

$$K^*(\mathbf{c}', \mathbf{c}) = r(\mathbf{c}', \mathbf{c}, d) + \gamma \cdot K^*(\mathbf{c}'', \mathbf{c}'), \quad (14)$$

$$J^*(\mathbf{c}', \mathbf{c}, d) = r(\mathbf{c}', \mathbf{c}, d) + \gamma \cdot K^*(\mathbf{c}'', \mathbf{c}'). \quad (15)$$

From Appendix B, it can be easily seen that $K^*(\mathbf{c}'', \mathbf{c}') = max_{d' \in \mathcal{D}} J^*(\mathbf{c}'', \mathbf{c}', d')$. Then, both optimal value and action-value functions can be derived as follows:

$$K^*(\mathbf{c}', \mathbf{c}) = r(\mathbf{c}', \mathbf{c}, d) + \gamma \cdot \max_{d' \in \mathcal{D}} J^*(\mathbf{c}'', \mathbf{c}', d'), \quad (16)$$

$$J^*(\mathbf{c}', \mathbf{c}, d) = r(\mathbf{c}', \mathbf{c}, d) + \gamma \cdot \max_{d' \in \mathcal{D}} J^*(\mathbf{c}'', \mathbf{c}', d'). \quad (17)$$

According to the target values calculated based on (14)-(17) that we would like to achieve at each TTI, the RL framework parameterizes the non-linear functions. Each of these functions defines the type of RL algorithm. We consider the evaluation of each RL algorithm in order to find the best policy for each parameterization schemes used to compute the on-line PDRs.

For our stochastic optimization problem, the optimality of value and action-value functions is not guaranteed. Then, we aim to find the approximations of these functions, such that: $\bar{K}^*(\mathbf{c}', \mathbf{c}) \approx K^*(\mathbf{c}', \mathbf{c})$ and $\bar{J}^*(\mathbf{c}', \mathbf{c}, d) \approx J^*(\mathbf{c}', \mathbf{c}, d)$ for all $d \in \mathcal{D}$. Also, the instantaneous state $(\mathbf{c}', \mathbf{c}) \in \mathcal{S}^C \times \mathcal{S}^C$ needs to be pre-processed to reduce the RL framework complexity.

IV. Proposed RL Framework

A. State Compression

We aim to solve the dimensionality and variability problems of controllable states by eliminating the dependence on the number of active users U_t. This is fundamental for our RL framework, since the input state needs to have a

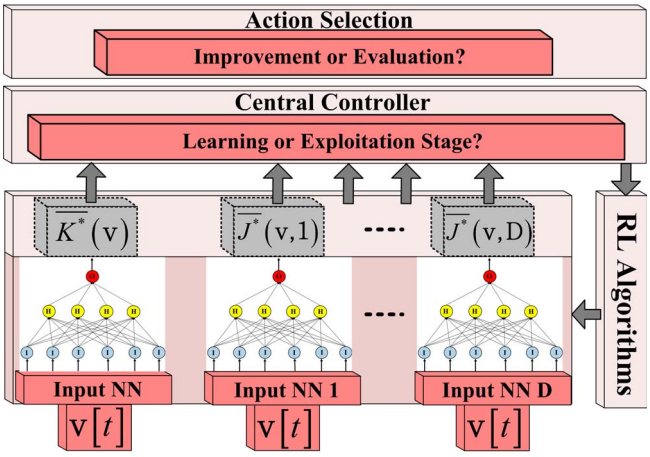


Fig. 2. Proposed RL Framework.

fixed dimension in order to update the same set of non-linear functions.

Theorem 3: At each TTI, the controllable states $\mathbf{c} \in \mathcal{S}^C$ can be modeled as normally distributed variables.

Proof 3: We group the controllable elements as follows: $\mathbf{c} = \{\mathbf{k}_{o_1}, \mathbf{k}_{o_2}, \lambda, \underline{\mathbf{k}}_{o_1}, \underline{\mathbf{k}}_{o_2}, \mathbf{q}\}$, where $\mathbf{c} = \{\mathbf{c}_n\}$, $n = \{1, \ldots, 6\}$. Each component depends on the number of users, such as: $\mathbf{c}_n = \{c_{n,u}\}$, where $u = \{1, 2, \ldots, U_t\}$. When n is fixed, each element $c_{n,u}$ can be normalized at each TTI t as follows:

$$\hat{c}_{n,u} = c_{n,u} / \left(1/U_t \cdot \sum\nolimits_{u'}^{U_t} c_{n,u'}\right). \tag{18}$$

By expanding (18) and fractioning between the pairs, we get the following recurrence relation:

$$\hat{c}_{n,u} = \left(c_{n,u} \cdot c_{n,u+1} / \hat{c}_{n,u+1}\right). \tag{19}$$

The normalized set $\hat{\mathbf{c}}_n = \{\hat{c}_{n,1}, \hat{c}_{n,2}, \ldots, \hat{c}_{n,U_t}\}$, $\forall n$ is normally distributed, if and only if, each element $\hat{c}_{n,u}$ is a product of random variables. Equation (19) proves the theorem. ■

We use the means and standard deviations to represent the distributions of the normalized controllable and semi-controllable elements based on maximum likelihood estimators [18]. Let us define the mean function $\mu(\hat{c}_n) : \mathbb{R}^{U_t} \rightarrow [-1, 1]$ and the standard deviation function $\sigma(\hat{c}_n) : \mathbb{R}^{U_t} \rightarrow [-1, 1]$. Based on maximum likelihood estimators [18], these functions are defined as follows:

$$\mu(\hat{c}_n) = 1/U_t \cdot \sum\nolimits_{u=1}^{U_t} ln\left(\hat{c}_{n,u}\right), \tag{20a}$$

$$\sigma(\hat{c}_n) = \sqrt{1/U_t \cdot \sum\nolimits_{u=1}^{U_t} \left[ln\left(\hat{c}_{n,u}\right) - \mu(\hat{c}_n)\right]^2}. \tag{20b}$$

The same principle for calculating the normalized values and the mean and standard deviations is used for next-state controllable elements $\hat{\mathbf{c}}' = \{\hat{\mathbf{c}}'_n\}$. To simplify the controllable state representation, we define the 24-dimensional vector as $\mathbf{v} \in \hat{\mathcal{S}}$, where $\mathbf{v} = [\mu(\hat{\mathbf{c}}'_n), \sigma(\hat{\mathbf{c}}'_n), \mu(\hat{\mathbf{c}}_n), \sigma(\hat{\mathbf{c}}_n)]$, and $n = \{1, \ldots, 6\}$.

B. Approximations of Value and Action-Value Functions

Figure 2 shows the insights of the proposed RL framework. Alongside a number of D neural networks used to approximate the action-value functions, we need an additional neural network to represent the value function. We approximate the optimal action-value functions by defining the function $\bar{J}^* : \hat{\mathcal{S}} \times \mathcal{A} \rightarrow \mathbb{R}$. Also, we define $\bar{K}^* : \hat{\mathcal{S}} \rightarrow \mathbb{R}$ as a function approximator for the optimal value function. Then, the non-linear representations of these functions are defined as follows:

$$\begin{aligned} \bar{K}^*(\mathbf{v}) &= h(\theta_t, \psi(\mathbf{v})), \\ \bar{J}^*(\mathbf{v}, d) &= h^d\left(\theta_t^d, \psi(\mathbf{v})\right), \end{aligned} \tag{21}$$

where, $\{h, h^1, h^2, \ldots, h^D\}$ are the neural networks used to approximate the value and action-value functions, respectively; $\psi(\mathbf{v})$ is the feature vector, and $\{\theta, \theta^1, \theta^2, \ldots, \theta^D\}$ is the set of weights that has to be tuned.

The NN structure is based on two off-line parameterizations: the number of layers and the number of nodes for each layer. Let us define L the number of NN layers and N_l the number of nodes for each layer $l \in \{1, 2, \ldots, L\}$. If the number of nodes for the input and output layers are known in advance (i.e., $N_1 = 24$, and $N_L = 1$), the number of hidden layers $L - 2$ and the number of nodes for each hidden layer must be determined in advance based on a priori testing.

The weights $\{\theta, \theta^1, \theta^2, \ldots, \theta^D\}$ are used to interconnect the nodes from layer to layer. Let us consider $\mathbf{W}_l = \{w_{b,m}, b = 1, \ldots, N_l, m = 1, \ldots, N_{l+1}\}$ the matrix of weights between layers l and $l + 1$. The total number of weights that has to be tuned in the learning stage between layers l and $l + 1$ is $(N_l + 1) \times N_{l+1}$. The compressed controllable states $\mathbf{v}$ are passed from layer to layer and they go through a set of non-linear transformations. The output of layer l, becomes [19]:

$$\mathbf{v}^{(l+1)} = \psi_{l+1}\left(\mathbf{W}_l^T \times \mathbf{v}_+^{(l)}\right), \tag{22}$$

where, $\mathbf{v}_+^{(l)}$ is the biased input state and ψ_{l+1} is the activation function of layer $l + 1$. On the largest scale, the compressed state is propagated through the entire NN according to [19]:

$$\bar{K}^*(\mathbf{v}) = \psi_L\left(\mathbf{W}_{L-1}^T \cdot \cdots \cdot \psi_{l+1}\left(\mathbf{W}_l^T \cdot \cdots \cdot \psi_2\left(\mathbf{W}_1^T \cdot \mathbf{v}\right)\right)\right). \tag{23}$$

Similarly, the controllable state $\mathbf{v} \in \hat{\mathcal{S}}$ is forwarded through all state-action NNs $\{h^1, h^2, \ldots, h^D\}$. The activation function $\psi_l = (\psi_{l,1}, \psi_{l,2}, \ldots, \psi_{l,N_l})$ is element-wise and the same function is considered for all nodes. The main idea is to learn $D + 1$ vectors of weights, but at each TTI t, only two NNs are updated $(\theta_t, \theta_t^{d*})$, and $a[t] = d^*$ is the rule applied in state $\mathbf{v}$.

