5MART: A 5G sMART Scheduling Framework for Optimizing QoS Through Reinforcement Learning

Ioan-Sorin Comşa, Ramona Trestian, Gabriel-Miro Muntean, and Gheorghiţă Ghinea

***Abstract*—The massive growth in mobile data traffic and the heterogeneity and stringency of Quality of Service (QoS) requirements of various applications have put significant pressure on the underlying network infrastructure and represent an important challenge even for the very anticipated 5G networks. In this context, the solution is to employ smart Radio Resource Management (RRM) in general and innovative packet scheduling in particular in order to offer high flexibility and cope with both current and upcoming QoS challenges. Given the increasing demand for bandwidth-hungry applications, conventional scheduling strategies face significant problems in meeting the heterogeneous QoS requirements of various application classes under dynamic network conditions. This paper proposes *5MART*, a 5G smart scheduling framework that manages the QoS provisioning for heterogeneous traffic. Reinforcement learning and neural networks are jointly used to find the most suitable scheduling decisions based on current networking conditions. Simulation results show that the proposed *5MART* framework can achieve up to 50% improvement in terms of time fraction (in sub-frames) when the heterogeneous QoS constraints are met with respect to other state-of-the-art scheduling solutions.**

***Index Terms*—5G, Radio Resource Management, Machine Learning, Scheduling, Traffic Prioritization, QoS Optimization.**

I. Introduction

The fast proliferation of affordable and high-end mobile devices and the increasing popularity of bandwidth-hungry applications (e.g., 360° video streaming, Virtual Reality (VR), autonomous cars, etc.) have led to an exponential increase in mobile broadband traffic putting significant pressure on the underlying wireless networks [1]. Recent predictions show that smartphones will generate more than 86% of mobile data traffic by 2021 with 15% of the world's mobile data representing VR traffic [2]. Looking at the current environment, it is apparent that these high expectations cannot be achieved with just a simple increase in system capacity without rethinking the entire network architecture. Thus, the next generation of wireless networks will integrate a number of hybrid solutions and key technologies, such as beamforming, massive antenna arrays or even a flexible RRM [3].

However, the integration of these hybrid emerging technologies will increase the complexity of the system even more. Thus, there is a need for intelligent decision-making solutions to enable a self-optimizing and self-organizing environment. One promising solution that could boost network performance is the use of Machine Learning (ML). In the context of 5G, an important entity that could benefit from the integration of ML is the RRM. Within RRM, the scheduler aims at dynamically sharing the available resources between the mobile users in each Transmission Time Interval (TTI). Many scheduling schemes have been adopted to deal with different QoS provisioning strategies for a wide range of applications. In the frequency domain, users are competing to get radio resources according to some scheduling rules that focus mainly on certain QoS objectives. For example, the Proportional-Fair (PF) [4] scheduler provides a trade-off between user fairness and throughput maximization. For real-time applications, the EXPonential (EXP) rule deals with delay minimization [5]. Other frequency-based schedulers such as the Opportunistic Packet Loss Fair (OPLF) [6] and the Barrier Function (BF) [7] are designed to deal with Packet Loss Rate (PLR) minimization and with meeting the Guaranteed Bit Rate (GBR) requirements, respectively.

Frequency domain schedulers are used in conjunction with time domain schemes to attain a certain prioritization between traffic classes with heterogeneous QoS requirements. Most of these hybrid schedulers will always favor users with more stringent QoS requirements to be scheduled in the frequency domain [8]–[11]. Moreover, the QoS provisioning scheme is divided among time and frequency domains. For example, the scheduler in [8] preselects users with the highest head-of-line packet delay and the frequency domain focuses more on meeting the GBR requirement for each preselected user. The Required Activity Detection Scheduler (RADS) [10] deals with delay minimization in the time domain, whereas the frequency domain performs the PF scheduling rule to achieve proper throughput-fairness trade-offs. To minimize PLR and packet delay, the Frame Level Scheduler (FLS) [11] estimates in the time domain the amount of real-time data to be transmitted in the next frame, while the same PF scheduling rule is performed in the frequency domain. In the frequency domain, other schedulers use heuristic algorithms to minimize the throughput loss that can be caused by the time domain prioritization [12], [13]. In multi-user, multi-service and multi-network environments, these decoupled time-frequency schedulers could be employed together with network reputation algorithms [14] to select the most convenient network that enables QoS provisioning.