For NN learning purposes, we consider the current state as $\mathbf{v}' = [\mu(\hat{\mathbf{c}}''_n), \sigma(\hat{\mathbf{c}}''_n), \mu(\hat{\mathbf{c}}'_n), \sigma(\hat{\mathbf{c}}'_n)] \in \hat{\mathcal{S}}$ and $\mathbf{v} \in \hat{\mathcal{S}}$ as a previous state. We aim to update the set of weights $\{\theta_t, \theta_t^{d*}\}$ by reinforcing the error values that are able to evaluate the performance of selecting the output of NN $d^* \in \mathcal{D}$ in state $\mathbf{v} \in \hat{\mathcal{S}}$ when the current state is $\mathbf{v}' \in \hat{\mathcal{S}}$. Let us define the value error function $\mathbf{e} : \mathbb{R}_{[-1,1]} \rightarrow \mathbb{R}_{[-1,1]}$ and the action-value error function of NN $d \in \mathcal{D}$ as $\mathbf{e}^d : \mathbb{R}_{[-1,1]} \rightarrow \mathbb{R}_{[-1,1]}$. These errors are calculated at each TTI based on the following equations:

$$\mathbf{e}_t\left(\theta_{t-1}, \mathbf{v}', \mathbf{v}\right) = K^T\left(\mathbf{v}', \mathbf{v}\right) - \bar{K}^*(\mathbf{v}), \tag{24a}$$

$$\mathbf{e}_t^d\left(\theta_{t-1}^d, \mathbf{v}', \mathbf{v}\right) = J^T\left(\mathbf{v}', \mathbf{v}, d\right) - \bar{J}^*(\mathbf{v}, d), \tag{24b}$$

where, $K^T : \hat{\mathcal{S}} \times \hat{\mathcal{S}} \rightarrow \mathbb{R}$ is the target value function defined based on (14) or (16) and $J^T : \hat{\mathcal{S}} \times \hat{\mathcal{S}} \times \mathcal{A} \rightarrow \mathbb{R}$ is the target action-value function calculated according to (15) or (17).

Both errors $\{\mathbf{e}_t, \mathbf{e}_t^d\}$ are back-propagated through the neural networks from layer to layer. Let us define the vector of value errors $\mathbf{E}^{(l)} = [\mathbf{e}_1^{(l)}, \mathbf{e}_2^{(l)}, \ldots, \mathbf{e}_{N_l}^{(l)}]$ being back-propagated to the output of layer $l \in \{1, 2, \ldots, L\}$. These errors are back-propagated based on the following equation [19]:

$$\mathbf{E}^{(l)} = \mathbf{W}_l^T \times \Delta^T\Big(\Psi'_{l+1}, \mathbf{E}^{(l+1)}\Big), \tag{25}$$

where, $\Psi'_{l+1} = [\psi'_{l+1,1}, \psi'_{l+1,2}, \ldots, \psi'_{l+1,N_{l+1}}]$ is the derivative set and $\Delta[\Psi'_{l+1}, \mathbf{E}^{(l+1)}] = [\psi'_{l+1,1} \cdot \mathbf{e}_1^{(l+1)}, \psi'_{l+1,2} \cdot \mathbf{e}_2^{(l+1)}, \ldots, \psi'_{l+1,N_{l+1}} \cdot \mathbf{e}_{N_{l+1}}^{(l+1)}]$. By using (25), the errors are back-propagated from layer to layer and the weights are updated each time based on the gradient descent principle. Then, the weight $w_{b,m}^t$ that interconnects node $b = 1, \ldots, N_l$ of layer l to node $m = 1, \ldots, N_{l+1}$ of layer $l+1$ at TTI t is updated according to the following formula [19]:

$$w_{b,m}^t = w_{b,m}^{t-1} + \eta_t \cdot v_b^{(l)} \cdot \psi'_{l+1,m} \cdot e_m^{(l+1)}, \tag{26}$$

where η_t is the learning rate, $v_b^{(l)}$ is the state element and $\psi'_{l+1,m}$ is the derivative function on $m \in \{1, 2, \ldots, N_{l+1}\}$.

C. RL Algorithms

A set of RL algorithms is used to update the approximations of optimal value and action-value functions. Among all RL algorithms, only five are investigated and used to reinforce the corresponding errors and optimize the NNs $\{h, h^1, h^2, \ldots, h^D\}$ based on dynamic network and traffic conditions.

QV-learning [20] combines the value and action-value functions to build its policy by considering a two-step updating process based on (14) and (15), respectively:

$$K^T\big(\mathbf{v}', \mathbf{v}\big) = r(\mathbf{v}, d^*) + \gamma \cdot \bar{K}^*\big(\mathbf{v}'\big), \tag{27a}$$

$$J^T\big(\mathbf{v}', \mathbf{v}, d^*\big) = r(\mathbf{v}, d^*) + \gamma \cdot \bar{K}^*\big(\mathbf{v}'\big), \tag{27b}$$

where, $a[t] = d^*$ is the action applied in the previous state $\mathbf{v} \in \hat{\mathcal{S}}$ and $r(\mathbf{v}, d^*)$ is the reward function calculated based on (4). The errors are calculated according to (24a) and (24b).

QV2-learning [21] keeps the same form of target functions as exposed in (27) with the only difference that, the value function error is back-propagated as follows:

$$\mathbf{e}_t\big(\theta_{t-1}, \mathbf{v}', \mathbf{v}\big) = K^T\big(\mathbf{v}', \mathbf{v}\big) - \bar{J}^*(\mathbf{v}, d^*). \tag{28}$$

QVMAX-learning [21] sets the error calculations and the target function $J^T(\mathbf{v}', \mathbf{v}, d^*)$ similar to the QV-learning. The only difference is the target value function, such as:

$$K^T\big(\mathbf{v}', \mathbf{v}\big) = r(\mathbf{v}, d^*) + \gamma \cdot \max_{d' \in \mathcal{D}} \bar{J}^*\big(\mathbf{v}', d'\big). \tag{29}$$

QVMAX2-learning [21] is a combination of QV, QV2 and QVMAX algorithms. The target action-value function $J^T(\mathbf{v}', \mathbf{v}, d^*)$ is defined similar to QV-learning as in (27b), the target value function $K^T(\mathbf{v}', \mathbf{v})$ is determined according to the QVMAX rule as in (29), the value error $\mathbf{e}_t(\theta_t, \mathbf{v}', \mathbf{v})$ is similar to QV2 and the action-value error $\mathbf{e}_t^{d^*}(\theta_t^{d^*}, \mathbf{v}', \mathbf{v})$ is determined similar to QV-learning.

For the *Actor Critic Learning Automata (ACLA)* [22], at each TTI, the value function $\bar{K}^*$ is updated according to (27a) and its error is determined based on (24a). If the value error $\mathbf{e}_t(\theta_t, \mathbf{v}', \mathbf{v})$ is positive, the action $d^* \in \mathcal{D}$ in state $\mathbf{v} \in \hat{\mathcal{S}}$ was a good choice and the probability of selecting that action in the future for the same approximated state should be increased. Otherwise, the probability of selecting that action is decreased. The target action-value function is determined as follows:

$$J^T\big(\mathbf{v}', \mathbf{v}, d^*\big) = \begin{cases} 1, & if\ \mathbf{e}\big(\theta_{t-1}, \mathbf{v}', \mathbf{v}\big) \geq 0, \\ -1, & if\ \mathbf{e}\big(\theta_{t-1}, \mathbf{v}', \mathbf{v}\big) < 0. \end{cases} \tag{30}$$

The action selection function from Fig. 2 plays a central role in the learning stage. The trade-off for improvement/evaluation steps is decided by $\epsilon-$greedy or Boltzmann distributions. If the $\epsilon-$greedy is decided to be used, the action $a[t+1]$ in state $\mathbf{v}'$ is selected based on the following policy [19]:

$$\pi\big(d \mid \mathbf{v}'\big) = \begin{cases} \epsilon_t^{(d)} & \epsilon \geq \epsilon_t, \\ h^d\Big(\theta_t^d, \psi\big(\mathbf{v}'\big)\Big] & \epsilon < \epsilon_t, \end{cases} \tag{31}$$

where $\epsilon_t^{(d)}$ is a random variable and ϵ_t is time-based parameter that decides when the improvement or the evaluation step is applied. If ϵ_t is very low, then we have more improvements steps. When ϵ_t gets higher values, the RL controller aims to exploit the output of NNs more. However, the $\epsilon-$exploration is not able to differentiate between the potentially good and worthless actions for given momentary states. The Boltzmann exploration takes into account the values of NNs at each TTI, in which, the actions with higher NNs values should have higher probabilities to be selected and the others will be neglected. The potentially good actions for the momentary state $\mathbf{v}' \in \hat{\mathcal{S}}$ are detected by using the following formula [19]:

$$\pi\big(d \mid \mathbf{v}'\big) = \frac{exp\Big[h^d\Big(\theta_t^d, \psi\big(\mathbf{v}'\big)\Big)/\tau\Big]}{\sum_{d'=1}^{D} exp\big[h^{d'}\big(\theta_t^{d'}, \psi(\mathbf{v}')\big)/\tau\big]}, \tag{32}$$

where τ is the temperature factor that sets how greedy the policy is. For instance, when $\tau \rightarrow 0$, the exploration is more greedy, and thus, the NNs with the highest outputs are selected. When $\tau \rightarrow \infty$, the action selection becomes more random, and thus, all actions have nearly the same selection probabilities. Regardless of the type of exploration that is used, the action is selected at each TTI according to (13). Algorithm 1 summarizes the introduced concepts and reasonings.

V. Simulation Results

The proposed framework was implemented in a RRM-Scheduler C/C++ object oriented tool that inherits the LTE-Sim simulator [15]. For the performance evaluation, an infrastructure of 10 Intel(R) 4-Core(TM) machines with i7-2600 CPU at 3.40GHz, 64 bits, 8GB RAM and 120 GB HDD Western Digital storage was used. The entire framework was simulated using the same network conditions for both learning and exploitation stages. The obtained results are averaged over 10 simulation runs and the STandard

Algorithm 1 RRM Scheduler Based on RL Algorithms

for each TTI t
  observe state $(\mathbf{c}'', \mathbf{c}') \in \mathcal{S}^C \times \mathcal{S}^C$, apply the compression functions based on (18), (19), (20a), and (20b), get $\mathbf{v}\prime \in \hat{\mathcal{S}}$.
  recall the previous state and action ($\mathbf{v}$, d), $d \in \mathcal{D}$ and **store** the actual state $\mathbf{v}' \in \hat{\mathcal{S}}$ at the controller MDP level
  calculate reward $r(\mathbf{v}, d^*)$ based on (4), (5) and (6).
  forward propagate states ($\mathbf{v}'$, $\mathbf{v}$) on $\bar{K}^*(\cdot) = h(\theta_{t-1}, \psi(\cdot))$ according to (22) and (23)
  forward propagate state $\mathbf{v}$ on $\bar{J}^*(\mathbf{v}, d) = h^d(\theta^d_{t-1}, \psi(\mathbf{v}))$, $d \in \mathcal{D}$ based on (22) and (23)
  if QV algorithm
    calculate value error $\mathbf{e}_t(\theta_{t-1}, \mathbf{v}', \mathbf{v})$ - (27a) and (24a)
    calculate error $\mathbf{e}^d_t(\theta^d_{t-1}, \mathbf{v}', \mathbf{v})$ - (27b) and (24b)
  if QV2 algorithm
    calculate value error $\mathbf{e}_t(\theta_{t-1}, \mathbf{v}', \mathbf{v})$ - (27a) and (28)
    calculate error $\mathbf{e}^d_t(\theta^d_{t-1}, \mathbf{v}', \mathbf{v})$ - (27b) and (24b)
  if QVMAX algorithm
    calculate value error $\mathbf{e}_t(\theta_{t-1}, \mathbf{v}', \mathbf{v})$ - (29) and (24a)
    calculate error $\mathbf{e}^d_t(\theta^d_{t-1}, \mathbf{v}', \mathbf{v})$ - (27b) and (24b)
  if QVMAX2 algorithm
    calculate value error $\mathbf{e}_t(\theta_{t-1}, \mathbf{v}', \mathbf{v})$ - (29) and (28)
    calculate error $\mathbf{e}^d_t(\theta^d_{t-1}, \mathbf{v}', \mathbf{v})$ - (27b) and (24b)
  if ACLA algorithm
    calculate value error $\mathbf{e}_t(\theta_{t-1}, \mathbf{v}', \mathbf{v})$ - (27a) and (24a)
    calculate error $\mathbf{e}^d_t(\theta^d_{t-1}, \mathbf{v}', \mathbf{v})$ - (30) and (24b)
  back propagate error $\mathbf{e}_t(\theta_{t-1}, \mathbf{v}', \mathbf{v})$ based on (25)
  update weights θ_{t-1} based on (26)
  back propagate error $\mathbf{e}^d_t(\theta^d_{t-1}, \mathbf{v}', \mathbf{v})$ based on (25)
  update weights θ^d_{t-1} based on (26)
  // **act** based on the learned policy
  apply $d^* = argmax_{d' \in \mathcal{D}}[\pi(d' \mid \mathbf{v}')]$ based on (31) or (32).
end for

Deviations (STDs) are analyzed in order to prove the veracity of proposed policies. In order to study the impact of the online PDR in the learned policies, different averaging settings are considered.

The aim of the simulations is two-fold: (a) to study the learning performance of five different RL algorithms (QV, QV2, QVMAX, QVMAX2, ACLA) under different traffic types, varying window size, objectives, and dynamic traffic conditions; (b) to study the performance of the proposed RL-based framework and learned policies vs. classical scheduling rules under different objectives, traffic types and network conditions. For the purpose of the performance evaluation, the proposed RL-based framework considers the set of state-of-the-art scheduling rules consisting of EXPonential 1 (EXP1) rule [7], EXPonential 2 (EXP2) and LOGarthmic (LOG) strategies [8] and Earliest Deadline First (EDF) rule [9].

A. Parameter Settings

For the purpose of the simulations, our system model considers the bandwidth of 20 MHz (100 RBs) and the ARQ scheme with maximum 5 retransmissions. Packets failing to be transmitted within this interval are declared lost. Since the packet loss rate is related more to the network conditions, we focus only on the ratio of dropped packets which is related more to scheduler performance. The online PDR KPI $k_{o_2,u}[t]$ for each user $u \in \mathcal{U}_t$ is calculated as follows: $k_{o_2,u}[t] = (\sum_{z=t}^{T} \bar{N}_u[z-t])/(\sum_{z=t}^{T} N_u[z-t])$, where N_u is the number of total transmitted packets and $\bar{N}_u$ is the number of dropped packets being caused by higher packet delays than those ones imposed by 3GPP. Parameter T is the time window that is calculated as the ratio between the total number of active users U_t and the maximum number of users N_s^m that can be scheduled within one TTI. Then, $T = \rho \cdot [U_t / N_s^m]$, where $[\,\cdot\,]$ is the integer part and ρ is the windowing factor. The role of ρ is to ensure the PDR satisfaction when U_t is variable. For instance, we have noticed that, when $U_t > N_s^m$, low windowing factors ρ = [5.5, 200] provides satisfactory performance for the PDR objective. When $U_t <= N_s^m$, the windowing factor can be increased such as ρ = [200, 400] in order to have larger horizons of time when dropping the packets, while the PDR objective is still satisfied. When $\rho > 400$, the PDR performance is seriously degraded. However, based on more general traffic settings (U_t is variable during learning and exploitation stages), we would like to find the most convenient range of ρ such that both packet delay and PDR objectives can be maximized. In this sense, we vary the windowing factor in $\rho = \{5.5, 100, 200, 400\}$ in order to cover a wider range for both aforementioned cases.

In the learning stage, the packet delay and PDR constraints are updated at each 1000 TTIs in the range of $\bar{k}_{o_1,u}[t] = \{50, 100, 150, 200, 250, 300\} ms$ and $\bar{k}_{o_2,u}[t] = \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$, respectively. When the delay exceeds any of these requirements from $\bar{k}_{o_1,u}[t]$, the packets are dropped and declared lost. To obtain better results for the satisfaction of delay, we aim to impose stricter requirements, such that: $\bar{\bar{k}}_{o_1,u}[t] = \upsilon \cdot \bar{k}_{o_1,u}[t]$ and $\upsilon = 0.9$. Packets exceeding these limits are not discarded, and the proposed policies are able to apply the best rule so that the PDR can be much improved. In order to increase the probability of reaching the terminal states $(r(\mathbf{c}', \mathbf{c}) = 1)$ for very high traffic load and low latency requirements, the delay sub-rewards $\{r^+_{o_1}(\mathbf{c}'_{o_1}), r^+_{o_1}(\mathbf{c}_{o_1})\}$ in (5) are modified as follows:

$$r^+_{o_1}(\mathbf{c}_{o_1}) = \begin{cases} 1, & \left[1/U_t \cdot \sum_{u=1}^{U_t} r^-_{o_n}(\mathbf{c}_{o_n,u})\right] \geq \kappa, \\ 1/U_t \cdot \sum_{u=1}^{U_t} r^-_{o_n}(\mathbf{c}_{o_n,u}), & otherwise, \end{cases} \tag{33}$$

where, $\kappa \in [0, 1]$ indicates the acceptable limit such that, for $(1-\kappa)\%$ of users that are in outage of delay requirements, the delay reward is still maximized. For our simulations, we impose $\kappa = 0.9$. When the global reward value is calculated, the same level of importance is given for both delay and PDR objectives, and consequently: $\delta_{o_1} = \delta_{o_2} = 0.5$.

In both learning and exploitation stages, the number of active users is changed every 1000 TTIs in the domain of $U_t = [15,\ 120]$ in order to better illustrate the superiority of the proposed policies. The user speed is 30 kmph and the mobility model is considered to be random direction for both learning and exploitation stages. 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

TABLE I
PACKET SCHEDULER PARAMETERS

Parameter	Value
System Bandwidth/Cell Radius	20 MHz (100 RBs)/1000m
User Speed/Mobility Model	120 Kmph/Random Direction
Channel Model	Jakes Model
Path Loss/Penetration Loss	Macro Cell Model/10dB
Interfered Cells/Shadowing STD	6/8dB
Carrier Frequency/DL Power	2GHz/43dBm
Frame Structure	FDD
CQI Reporting Mode	Full-band, periodic at each TTI
PUCCH Model	Errorless
Scheduler Type	EXP1 [7], EXP2 [8], LOG [8], EDF [9], RL Policies
Traffic Type	CBR, VBR
No. of schedulable users N_s^m	10 each TTI
RLC ARQ	Acknowledged Mode (5 retransmissions)
AMC Levels	QPSK, 16-QAM, 64-QAM
Target BLER	10%
Number of Users ($\|\mathcal{U}_t\|$)	Variable: 15-120
RL Algorithms	QV [20], QV2 [21], QVMAX [21], QVMAX2 [21], ACLA [22]
Exploration/Exploitation Duration	500s/95s
Windowing Factor (ρ)	$\{5.5, 100, 200, 400\}$

TABLE II
CONTROLLER PARAMETERS

RL Algs.	Learning Rate (η_Q)	Learning Rate(η_V)	Discount Factor(γ)	Exploration (ϵ, τ)
QV	10^{-3}	10^{-5}	0.99	$\tau = 10$
QV2	10^{-3}	10^{-5}	0.95	$\tau = 10$
QVMAX	10^{-3}	10^{-5}	0.99	$\tau = 10$
QVMAX2	10^{-3}	10^{-5}	0.95	$\tau = 10$
ACLA	10^{-4}	10^{-4}	0.99	$\epsilon = 5 \cdot 10^{-5}$

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.