Due to application diversity and the heterogeneity of QoS requirements, most of these state-of-the-art schedulers are rather static, being unable to meet the QoS requirements under dynamic networking conditions [15]. By performing time domain prioritization under the same static metric, some traffic classes will be over-provisioned while others will be starved, resulting in poor QoS provisioning. Moreover, only certain QoS objectives are targeted in each scheduling domain while

I.-S. Comşa and G. Ghinea are with Brunel University London, U.K. (e-mails: ioan-sorin.comsa@brunel.ac.uk and george.ghinea@brunel.ac.uk).

R. Trestian is with Middlesex University, London, U.K. (e-mail: r.trestian@mdx.ac.uk).

G.-M. Muntean is with Dublin City University, Dublin, Ireland (e-mail: gabriel.muntean@dcu.ie).

the remaining ones are ignored. To improve the scheduling performance and significantly increase the fraction of time (in TTIs) when the heterogeneous QoS constraints are met, Reinforcement Learning (RL) is seen as a promising solution [16] that will learn the most convenient scheduling strategy to be applied in each scheduler state.

In resource scheduling problems [17], different RL algorithms are studied to get the best policy for on-line parameterization of the PF scheduling rule to meet a given fairness criterion. The RL solution proposed in [18] selects at each TTI in the frequency domain the most suitable scheduling rule to maximize the multi-objective QoS target in terms of GBR, delay and PLR for homogeneous traffic only. Similarly, a RL framework is proposed in [19] for homogeneous traffic with constant and variable bit rates, to deal with delay and PLR objectives. The work in [19] provides a comparison between different RL algorithms and aims to find the most suitable scheduling policy for various time window lengths used to compute the online PLR indicators. Compared to such approaches where only homogeneous traffic is considered, this work proposes an intelligent 5G downlink scheduling framework aiming at improving QoS provisioning in terms of GBR, delay and PLR requirements for heterogeneous traffic. However, to deal with heterogeneous traffic scheduling, a two-stage learning approach is proposed in [20]. The first stage employs a separate learning phase for each traffic class to approximate the best scheduling rule to be used. The second stage decides the best prioritization order of the traffic classes at each TTI. Only when both learning stages are completed, the entire structure can be exploited. Compared to [20], in this work, the proposed framework will learn both decisions at once, involving a much lower system complexity.

A. Contributions

We propose an intelligent 5G scheduler for OFDMA downlink systems that makes use of dynamic traffic class selection at each TTI in order to avoid QoS over-provisioning of some classes while affecting others. The conventional OFDMA access scheme is considered less complex and very efficient when compared to the non-orthogonal schemes, making it suitable for 5G networks [21]. In this context, the main contributions of this work are as follows:

1) 5MART Scheduler Framework: an intelligent 5G downlink scheduler able to change dynamically (at each TTI) based on current scheduler states: *a)* the traffic class to be prioritized at first in the time domain and *b)* the most convenient scheduling rule to be performed in the frequency domain; the aim is to increase the fraction of time (in TTIs) when the heterogeneous QoS constraints are met.

2) Neural Network Approximation: Due to its simplicity and reduced complexity, a Neural Network (NN) is employed as a non-linear function approximator to provide scheduling decisions at each TTI. The neural network takes the instantaneous scheduler state as input and provides as output the multi-dimensional decision vector which is decoded in traffic class prioritization and scheduling rule selection.