CBR and VBR traffic types are considered to model a wide range of applications (e.g., video, VoIP, FTP, Web browsing) with different traffic characteristics. Thus, the CBR traffic is generated based on the following set of arrival rates $\lambda_i[t] = \{32, 64, 128, 256, 512, 1024\}$ randomly generated at each 1000 TTIs (for all active users). The VBR traffic is generated following a Pareto distribution for packet size and geometric distributions for the arrival rates [18]. The obtained policies can provide very high degrees of generalizations, and thus they can be applied to realistic environments. The remaining set of parameters for the RRM packet scheduler is listed in Table I.

B. Optimization of RL Controller

When optimizing the controller, our aim is to find the best parameterization scheme (learning rates, discount factors and exploration parameters) that minimizes the NNs output errors $(\mathbf{e}_t, \mathbf{e}_t^1, \mathbf{e}_t^2, \ldots, \mathbf{e}_t^D)$ for a given duration of the learning stage. Different configurations are simulated and Table II illustrates the most suited parameters for each considered RL algorithm.

The parameterization of neural networks (L, N_l), $l \in \{1, 2, \ldots, L\}$ constitutes another important aspect that has to be considered before launching the learning stage. When the neural network is too flexible (high number of layers and hidden nodes), the complexity is higher, the learning speed slower, and there is a risk to *overfit* the input state in the sense that, the function approximator will represent not only the interest data, but also the noise in the scheduler state [19]. When the neural network is inflexible (insufficient number of layers and hidden nodes), the system complexity is lower, the RL framework can learn faster, and parts of the scheduler state space may not be represented by the approximator. As a consequence, we get poor generalizations and the function approximator is said to *underfit* the input state [19]. In both under-fitting and over-fitting cases, the state error starts to increase inexplicably at a certain point in the learning stage. We carefully tested different NN configurations, such as $L = \{3, 5, 7\}$, where the number of neurons for each hidden layer was varied in the interval of {50,100,150,200}. Considering under-fitting, over-fitting and system complexity trade-off, and based on preliminary simulations we found out that $L = 3$ and $N_2 = 100$ are enough to represent the state space for delay and PDR objectives. For each simulation setting, we considered the same topology of neural networks (i.e., the same numbers of layers, nodes and activation functions). The activation functions for the input and output layers are linear, whilst for hidden nodes, the activation function is tangent hyperbolic [18].

C. Learning Performance

This subsection presents the learning performance of the five considered RL algorithms in terms of the mean percentage of TTIs under punishments and moderate rewards and for varying windowing factor, different objectives (e.g., delay only, PDR only, and both delay and PDR) and different traffic classes (e.g., CBR and VBR). By punishment we understand that the reward is negative at TTI *t*, such that $-1 < r < 0$ while in case of moderate reward we have $0 \leq r < 1$. Thus, for this case, we consider the mean percentage of TTIs under punishments and moderate rewards, such as $\overline{p}(-1 < r < 0; 0 \leq r < 1)$. Then, $\overline{p}_1(-1 < r_{o_1} < 0; 0 \leq r_{o_1} < 1)$ is the mean percentage of TTIs under punishments and moderate rewards for the delay objective; similarly $\overline{p}_2(-1 < r_{o_2} < 0; 0 \leq r_{o_2} < 1)$ is the mean percentage of TTIs corresponding to PDR objective; and, $\overline{p}_{12}(-1 < r < 0; 0 \leq r < 1)$ is the mean percentage of TTIs for both delay and PDR objectives. Figures 3 (a)-(f) illustrate the performance of considered RL algorithms in the exploitation stage when considering the mean percentage of TTIs under moderate rewards and punishments for each traffic class, objective and varying windowing factor.

Figures 3 (a)-(c) show the performance of the RL algorithms in terms of mean percentage of TTIs under punishments and moderate rewards for varying windowing factor and objectives, for the CBR traffic only. We notice that in the case of delay objective, the percentages $\overline{p}_1(-1 < r_{o_1} < 0; 0 \leq r_{o_1} < 1)$ remain relatively constant for each of the RL algorithms when varying the windowing factor. However, it can be observed that ACLA performs better than other choices for $\rho = \{5.5, 100\}$ while QVMAX and QVMAX2 perform better for $\rho = \{200, 400\}$. When considering the

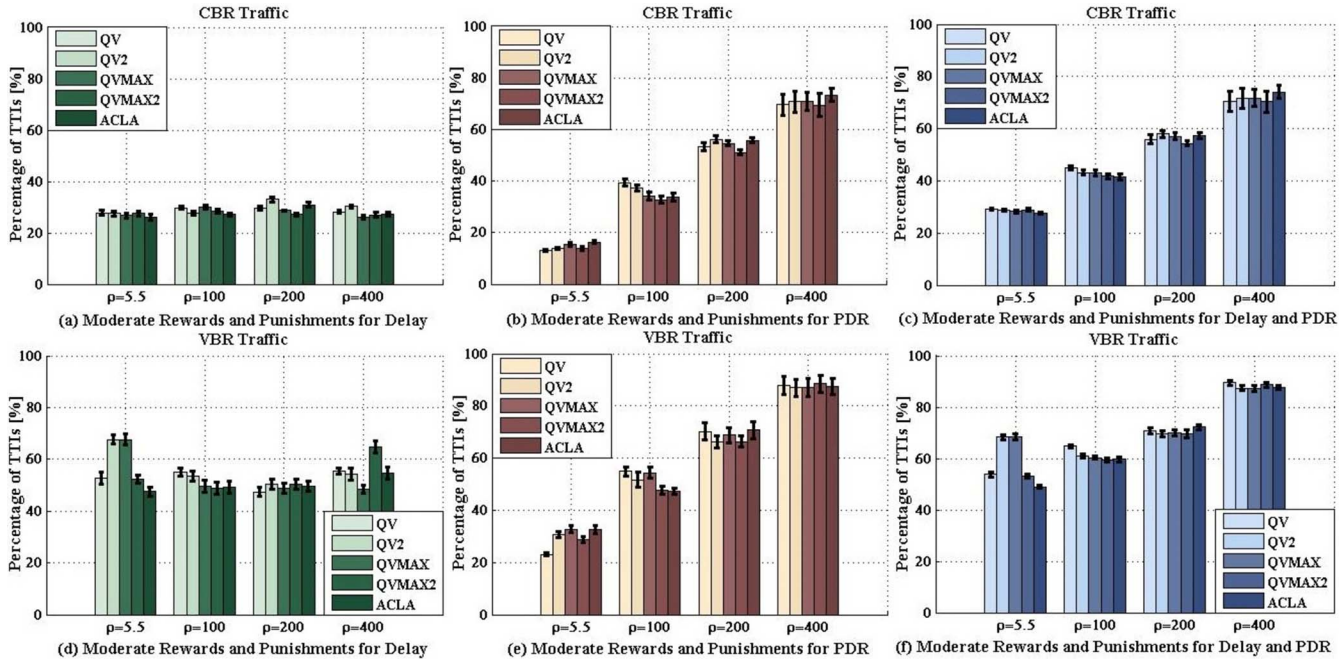


Fig. 3. Learning Performance: Punishment and Moderate Rewards.

PDR objective only (Fig. 3 (b)), the QV policy accumulates the least amount of punishments and moderate rewards for $\rho = \{5.5\}$ and the QVMAX2 algorithm learns the best when $\rho \in \{100, 200, 400\}$. However, when both delay and PDR objectives are considered (Fig. 3 (c)), ACLA and QVMAX2 achieve the lowest mean percentage $\overline{p}_{12}(-1 < r < 0; 0 \leq r < 1)$ when $\rho \in \{5.5, 100, 200\}$ and QVMAX2 is the best choice for $\rho = 400$. We observe that the STD of $\overline{p}_2(-1 < r_{o_2} < 0; 0 \leq r_{o_2} < 1)$ becomes higher as ρ increases. This shows that if very large windows are used in the PDR computations, the policies show their limitations in applying appropriate scheduling rules that can maximize the satisfaction of PDR requirements.

The learning performance when scheduling VBR traffic is highlighted in Figs. 3 (d)-(f). QV, QVMAX2 and ACLA perform best for the delay objective (Fig. 3 (d)) for $\rho = 5.5$. For $\rho = \{100, 200\}$, QVMAX, QVMAX2 and ACLA provide a good performance, while QVMAX minimizes $\overline{p}_1(-1 < r_{o_1} < 0; 0 \leq r_{o_1} < 1)$ when $\rho = 400$. The PDR objective (Fig. 3 (e)) satisfaction is achieved for larger periods of time when $\rho = 5.5$ by the QV policy. However, when increasing the windowing factor, $\rho = \{200, 400\}$, the best candidates are ACLA, QVMAX2 and QV2. When combining both delay and PDR objectives (Fig. 3 (f)), the following policies perform the best: QV, QVMAX2, ACLA for $\rho = 5.5$, ACLA, QVMAX, QVMAX2 for $\rho = 100$, and QVMAX for $\rho = \{200, 400\}$.

Looking at the impact of different traffic classes, by comparing Figs. 3(c) and 3(f), we notice that the RL policies can learn better under VBR traffic, since the STDs are considerably reduced when compared with the CBR traffic. Moreover, as indicated by (33), we aim to maximize the reward when 90% of active users achieve their delay requirements. In this sense, the RL policy that shows the best performance in terms of learning performance, may not be the best option when we measure the network performance for 100% satisfied users.

D. Policies' Performance

The objective of this subsection is to analyze if the considered RL policies are able to ensure the best performance when measuring the objective satisfaction for all active users. In this sense, we measure the mean percentage of TTIs when 100% of active users satisfy: a) the delay requirements only ($\overline{p}_1(100\%)$); b) the PDR requirements only ($\overline{p}_2(100\%)$); and c) both, delay and PDR requirements ($\overline{p}_{12}(100\%)$).

The results are listed in Table III for each considered RL algorithm under a varying windowing factor, different objectives and traffic classes. The top scheduling policies under each objective and for each windowing factor are highlighted. When the results obtained in Table III are compared with Fig. 3, a discrepancy between $\{\overline{p}_1(100\%), \overline{p}_2(100\%), \overline{p}_{12}(100\%)\}$ and $1 - \{\overline{p}_1(-1 < r_{o_1} < 0; 0 \leq r_{o_1} < 1), \overline{p}_2(-1 < r_{o_2} < 0; 0 \leq r_{o_2} < 1), \overline{p}_{12}(-1 < r < 0; 0 \leq r < 1)\}$ can be observed in the sense that even if some RL approaches are able to provide good performance when minimizing the percentage of TTIs with punishment and moderate rewards, the mean percentage of TTIs when all active users are satisfied is seriously degraded. For example, in Fig. 3(d), for $\rho = 5.5$, ACLA aims to minimize the number of punishments and implicitly to maximize the number of maximum rewards when 90% of users are satisfied, whereas, in Table III, QV policy is the best option when measuring $\overline{p}_1(100\%)$. Similarly, in Fig. 3(f), for $\rho = 5.5$, ACLA and QVMAX2 policies are the best options to minimize the amount of punishment and moderate rewards. However, in Table III, the QV policy is the one that maximizes $\overline{p}_{12}(100\%)$. Moreover, in Fig. 3(f), QVMAX achieves similar performance as ACLA and QVMAX2 when $\rho = 100$. In Table III, its

TABLE III
POLICIES PERFORMANCE FOR CBR AND VBR TRAFFIC TYPES

		CBR Traffic				VBR Traffic			
		Mean Percentage of Feasible TTIs [%]				Mean Percentage of Feasible TTIs [%]			
Objectives	**Met.**	$\rho = 5.5$	$\rho = 100$	$\rho = 200$	$\rho = 400$	$\rho = 5.5$	$\rho = 100$	$\rho = 200$	$\rho = 400$
Delay	**QV**	9.367	15.108	27.812	20.808	38.906	38.532	32.942	18.515
	QV2	51.19	60.336	43.71	46.599	1.138	35.296	32.417	36.347
	QVMAX	40.684	58.943	26.662	64.502	0.703	18.528	39.82	41.216
	QVMAX2	66.13	52.252	59.146	65.713	15.545	37.817	18.837	14.03
	ACLA	65.195	62.937	58.538	50.711	32.951	42.338	18.944	31.718
PDR	**QV**	86.786	60.483	46.576	30.338	76.793	44.996	29.657	12.037
	QV2	86.117	62.438	43.788	29.126	69.149	48.176	33.549	13.045
	QVMAX	84.427	65.768	45.25	28.977	67.237	45.577	31.054	12.845
	QVMAX2	85.997	67.166	48.862	30.54	71.194	52.169	33.528	11.402
	ACLA	83.56	66.233	44.048	26.516	67.337	52.569	29.198	12.416
Delay + PDR	**QV**	9.051	8.379	15.902	10.071	38.322	32.296	21.859	0.323
	QV2	50.474	48.815	39.205	25.784	0.971	33.967	18.03	11.771
	QVMAX	39.894	51.355	15.229	25.761	0.54	12.208	26.919	11.7
	QVMAX2	65.455	49.121	36.304	27.642	15.169	31.298	18.837	10.394
	ACLA	64.369	55.376	39.19	17.501	32.395	36.269	11.791	11.556

performance is seriously degraded since $\overline{p}_{12}(100\%) = 12.2$, which is three times less than ACLA.

These discrepancies are obtained since some policies prefer to select those scheduling rules that aim to keep some users in outage in terms of packet delay for longer periods. In Table III, the results show that more than 10% of feasible TTIs for the entire set of RL algorithms and windowing factors are lost when scheduling VBR traffic.

Other discrepancies refer to the performance differences between windowing factors and traffic types. For example, for CBR traffic in the case of the PDR objective when the value of the windowing factor is increased (e.g., 200, 400), QVMAX2 achieves the best performance. Whereas, in the case of VBR traffic, QV2 performs the best under the same settings. Even if the same network and traffic conditions are used when training the NNs under different RL algorithms, the sequence of these conditions differs from one setting of ρ to another. This explains the performance variability of the obtained RL policies under $\rho \in \{5.5, 100, 200, 400\}$ when scheduling CBR and VBR traffic. However, it can be concluded that ACLA is a better option for shorter time windows in PDR computations, whereas the QVMAX and QVMAX2 algorithms can learn better for much longer time windows used in PDR computations.

E. Comparison With State-of-the-Art Strategies

The aim of this subsection is to analyze the performance of the proposed RL-based framework and compare it against the conventional scheduling rules, such as EDF, LOG, EXP1, and EXP2. Through this performance evaluation we want to show that by using only one scheduling rule it cannot fully satisfy the objectives under dynamic network conditions and traffic types. Thus, the proposed RL-based framework will select the most suitable scheduling rule to be applied at each TTI based on the current network conditions. The performance of the proposed RL-based framework and the best policies from Table III are compared against other state-of-the-art scheduling solutions in terms of mean percentage of TTIs when the active users are satisfied in percentage of $q\% = \{90, 92, 94, 96, 98, 100\}$ for different objective targets, such as $\overline{p}_1(q)$, $\overline{p}_2(q)$, $\overline{p}_{12}(q)$. The results are collected for CBR and VBR traffic under the three objectives and varying windowing factor and listed in Fig. 4.

Looking at the delay objective only, it can be easily observed that the proposed framework is able to outperform the classic scheduling rules in terms of $\overline{p}_1(100\%)$ for both cases of CBR (Fig. 4(a)) and VBR traffic (Fig. 4(d)). When scheduling the CBR traffic, more than 10% of feasible TTIs are gained by the proposed framework under all windowing factor settings. Calling the appropriate scheduling rule at each TTI enables all active users to respect the lower delay bound. A degradation of $\overline{p}_1(q)$ can be observed when scheduling the VBR traffic with $\rho = 5.5$ for $q[\%] = \{90, 92, 94\}$. This is because the main purpose of the obtained policy is to minimize the mean delay for all users with the minimum STD delay values. Some scheduling rules aim to keep some users in outage for longer periods by increasing the STD of packet delays.

The PDR objective for both traffic types under an increasing windowing factor ρ, $\overline{p}_2(q)$ decreases and the results' variation becomes larger (Figs. 4(b) and 4(e)). However, the proposed framework works best when $q[\%] = \{90, 92, 94, 96, 98, 100\}$.

When maximizing the percentages of feasible TTIs when all active users are satisfied in terms of both packet delay and PDR requirements, the proposed framework performs the best as seen in Figs 4(c) and 4(f). By selecting appropriate scheduling rules for different traffic loads, network conditions and QoS requirements, the proposed framework gains more than 15% of $\overline{p}_{12}(100\%)$ when compared with classical scheduling rules for the CBR traffic type and the windowing factor of $\rho = \{5.5, 100, 200, 400\}$. For the VBR traffic, the proposed framework indicates a gain of about 10%. This is because some packets have larger sizes when compared with CBR.

F. General Remarks

The performance of the RL controller depends on the following factors: the type of RL algorithm, the input data set being used in the training stage, data processing, controller

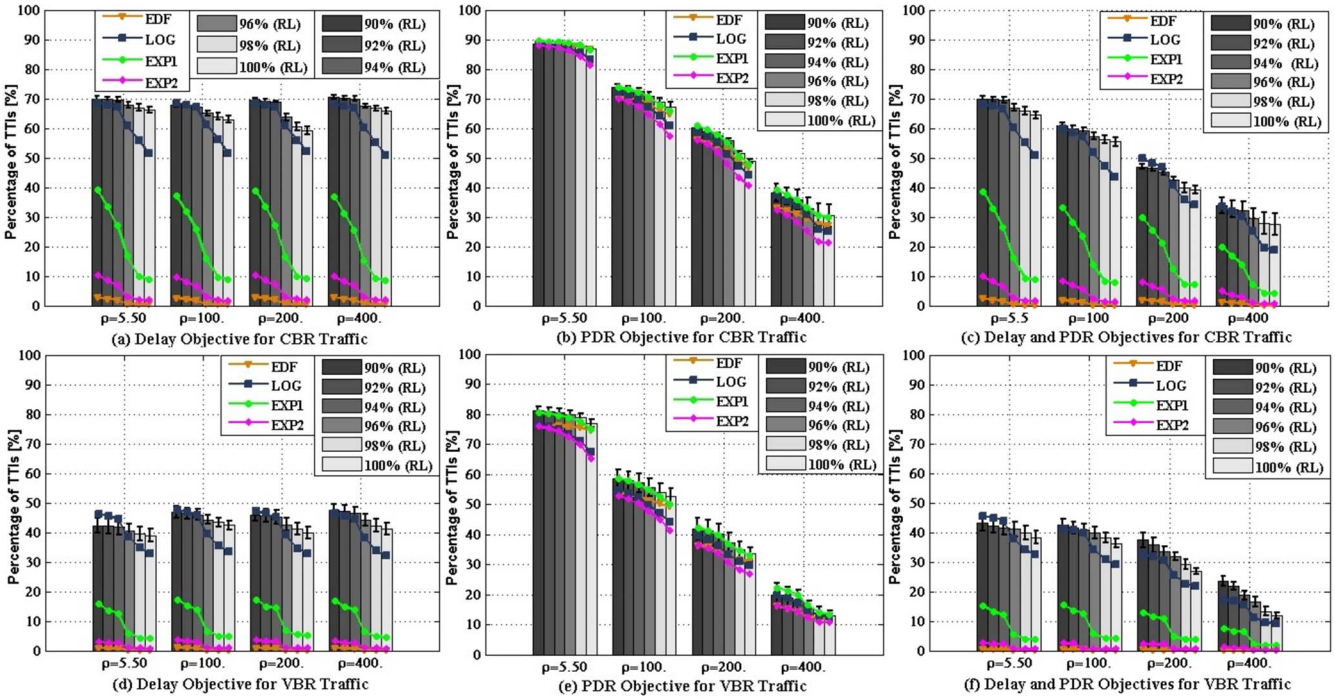


Fig. 4. Exploitation Performance: Percentages of TTIs when Delay, PDR, and both Delay and PDR Objectives are Satisfied.