3) 5MART RL-based Intelligent Controller: an intelligent controller that uses a light-weight Actor-Critic RL algorithm to update at each TTI the neural network based on the interaction with the *5MART* scheduler; we extend [22] by focusing on low complexity in order to target online deployment.

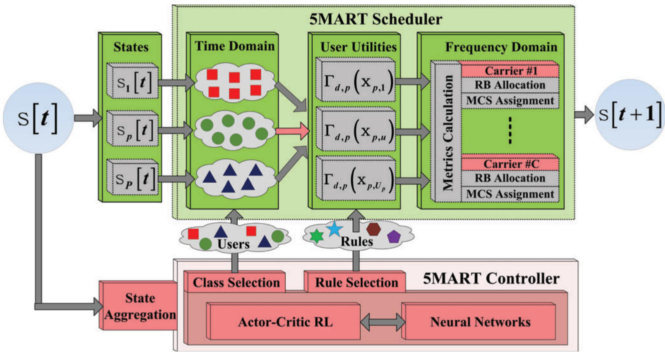


Fig. 1. Proposed 5MART framework.

4) State Space Compression: Additionally, an aggregation technique is introduced for the scheduler state space to eliminate the dependence on the number of users for each traffic class. This reduces the complexity of the proposed *5MART* framework and enhances the learning performance.

B. Paper Organization

This paper is organized as follows: Section II presents the system model. Section III introduces the *5MART* RL framework and the Actor-Critic RL scheme is detailed in Section IV. Simulation results are presented and discussed in Section V and the paper is concluded in Section VI.

II. System Model

The OFDMA downlink transmission is considered, where the available bandwidth is divided in Resource Blocks (RBs), the smallest resource unit that can be allocated to a user within a single TTI. In order to accommodate bandwidth-hungry traffic classes with very high data rates, we make use of carrier aggregation where a number of C carrier components are considered. Since a user can be scheduled on multiple carrier components, both time and frequency prioritization domains support joint scheduling on multiple carrier components. A User Equipment (UE) is characterized by a wide range of traffic classes, such as: 360° video, virtual reality, augmented reality, gaming, 2D video, VoIP, File Transfer Protocol (FTP), etc. Each of these traffic classes are constrained by different QoS requirements in terms of GBR, PLR and packet delay. Any UE can switch its state from *idle* to *active* and back to create a dynamic environment. The set of most used parameters is listed in Table I.

A. 5MART Scheduling Model

Fig. 1 presents the proposed *5MART* system model as an interaction framework between a specialized controller and the decoupled time-frequency packet scheduler. In TTI t, a state is perceived including observations and network measurements, such as: QoS indicators, channel quality measurements, etc. Each perceived state is used by: *a)* the intelligent controller to decide the traffic class to be prioritized in the time domain and the scheduling rule to be employed in the frequency domain; *b)* the packet scheduler to perform the resource allocation given the priority order and the scheduling rule a priori decided. The scheduler controller compresses the initial scheduler state