parameterization and the learning termination condition. If the sequence of provided input data is not similar, the performance of the RL algorithms can differ as we observed in Table III. The training data has to be carefully chosen in order to permit the controller to explore as many states as possible and to avoid the local minima problems. The input observations must be pre-processed before applying to the RL controller in order to avoid the dependency for some parameters that may change over time, such as the number of active users. The controller setting has to be determined a priori by using extensive simulation results. The NN configuration for our scheduling model makes use of $L = 3$ layers, where: $N_1 = 24$, $N_2 = 100$, $N_3 = 1$. Finally, the learning termination condition indicates when the training stage should be stopped. For our simulations, the termination condition is performed after 500s, the moment of time when the errors for all five RL algorithms are nearly the same and the weights of NNs are saved.

VI. Related Work

One key aspect in obtaining optimal performance within the radio access network is the dynamically scheduling of the limited radio resources. Different radio resource allocation strategies and scheduling rules have been proposed in the literature to optimize the distribution of radio resources among different users by considering the dynamic channel conditions as well as QoS requirements. For example, the EXP1 rule proposed in [7] is able to enhance the PDR at the cost of throughput degradation when scheduling video streaming services. Two rules EXP2 and LOG proposed in [8] are able to minimize the overflow of data queues when compared to Modified Largest Weighted Delay First (MLWDF). However, the MLWDF rule provides poor PDR performance when CBR traffic is scheduled [23]. The EDF strategy in [9] outperforms MLWDF, LOG, EXP1, and EXP2 in terms of PDR with delay degradation when higher real-time traffic load is scheduled.

RL has been widely used in RRM decision-making problems, such as: inter-cell interference coordination [24], self-organizing networks [25], energy savings [26], Adaptive Modulation and Coding selection [27], radio resource allocation and packet scheduling [28]–[30]. In many RRM optimization problems, the states (network conditions) and actions (RRM decisions) are continuous and multidimensional variables that increase the complexity of RL algorithms. Different approaches are proposed to avoid these drawbacks.

Clustering methods are used in [31] to convert the continuous state space into its discrete representation. In [26], the discrete state space is achieved through fuzzy logic mechanism by using linguistic variables. Another solution is to integrate RL algorithms with artificial Neural Networks, which are able to approximate non-linear functions that map the continuous state into desired scheduling decisions for the proportional-fair rule parameterization [29], [30]. However, some pre-processing tools are needed to compress the NN input state dimension, and consequently, to speed-up the learning process.

A form of compression is considered in [28], where the modulation and coding scheme is adapted based on average CQI reports received from all users. This method can be used only in wide-band CQI reporting scenarios, becoming automatically unfeasible when the sub-band reporting is required. For the self-organizing mechanism proposed in [25], the state compression considers only the conflicting parameters with neighboring cells. Some approaches consider the division of the multidimensional state into smaller sub-states to be approximated as indicated in [32].

VII. Conclusion

This paper proposes a flexible RRM packet scheduler which is able to adapt based on dynamic scheduling conditions and to enable QoS provisioning. The proposed approach makes use of Reinforcement Learning to determine, for each instantaneous scheduler state, a better scheduling rule to be applied. Additionally, an innovative technique that compresses the controllable momentary state such that the dependency on the number of users is eliminated is also introduced. Through extensive simulation results we have demonstrated that different RL approaches behave differently under varying network conditions and system settings. However, we show that by using the proposed framework together with the best RL policies in the exploitation stage, the proposed RRM scheduler outperforms the classical scheduling rules in terms of both packet delay and PDR objectives.

As part of future work, we plan to investigate the proposed RL framework as a possible solution for the optimization problems that consider non-orthogonal multiple access schemes as well as heterogeneous traffic conditions with strict QoS.

Appendix A
Proof of Theorem 1

The reward function is decomposed as indicated in (34) when starting with the definition in (3), where, the $(*)$ property indicates that, the uncontrollable state $\mathbf{z}[t] = \mathbf{z} \in \mathcal{S}^U$ can

$$\begin{aligned} r(\mathbf{s}, d) &\overset{(3)}{=} \mathbb{E}[\mathcal{R}_{t+1} | \mathbf{s}[t] = \mathbf{s}, a[t] = d] \\ &\overset{(2)}{=} \mathbb{E}[\mathcal{R}_{t+1} | \mathbf{c}[t+1] = f(\mathbf{s}, d), \mathbf{s}[t] = \mathbf{s}, a[t] = d] \\ &= \mathbb{E}\Big[\mathcal{R}_{t+1} | \mathbf{c}[t+1] = \mathbf{c}'_{(d)}, \mathbf{c}[t] = \mathbf{c}, \mathbf{z}[t] = \mathbf{z}, a[t] = d\Big] \\ &\overset{(*)}{=} \mathbb{E}\Big[\mathcal{R}_{t+1} | \mathbf{c}[t+1] = \mathbf{c}'_{(d)}, \mathbf{c}[t] = \mathbf{c}, a[t] = d\Big] \\ &= r\Big(\mathbf{c}'_{(d)}, \mathbf{c}, d\Big). \end{aligned} \tag{34}$$

be reproduced if the controllable elements $\{\mathbf{c}, \mathbf{c}'_{(d)}\} \in \mathcal{S}^C$ from the actual and future scheduler states are known. For instance, by having the tuple $\{\lambda, \lambda'_{(d)}\}$, the effective SINR can be reproduced, and consequently, the CQI reports for each user can be approximated. The KPI requirements at TTI t $\bar{\mathbf{k}}$ are determined based on the controllable elements $\{\mathbf{c}, \mathbf{c}'_{(d)}\}$. The arrival bit rates in data queues at TTI t are obtained based on the differences between consecutive sizes of queues $\mathbf{q}'$ and $\mathbf{q}$. The queue sizes denote here the number of bits from each user's queue being impacted only by the scheduling decision. The arriving bits in data queues depend on the traffic type and are included in the uncontrollable state space. Also, the number of active users U_t at TTI t can be easily determined by simply setting $\lambda_u = 0$ for those users in the IDLE state.

Appendix B
Proof of Theorem 2

We develop the initial value function as shown in (35) at the top of the next page. By starting with the definition from (8), the sum of expectations keeps the same value when considering the transition function of controllable elements from (2). The $(*)$ property has the same meaning as in Equation (34). At TTI $t+1$ when $\mathbf{c}'_{(d)} = \mathbf{c}'$, we obtain the value function representation as shown in (10). The action-value function is decomposed as shown below:

$$\begin{aligned} &Q^{\boldsymbol{\pi}}(\mathbf{s}, d) \\ &\overset{(9)}{=} \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum\nolimits_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} | \mathbf{s}[0] = \mathbf{s}, a[0] = d\Big] \\ &\overset{(2)}{=} \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} | \mathbf{c}[1] = f(\mathbf{s}, d), \mathbf{s}[0] = \mathbf{s}, a[0] = d\right] \\ &= \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} | \mathbf{c}[1] = \mathbf{c}'_{(d)}, \mathbf{c}[0] = \mathbf{c}, \mathbf{z}[0] = \mathbf{z}, a[0] = d\right] \\ &\overset{(*)}{=} \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum\nolimits_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} | \mathbf{c}[1] = \mathbf{c}'_{(d)}, \mathbf{c}[0] = \mathbf{c}, a[0] = d\Big] \\ &= J^{\boldsymbol{\pi}}\Big(\mathbf{c}'_{(d)}, \mathbf{c}, d\Big). \end{aligned} \tag{36}$$

The action-value is developed in (36). At TTI t+1 when $\mathbf{c}'_{(d)} = \mathbf{c}'$, the new action-value function is defined as follows:

$$\begin{aligned} J^{\boldsymbol{\pi}}\big(\mathbf{c}', \mathbf{c}, d'\big) &= \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum\nolimits_{t=1}^{\infty} \gamma^{t-1} \mathcal{R}_t | \mathbf{c}[1] = \mathbf{c}', \\ &\qquad \mathbf{c}[0] = \mathbf{c}, a[1] = d', a[0] = d\Big] \\ &\overset{(**)}{=} \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum\nolimits_{t=1}^{\infty} \gamma^{t-1} \mathcal{R}_t | \mathbf{c}[1] = \mathbf{c}', \\ &\qquad \mathbf{c}[0] = \mathbf{c}, a[1] = d'\Big], \end{aligned} \tag{37}$$

where, $(**)$ stands with the MDP property.

Appendix C
Function Transitions

We want to find a relationship for value and action-value functions in between two consecutive controller states such as $(\mathbf{c}', \mathbf{c})$ and $(\mathbf{c}'', \mathbf{c}')$. The state function in (35) is reloaded and developed in reverse way as shown in (38) at the top of the next page. The property $(*, 2)$ indicates the MDP property in which the current state depends only on the previous one but not on all the previous versions. Also, the transition of controllable elements at TTI $t = 2$ is determined according to (2). The property $(**)$ indicates in fact that we are in state $(\mathbf{c}'', \mathbf{c}')$ and we would like to update $J^{\pi}(\mathbf{c}', \mathbf{c}, d)$ and $K^{\pi}(\mathbf{c}', \mathbf{c})$, so the uncontrollable state $\mathbf{z}[2] \in \mathcal{S}^U$ is known. The relation (34*) reveals the property elaborated in Appendix A and the uncontrollable elements $\mathbf{z}[2] = \mathbf{z}'' \in \mathcal{S}^U$ can be reconstituted if the set of elements $\mathbf{c}[2] = \mathbf{c}'' \in \mathcal{S}^C$ and $\mathbf{c}[1] = \mathbf{c}' \in \mathcal{S}^C$ are known by the RL controller. The transition for action-value function can be developed similarly to the value function $K^{\pi}(\mathbf{c}', \mathbf{c})$ with the only difference being that, when updating function $J^{\pi}(\mathbf{c}', \mathbf{c}, d)$ at TTI $t = 2$, a different function $J^{\pi}(\mathbf{c}'', \mathbf{c}', d'')$ may be used in this computation since $d'' = argmax_{q \in \mathcal{D}}[J^{\pi}(\mathbf{c}'', \mathbf{c}', q)]$. This is revealed in fact by the dynamic programming property that permits the action-value function to be updated at each TTI according to the best scheduling rule in that state.

Acknowledgment

The authors would like to acknowledge the contributions of their colleagues in the project although the views expressed

$$
\begin{aligned}
V^{\boldsymbol{\pi}}(\mathbf{s}) &\overset{(8)}{=} \sum_{d\in\mathcal{D}}\left\{\mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t\mathcal{R}_{t+1}|\mathbf{s}[0]=\mathbf{s},a[0]=d\right]\cdot\pi(d\mid\mathbf{s})\right\}\overset{(2)}{=}\sum_{d\in\mathcal{D}}\left\{\mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t\mathcal{R}_{t+1}|\mathbf{c}[1]=f(\mathbf{s},d),\mathbf{s}[0]=\mathbf{s},a[0]=d\right]\pi(d\mid\mathbf{s})\right\}\\
&= \sum_{d\in\mathcal{D}}\left\{\mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t\mathcal{R}_{t+1}|\mathbf{c}[1]=f(\mathbf{s},d),\mathbf{c}[0]=\mathbf{c},\mathbf{z}[0]=\mathbf{z},a[0]=d\right]\cdot\pi(d\mid\mathbf{s})\right\}\\
&\overset{(*)}{=} \sum_{d\in\mathcal{D}}\left\{\mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t\mathcal{R}_{t+1}|\mathbf{c}[1]=\mathbf{c}'_{(d)},\mathbf{c}[0]=\mathbf{c},a[0]=d\right]\cdot\pi\Big[d\mid\Big(\mathbf{c}'_{(d)},\mathbf{c}\Big)\Big]\right\}\\
&= \mathbb{E}_{\boldsymbol{\pi}}\Big[\textstyle\sum_{t=0}^{\infty}\gamma^t\mathcal{R}_{t+1}|\mathbf{c}[1]=\mathbf{c}'_{(d)},\mathbf{c}[0]=\mathbf{c}\Big]=K^{\boldsymbol{\pi}}\Big(\mathbf{c}'_{(d)},\mathbf{c}\Big). \qquad (35)\\
K^{\boldsymbol{\pi}}(\mathbf{c}',\mathbf{c}) &= \textstyle\sum_{d'\in\mathcal{D}}J^{\boldsymbol{\pi}}(\mathbf{c}',\mathbf{c},d')\cdot\pi\big[d'\mid(\mathbf{c}',\mathbf{c})\big]=\sum_{d'\in\mathcal{D}}\mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=1}^{\infty}\gamma^{t-1}\mathcal{R}_t|\mathbf{c}[1]=\mathbf{c}',\mathbf{c}[0]=\mathbf{c},a[1]=d'\Big]\cdot\pi\big[d'\mid(\mathbf{c}',\mathbf{c})\big]\\
&= r(\mathbf{c}',\mathbf{c},d')+\gamma\cdot\sum_{\mathbf{b}'}\sum_{d''}\mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=2}^{\infty}\gamma^{t-2}\mathcal{R}_{t-1}|\mathbf{c}[1]=\mathbf{c}',\mathbf{c}[0]=\mathbf{c},\mathbf{z}[2]=\mathbf{b}',\mathbf{z}[1]=\mathbf{z}',a[2]=d'',a[1]=d'\right]\cdot\pi\big[d''\mid(\mathbf{c}'',\mathbf{c}')\big]\\
&\overset{(*,2)}{=} r(\mathbf{c}',\mathbf{c},d')+\gamma\cdot\sum_{\mathbf{b}'}\sum_{d''}\mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=2}^{\infty}\gamma^{t-2}\mathcal{R}_{t-1}|\mathbf{c}[2]=f\big[(\mathbf{c}',\mathbf{b}'),d'\big],\mathbf{c}[1]=\mathbf{c}',\mathbf{z}[2]=\mathbf{b}',a[2]=d''\right]\cdot\pi\Big[d''\mid\Big(\mathbf{c}''_{(d'')},\mathbf{c}'\Big)\Big]\\
&\overset{(**)}{=} r(\mathbf{c}',\mathbf{c},d')+\gamma\cdot\sum_{d''}\mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=2}^{\infty}\gamma^{t-2}\mathcal{R}_{t-1}|\mathbf{c}[2]=\mathbf{c}''_{(d'')},\mathbf{c}[1]=\mathbf{c}',\mathbf{z}[2]=\mathbf{z}'',a[2]=d''\Big]\cdot\pi\Big[d''\mid\Big(\mathbf{c}''_{(d'')},\mathbf{c}'\Big)\Big]\\
&\overset{(34*)}{=} r(\mathbf{c}',\mathbf{c},d')+\gamma\cdot\sum_{d''}\mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=2}^{\infty}\gamma^{t-2}\mathcal{R}_{t-1}|\mathbf{c}[2]=\mathbf{c}''_{(d'')},\mathbf{c}[1]=\mathbf{c}',a[2]=d''\Big]\cdot\pi\Big[d'\mid\Big(\mathbf{c}''_{(d'')},\mathbf{c}'\Big)\Big]\\
&= r(\mathbf{c}',\mathbf{c},d')+\gamma\cdot\sum_{d''\in\mathcal{D}}J^{\boldsymbol{\pi}}\Big(\mathbf{c}''_{(d')},\mathbf{c}',d''\Big)\cdot\pi\Big[d''\mid\Big(\mathbf{c}''_{(d'')},\mathbf{c}'\Big)\Big]\overset{(37)}{=}r(\mathbf{c}',\mathbf{c},d')+\gamma\cdot K^{\pi}(\mathbf{c}'',\mathbf{c}') \qquad (38)
\end{aligned}
$$

in this paper are those of the authors and do not represent the project.

REFERENCES

[1] *5G White Paper*, Next Gener. Mobile Netw., Frankfurt, Germany, Feb. 2015.

[2] Ericsson. (2016). *5G Radio Access White Paper*. [Online]. Available: http://www.ericsson.com/res/docs/whitepapers/wp-5g.pdf

[3] J. G. Andrews *et al.*, "What will 5G be?" *IEEE J. Sel. Areas Commun.*, vol. 3, no. 6, pp. 1065–1082, Jun. 2014.

[4] T. O. Olwal, K. Djouani, and A. M. Kurien, "A survey of resource management toward 5G radio access networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1656–1686, 3rd Quart., 2016.

[5] P. K. Agyapong, M. Iwamura, D. Staehle, W. Kiess, and A. Benjebbour, "Design considerations for a 5G network architecture," *IEEE Commun. Mag.*, vol. 52, no. 11, pp. 65–75, Nov. 2014.

[6] A. Ghosh, J. Zhang, J. G. Andrews, and R. Muhamed, *Fundamentals of LTE*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, Sep. 2010.

[7] J. Rhee, J. Holtzman, and D.-K. Kim, "Scheduling of real/non-real time services: Adaptive EXP/PF algorithm," in *Proc. IEEE Veh. Technol. Conf.*, vol. 1, Apr. 2003, pp. 462–466.

[8] B. Sadiq, R. Madan, and A. Sampath, "Downlink scheduling for multiclass traffic in LTE," *EURASIP J. Wireless Commun. Netw.*, vol. 2009, no. 14, pp. 1–18, 2009.

[9] B. Liu, H. Tian, and L. Xu, "An efficient downlink packet scheduling algorithm for real time traffics in LTE systems," in *Proc. IEEE Consum. Commun. Netw. Conf. (CCNC)*, vol. 1, Jan. 2013, pp. 364–369.

[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: IMT Press, 2012.

[11] M. Bkassiny, Y. Li, and S. K. Jayaweera, "A survey on machine-learning techniques in cognitive radios," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 1136–1159, 3rd Quart., 2013.

[12] O. C. Iacoboaiea, B. Sayrac, S. B. Jemaa, and P. Bianchi, "SON coordination in heterogeneous networks: A reinforcement learning framework," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 5835–5847, Sep. 2016.