Table I List of Notations

Parameter	Description
$\mathcal{P}$	Set of traffic classes in the priority order given by [23]
p, P	Arbitrary traffic $p \in \mathcal{P}$, max. no. of traffic classes
$\mathcal{D}$	Set of scheduling rules
d	Arbitrary scheduling rule $d \in \mathcal{D}$
D	Max. no. of scheduling rules
$\mathcal{B}$	Set of resource blocks from different carriers
b	Arbitrary resource block $b \in \mathcal{B}$
B	Max. no. of resource blocks
$\mathcal{U}_p$	Set of active users belonging to class $p \in \mathcal{P}$
u	Arbitrary user $u \in \mathcal{U}_p$
U_p	Max. no. of users belonging to class $p \in \mathcal{P}$
$\mathcal{O}$, o	Set of objectives, arbitrary objective $o \in \mathcal{O}$
$x_{o,p,u}$	QoS indicator of $o \in \mathcal{O}$, user u and class $p \in \mathcal{P}$
$\bar{x}_{o,p,u}$	QoS requirement of $o \in \mathcal{O}$, user u and class $p \in \mathcal{P}$
$\Gamma_{d,u}$	Utility function of rule $d \in \mathcal{D}$ and user $u \in \mathcal{U}_p$
$\mathcal{S}$	Continuous and multi-dimensional scheduler state space
$\mathbf{s}$	Momentary scheduler state $\mathbf{s} \in \mathcal{S}$ at TTI t
$\mathbf{c}$	Controllable sub-state at TTI t as a subset of $\mathbf{s} \in \mathcal{S}$
$\mathcal{A}$	Discrete and two-dimensional controller action space
$\mathbf{a}$	Momentary action $\mathbf{a} \in \mathcal{A}$ decided at TTI t
$r(\mathbf{c}', \mathbf{c})$	System reward value received at TTI t+1
$\mathbf{z}$	Controller state at TTI t (compressed scheduler state)
$\bar{H}^*(\mathbf{z})$	Approximation of optimal state-value function
$A(\mathbf{z})$	Approximation of optimal action-value function
L, L_H	Total number of NN layers, number of hidden layers
$\mathbf{W}_{l,l+1}^{+}$	Biased matrix of weights between layers l and l+1
e^c, $\mathbf{e}^a$	Critic and actor errors, respectively
$\pi(\mathbf{A} \mid \mathbf{z})$	Probability of selecting decision vector $\mathbf{A}$ in state $\mathbf{z}$

in order to obtain a fixed dimension and make the learning procedure possible. The internal structure makes use of a feed forward neural network which is learnt over time by an actor-critic RL algorithm to approximate based on the received compressed state, at each TTI, the best traffic class to be prioritized and the best scheduling rule to be applied in the frequency domain. For time domain scheduling, we consider the set $\mathcal{P}$ of traffic classes initially prioritized, as follows: $\mathcal{P}$={1(red square),...,p(green circle),...,P(blue triangle)}, where class 1 requires the highest priority while class P is associated with the lowest priority among P number of traffic classes. However, at TTI t, the proposed *5MART* controller may decide to violate the initial ordering scheme by deciding to prioritize class $p \in \mathcal{P}$ (green circle) over other traffic classes. In frequency domain scheduling, users requesting traffic $p \in \mathcal{P}$ are competing to each other in order to get the maximum number of radio resources. We consider the frequency band as a set of equally distributed RBs over a number of C carrier components, as follows: $\mathcal{B} = \{1, 2, ..., B\}$, where B is the number of aggregate RBs. To exchange the scheduling information (e.g., controller decision, QoS indicators, etc.) between multiple component carriers, the frequency prioritization or the RB allocation is jointly performed on all carriers. The selected scheduling rule aims to quantify the selection of each RB $b \in \mathcal{B}$ for each user belonging to $p \in \mathcal{P}$ in terms of QoS provisioning. Once these metrics are computed for each $b \in \mathcal{B}$ and user, the frequency prioritization allocates each RB to the user with the highest utility metric. Once the RB allocation is performed, the Transport Block (TB) size and the Modulation and Coding Scheme (MCS) computation for each scheduled user are separately performed on each carrier component.