[13] N. Pratas and N. H. Mahmood, "D4.1: Technical results for service specific multi-node/multiantenna solutions," in *The Framework European Project: Flexible Air Interface for Scalable Service Delivery Within Wireless Communication Networks of the 5th Generation (FANTASTIC-5G)*, 2016, pp. 19–39. [Online]. Available: http://fantastic5g.com/wp-content/uploads/2016/06/FANTASTIC-5GßD4.1ßFinal.pdf

[14] G. Song and Y. Li, "Utility-based resource allocation and scheduling in OFDM-based wireless broadband networks," *IEEE Commun. Mag.*, vol. 43, no. 12, pp. 127–134, Dec. 2005.

[15] G. Piro, L. A. Grieco, G. Boggia, F. Capozzi, and P. Camarda, "Simulating LTE cellular systems: An open-source framework," *IEEE Trans. Veh. Technol.*, vol. 60, no. 2, pp. 498–513, Feb. 2011.

[16] *Technical Specification Group Services and System Aspects; Policy and Charging Control Architecture Release 12, V.12.2.0*, 3GPP, Sophia Antipolis, France, 2013.

[17] C. Szepesvari, *Algorithms for Reinforcement Learning: Synthesis Lectures on Artificial Intelligence and Machine Learning*. San Rafael, CA, USA: Morgan, 2010.

[18] I.-S. Comşa, *Sustainable Scheduling Policies for Radio Access Networks Based on LTE Technology*. Luton, U.K.: Univ. Bedfordshire, 2014.

[19] H. P. van Hasselt, *Insights in Reinforcement Learning Formal Analysis and Empirical Evaluation of Temporal-Difference Learning Algorithms*, Utrecht, The Netherlands: Univ. Utrecht, 2011.

[20] M. Wiering, "QV(lambda)-learning: A new on-policy reinforcement learning algorithm," in *Proc. 7th Eur. Workshop Reinforcement Learn.*, 2005, pp. 17–18.

[21] M. A. Wiering and H. van Hasselt, "The QV family compared to other reinforcement learning algorithms," in *Proc. IEEE Int. Symp. Approx. Dyn. Program. Reinforcement Learn. (ADPRL)*, 2009, pp. 101–108.

[22] H. Van Hasselt and M. A. Wiering, "Using continuous action spaces to solve discrete problems," in *Proc. Int. Joint Conf. Neural Netw.*, 2009, pp. 1149–1156.

[23] R. Basukala, H. M. Ramli, and K. Sandrasegaran, "Performance analysis of EXP/PF and M-LWDF in downlink 3GPP LTE system," in *Proc. 1st Asian Himalayas Int. Conf. Internet*, vol. 1, Nov. 2009, pp. 1–5.

[24] M. Simsek, M. Bennis, and I. Guvenc, "Learning based frequency- and time-domain inter-cell interference coordination in HetNets," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4589–4602, Oct. 2015.

[25] O. Iacoboaiea, B. Sayrac, S. Ben Jemaa, and P. Bianchi, "Coordinating SON instances: Reinforcement learning with distributed value function," in *Proc. IEEE 25th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, vol. 1, Sep. 2014, pp. 1642–1646.

[26] 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 *Proc. IEEE Int. Conf. Commun. (ICC)*, 2016, pp. 1–6.

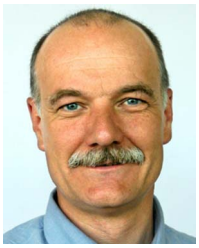
[27] R. Bruno, A. Masaracchia, and A. Passarella, "Robust adaptive modulation and coding (AMC) selection in LTE systems using reinforcement learning," in *Proc. IEEE Veh. Technol. Conf. (VTC-Fall)*, Sep. 2014, pp. 1–6.

[28] A. Chiumento, C. Desset, S. Pollin, L. Van der Perre, and R. Lauwereins, "Impact of CSI feedback strategies on LTE downlink and reinforcement learning solutions for optimal allocation," *IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 550–562, Jan. 2017.
[29] I.-S. Comşa *et al.*, "Scheduling policies based on dynamic throughput and fairness tradeoff control in LTE-A networks," in *Proc. IEEE Local Comput. Netw. Conf. (LCN)*, Sep. 2014, pp. 418–421.
[30] I.-S. Comşa *et al.*, "Adaptive proportional fair parameterization based LTE scheduling using continuous actor-critic reinforcement learning," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Dec. 2014, pp. 4387–4393.
[31] D. Zhao and Y. Zhu, "MEC—A near-optimal online reinforcement learning algorithm for continuous deterministic systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 2, pp. 346–356, Feb. 2015.
[32] T. Mori and S. Ishii, "Incremental state aggregation for value function estimation in reinforcement learning," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 5, pp. 1407–1416, Oct. 2011.

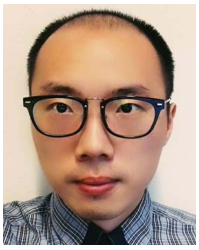
Ioan-Sorin Comşa received the B.Sc. and M.Sc. degrees in telecommunications from the Technical University of Cluj-Napoca, Romania, in 2008 and 2010, respectively, and the Ph.D. degree from the Institute for Research in Applicable Computing, University of Bedfordshire, U.K., in 2015. He is a Research Scientist in 5G radio resource scheduling with Brunel University, London, U.K. He was also a Ph.D. Researcher with the Institute of Complex Systems, University of Applied Sciences of Western Switzerland, Switzerland. Since 2015, he has been a Research Engineer with CEA-LETI, Grenoble, France. His research interests include intelligent radio resource and QoS management, reinforcement learning, data mining, distributed and parallel computing, and adaptive multimedia/mulsemedia delivery.

Sijing Zhang received the B.Sc. and M.Sc. degrees in computer science from Jilin University, Changchun, China, in 1982 and 1988, respectively, and the Ph.D. degree in computer science from the University of York, U.K., in 1996. He then joined the Network Technology Research Centre, Nanyang Technological University, Singapore as a Post-Doctoral Fellow. In 1998, he was a Research Fellow with the Centre for Communications Systems Research, University of Cambridge, U.K. He joined the School of Computing and Technology, University of Derby, U.K., as a Senior Lecturer in 2000. Since 2004, he has been a Senior Lecturer with the Department of Computer Science and Technology, University of Bedfordshire, U.K. His research interests include wireless networking, schedulability tests for hard real-time traffic, performance analysis and evaluation of real-time communication protocols, QoS provision, vehicular ad-hoc networks, and wireless networks for real-time industrial applications.

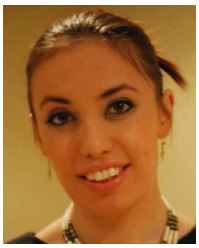
Mehmet Emin Aydin (M'06–SM'16) received the B.Sc. degree from Istanbul Technical University, the M.A. degree from Istanbul University, and the Ph.D. degree from Sakarya University. He is a Senior Lecturer with the Department of Computer Science and Creative Technology, University of the West of England. His research interests include parallel and distributed metaheuristics, resource planning, scheduling and optimization, combinatorial optimization, evolutionary computation, and intelligent agents and multiagent systems. He has conducted guest editorial for a number of special issues of peer-reviewed international journals. He is a member of advisory committees of many international conferences and an editorial board member of various peer-reviewed international journals. He is currently a fellow of Higher Education Academy, U.K., and a member of EPSRC College, OR Society U.K., and ACM.



Pierre Kuonen received the master's degree in electrical engineering from the École Polytechnique Fédérale de Lausanne (EPFL) in 1982 and the Ph.D. degree in computer science in 1993. After six years of experience in the industry (oil exploration in Africa, then development of computer-assisted manufacturing tools in Geneva), he joined the Laboratory of Computer Science Theory with EPFL in 1988, where he began working in the field of parallel and distributed systems for high-performance computing. He founded and led the GRIP (Parallel Computing Research Group) with EPFL. He joined the University of Applied Sciences of Western Switzerland (HES-SO) in 2000. Since 2003, he has been a Full Professor with the University of Applied Sciences and Architecture, Fribourg, where he leads the GRID and Cloud Computing Group and co-leads the Institute of Complex Systems. In 2014, in collaboration with Prof. J. Hennebert (HES-SO, Fribourg) and Prof. P. Cudre-Mauroux (University of Fribourg) he founded the Data Analysis and Processing Lab that aims at facilitating the access for enterprises and universities to emerging technologies in the area of big data and intelligent data analysis.



Yao Lu received the master's degree in computer system and architecture from Chongqing University, China, in 2012. He is currently pursuing the Ph.D. degree with the Department of Informatics, University of Fribourg, Switzerland. He was also with the GRID and Cloud Computing Group, University of Applied Sciences of Western Switzerland. He also actively participated in international conferences and published papers in academic journals. His research interests focuses on wireless sensor networks, routing protocol, data aggregation, and swarm intelligence.



Ramona Trestian (S'08–M'12) received the Ph.D. degree from Dublin City University, Ireland, in 2012. She is a Senior Lecturer with the Design Engineering and Mathematics Department, Middlesex University, London, U.K. She was with Dublin City University as an IBM/IRCSET Exascale Post-Doctoral Researcher. She published in prestigious international conferences and journals and has three edited books. Her research interests include mobile and wireless communications, user perceived quality of experience, multimedia streaming, and handover and network selection strategies. She is a member of IEEE Young Professionals, IEEE Communications Society, and IEEE Broadcast Technology Society.

Gheorghiţă Ghinea (M'01) received the first B.Sc. degree in computer science, the second B.Sc. degree (Hons.) in mathematics, and the M.Sc. degree in computer science from the University of the Witwatersrand, Johannesburg, South Africa, in 1993, 1994, and 1996, respectively, and the Ph.D. degree in computer science from the University of Reading, U.K., in 2000. He is a Professor with the Computer Science Department, Brunel University, U.K. His work focuses on building adaptable cross-layer end-to-end communication systems incorporating user perceptual requirements. He is a member of the British Computer Society.