For each traffic class $p \in \mathcal{P}$, we associate the set of users at TTI t defined as: $\mathcal{U}_p = \{u_1, u_2, ..., u_{U_p}\}$, where U_p is the total number of users belonging to class $p \in \mathcal{P}$. For each user $u \in \mathcal{U}_p$, we define a set of three QoS objectives $\mathcal{O} = \{o_1, o_2, o_3\}$, where $o_1 = GBR$, $o_2 = PLR$ and $o_3 = DELAY$. Furthermore, we consider $x_{o,p,u}$ the QoS indicator corresponding to objective $o \in \mathcal{O}$ and $\bar{x}_{o,p,u}$ its associated QoS requirement. Objective $o \in \mathcal{O}$ of user $u \in \mathcal{U}_p$ is fulfilled if and only if $x_{o,p,u}$ meets its requirement $\bar{x}_{o,p,u}$. To this extent, it is said that all three QoS objectives $\mathcal{O}$ are met for user $u \in \mathcal{U}_p$, if the QoS vector $\mathbf{x}_{p,u} = [x_{o_1,p,u}, x_{o_2,p,u}, x_{o_3,p,u}]$ respects its requirement vector $\bar{x}_{p,u} = [\bar{x}_{o_1,p,u}, \bar{x}_{o_2,p,u}, \bar{x}_{o_3,p,u}]$. On a larger scale, the set of objectives $\mathcal{O}$ is met for the entire set of users $\mathcal{U}_p$ if the vector of QoS indicators for class $p \in \mathcal{P}$ $\mathbf{x}_p = [\mathbf{x}_{p,u_1}, \mathbf{x}_{p,u_2}, ..., \mathbf{x}_{p,u_{U_p}}]$ respects the intra-class requirement vector $\bar{\mathbf{x}}_p = [\bar{\mathbf{x}}_{p,u_1}, \bar{\mathbf{x}}_{p,u_2}, ..., \bar{\mathbf{x}}_{p,u_{U_p}}]$. On the largest scale, our proposal aims to take the most suitable prioritization decisions in order to increase the fraction of time (in TTIs) when the QoS vector $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_P]$ respects the QoS requirements from vector $\bar{\mathbf{x}} = [\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, ..., \bar{\mathbf{x}}_P]$.

In the time domain, a Static Prioritization (SP) scheme reveals that, regardless of the network conditions, the scheduling process is conducted following the order $\mathcal{P} = \{1, 2, ..., p-1, p, p+1, ..., P\}$, where the first prioritized class takes always most of the radio resources while degrading the QoS provisioning of other classes with lower priority. To avoid this drawback, the *5MART* framework will decide at each TTI t a new prioritization set $\mathcal{P}[t] = \{p, 1, 2, ..., p-1, p+1, ..., P\}$, in which, an arbitrary class $p \in \mathcal{P}$ will be allocated firstly, being followed by the rest of classes from $\{1, 2, ..., p-1, p+1, ..., P\}$ depending on the amount of remaining resources. A more ambitious framework would be able to modify at each TTI the entire traffic order, such as for instance: $\{p, 5, p+1..., P, p-1, ..., P-1\}$. However, this approach would require a much higher output dimension for the neural network that increases the computational complexity of the system to the extent of making it unsuitable for real-time scheduling.

In the frequency domain, users $u \in \mathcal{U}_p$ are prioritized to get as many resources as possible through the use of some scheduling rules. We define the set of scheduling rules $\mathcal{D} = \{1, 2, .., D\}$, where each scheduling rule $d \in \mathcal{D}$ has a different impact on meeting objectives $\{o_1, o_2, o_3\} \in \mathcal{O}$ based on current network conditions and traffic class requirements. Each scheduling rule has an associated utility function $\Gamma_{d,p}(\mathbf{x}_{p,u})$, that takes as input the QoS vector $\mathbf{x}_{p,u}$ and provides the priority order for each pre-selected user to be allocated in the frequency domain on each RB $b \in \mathcal{B}$ [24]. Alongside time domain prioritization, the *5MART* controller is able to dynamically select at each TTI a different utility function Γ_d (the same for all classes each time) in order to contribute to the framework performance by increasing the fraction of time (in TTIs) when the heterogeneous requirements $\bar{\mathbf{x}}$ are met by $\mathbf{x}$. Once $\mathcal{P}_p[t]$ and $\Gamma_d[t]$ are decided, the resource allocation procedure follows four steps: metrics calculation, RBs allocation, TB size calculation and MCS assignment. If some RBs remain unoccupied once class $p \in \mathcal{P}$ is allocated, then the scheduling is repeated for other classes by following the order $\{1, 2, ..., p-1, p+1, ..., P\}$.

B. Optimization Problem

Compared to classical resource allocation and user selection problem, the proposed optimization framework is more difficult to solve in real-time scheduling since both traffic class and scheduling rule assignments need to be performed at the same time. Our aim is to find the most suitable decision variables (e.g., scheduling rule, traffic class, user selection and RB allocation) that can maximize the optimization problem from (1), subject to some constraints (1.a)-(1.m).

$$\max_{m,n,j} \sum_{d\in\mathcal{D}} \sum_{p\in\mathcal{P}} \sum_{u\in\mathcal{U}_p} \sum_{b\in\mathcal{B}} m_{d,p}[t] \cdot n_{p,u}[t] \cdot j_{u,b}[t] \cdot \Gamma_{d,p}(\mathbf{x}_{p,u}[t]) \cdot \gamma_{u,b}[t], \tag{1}$$

$s.t.$

$$\sum\nolimits_{u} j_{u,b}[t] \leq 1, \quad b = 1, ..., B, \tag{1.a}$$

$$\sum\nolimits_{p} n_{p,u}[t] \leq 1, \quad u = u_1, ..., u_{U_p}, p = 1, ..., P, \tag{1.b}$$

$$\sum\nolimits_{u} n_{p^*,u}[t] = U_{p^*}, \quad p^* \in \mathcal{P}, \tag{1.c}$$

$$\sum\nolimits_{u} n_{p^\otimes,u}[t] = 0, \quad \forall p^\otimes \in \mathcal{P}\backslash\{p^*\}, \tag{1.d}$$

$$\sum\nolimits_{d} m_{d,p}[t] = 1, \quad p = 1, 2, ..., P, \tag{1.e}$$

$$\sum\nolimits_{p} m_{d^*,p}[t] = P, \quad d^* \in \mathcal{D}, \tag{1.f}$$

$$\sum\nolimits_{p} m_{d^\otimes,p}[t] = 0, \quad \forall d^\otimes \in \mathcal{D}\backslash\{d^*\}, \tag{1.g}$$

$$m_{d,p}[t] \in \{0,1\}, \quad \forall d \in \mathcal{D}, \forall p \in \mathcal{P}, \tag{1.h}$$

$$n_{p,u}[t] \in \{0,1\}, \quad \forall p \in \mathcal{P}, \forall u \in \mathcal{U}_p, \tag{1.i}$$

$$j_{u,b}[t] \in \{0,1\}, \quad \forall u \in \mathcal{U}_p, \forall b \in \mathcal{B}, \tag{1.j}$$

$$x_{o_1,p,u}[t+1] \geq \bar{x}_{o_1,p,u}, \quad \forall p \in \mathcal{P}, \forall u \in \mathcal{U}_p, \tag{1.k}$$

$$x_{o_2,p,u}[t+1] \leq \bar{x}_{o_2,p,u}, \quad \forall p \in \mathcal{P}, \forall u \in \mathcal{U}_p, \tag{1.l}$$

$$x_{o_3,p,u}[t+1] \leq \bar{x}_{o_3,p,u}, \quad \forall p \in \mathcal{P}, \forall u \in \mathcal{U}_p. \tag{1.m}$$

In the maximization problem, $\gamma_{u,b}[t]$ is the achievable rate of user $u \in \mathcal{U}_p$ on RB $b \in \mathcal{B}$ at TTI t. Basically, $\gamma_{u,b}[t]$ is the maximum number of bits that can be transmitted on RB $b \in \mathcal{B}$ if user $u \in \mathcal{U}_p$ would have the highest priority. This number of bits is calculated based on the Channel Quality Indicator (CQI) transmitted as a control message from each UE to the base station, evaluating the link quality on each RB $b \in \mathcal{B}$. In (1), $j_{u,b}[t]$ is the resource allocation variable (i.e., $j_{u,b}[t] = 1$ if RB $b \in \mathcal{B}$ is allocated to user $u \in \mathcal{U}_p$ and $j_{u,b}[t] = 0$, otherwise). These variables are subject to constraints (1.a), according to which, a single user can get more than one RB, but a RB itself cannot be allocated to more than one UE. The variable $n_{p,u}[t]$ selects the traffic class to be prioritized (i.e., $n_{p,u}[t] = 1$, if class $p \in \mathcal{P}$ is prioritized, and $n_{p,u}[t] = 0$, otherwise). Each user can receive at each TTI at most one traffic class, as shown in (1.b). Constraints (1.c) indicate that only the active users requesting the prioritized service class $p^* \in \mathcal{P}$ are passed to the frequency domain, while other classes $p^\otimes \in \mathcal{P}\backslash\{p^*\}$ are avoided this time, as shown by constraints (1.d). Indicator $m_{d,p}[t]$ is the scheduling rule assignation variable (i.e., $m_{d,p}[t] = 1$ if scheduling rule $d \in \mathcal{D}$ is assigned for scheduling to class $p \in \mathcal{P}$, and $m_{d,p}[t] = 0$, otherwise). Only one scheduling rule can be assigned at once for each active user of each class $p \in \mathcal{P}$ as shown by (1.e). For complexity reasons, the same scheduling rule will be assigned to all traffic classes. This is revealed by (1.f) and (1.g). Constraints (1.h)-(1.j) make the entire problem combinatorial. The adopted solution that solves the problem from (1), should be able to select at each TTI the best assignation variables $\{n_{p,u}, m_{d,p}\} \in \{0,1\}$, such that constraints (1.k)-(1.m) are met when the scheduling task evolves to the next TTI.

The optimization problem in (1) is difficult to solve due to: *a)* the assignments of scheduling rule, prioritized traffic class, users and RBs having to be jointly performed in order to keep the linearity of the problem; this requires a very high decision space where a solution must be found at each TTI; *b)* such combinatorial problems are generally NP-hard; *c)* constraints (1.k)-(1.m) require a priori knowledge on the performance of QoS objectives at TTI t+1; in real-time systems, the QoS provisioning in a given state is evaluated only when the system evolves to the next state.

To solve such a complex optimization problem, we split the framework in sub-optimal sub-problems where: *(a)* we decide the traffic class to be prioritized and the scheduling rule to be applied; and *(b)* we perform resource allocation according to the pre-selected users and selected scheduling rule. To this end, we use reinforcement learning to find the approximations for the best scheduling decisions to be applied in every state.

C. 5MART RL-based Scheduling Solution

The proposed *5MART* RL framework aims to determine the assignment variables $\{m_{d,p}[t], n_{p,u}[t]\}$ by deciding the most appropriate traffic class prioritization and scheduling rule in each current state. Based on the interaction with the RRM environment, the intelligent controller learns over time to take the best scheduling decisions such that the heterogeneous QoS provisioning will reach the highest outcome in each state as requested by constraints (1.k)-(1.m). This stage is entitled *learning*. Since the state-action pairs cannot be enumerated exhaustively due to multi-dimensional and very large scheduler state space, we adopt the neural network representation to approximate the best scheduling decisions at each TTI. To improve the learning speed and accuracy, we employ an actor-critic RL scheme that makes use of two neural networks: *a)* the actor NN is learnt to provide appropriate scheduling decisions; *b)* the critic NN is used to decide when the actor NN weights are updated based on the applied scheduling actions. In the *exploitation* stage, only the actor NN is implemented at the controller level. In order to get aligned with the terminology and data formats from the machine learning domain, the upcoming section provides the prospects needed when employing a RL-based solution to enhance the decision-making in complex scheduling problems.

III. RL Prospects in Scheduling Decisions

At each TTI t, the RL controller perceives the current *state* and takes an *action*. At TTI t+1, a new state is received based on the previous applied action and scheduling/transmission process. A *reward* value is calculated based on constraints (1.k)-(1.m) to evaluate the effectiveness of applying the previous action in the previous state. The RL framework needs a large number of state-to-state iterations in order to improve over time its actions at each state. Then, at each iteration, the selection of the previous action in the previous state must be adapted based on the received reward value in current state.

When employing a NN-based RL framework to approximate the decisions for our *5MART* meta-scheduler, the adaptation process of the action selection at each iteration involves essentially the refinement of neural network weights. In the learning stage, the *5MART* framework may decide to improve or to evaluate the actor NN decisions according to some probability distributions. If the improvement step is decided, then a random output vector is considered to enlarge the exploration of the scheduler state space. When the evaluation step is decided, we aim to consider the output vector provided by the actor NN. Whatever the decision is, the output vector is decoded in traffic class prioritization and scheduling rule selection. The actor NN weights are updated only when the error to be reinforced through the critic NN is positive. The learning stage continues for a high number of iterations until the critic error gets stabilized and reaches a given threshold.

A. State Space

We define the measurable, multi-dimensional and continuous scheduler state space as $\mathcal{S} = \mathcal{S}^C \cup \mathcal{S}^U$, where $\mathcal{S}^C$ is the controllable sub-space containing the QoS indicators and users' queue lengths; $\mathcal{S}^U$ is the uncontrollable sub-space being defined by CQI reports and data rates of given applications. As the notations suggest, $\mathcal{S}^C$ is related more to the scheduling decisions taken each time, whereas $\mathcal{S}^U$ is related to the stochastic nature of the scheduling process. The momentary scheduler state at TTI t is defined as: $\mathbf{s}[t] = [\mathbf{c}[t], \mathbf{v}[t]] \in \mathcal{S}$, where $\mathbf{c} \in \mathcal{S}^C$ and $\mathbf{v} \in \mathcal{S}^U$. The instantaneous controllable state is a vector of $\mathbf{c}[t] = [\mathbf{x}, \underline{\mathbf{x}}, \mathbf{q}]$, where $\underline{\mathbf{x}} = [\underline{\mathbf{x}}_1, \underline{\mathbf{x}}_2, ..., \underline{\mathbf{x}}_P]$ measures the accomplishment degree of QoS requirements, where $\underline{\mathbf{x}}_p = [\underline{\mathbf{x}}_{p,u_1}, ..., \underline{\mathbf{x}}_{p,u_{U_p}}]$ and $\underline{\mathbf{x}}_{p,u} = [\underline{x}_{o_1,p,u}, \underline{x}_{o_2,p,u}, \underline{x}_{o_3,p,u}]$. According to these values, we would like to know how far each QoS indicator $x_{o,p,u}$ is from its own requirement $\bar{x}_{o,p,u}$. For each objective, this element is calculated as follows:

$$\underline{x}_{o,p,u} = \begin{cases} \bar{x}_{o,p,u} - x_{o,p,u}, & o = o_1, \bar{x}_{o,p,u} > x_{o,p,u}, \\ x_{o,p,u} - \bar{x}_{o,p,u}, & o \in \{o_2, o_3\}, x_{o,p,u} > \bar{x}_{o,p,u}, \\ 0, & otherwise. \end{cases} \tag{2}$$

We try to highlight when these requirements are not met in a momentary state in order to find out the best policy of scheduling decisions that can immediately improve the performance of QoS provisioning. Finally, $\mathbf{q} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_P]$ is the vector of queue lengths, $\mathbf{q}_p = [q_{p,u_1}, q_{p,u_2}, ..., q_{p,u_{U_p}}]$, where $q_{p,u}$ is the queue length of user $u \in \mathcal{U}_p$ of class $p \in \mathcal{P}$.

B. Action Space

The action space we are referring to is discrete and two-dimensional being defined as: $\mathcal{A} = \{\mathbf{a}_1, ... , \mathbf{a}_{D \times P}\}$, where the action taken at TTI t $\mathbf{a}[t] = [p, d]$ gives to traffic class $p \in \mathcal{P}$ the first priority to be scheduled and selects the scheduling rule $d \in \mathcal{D}$ to perform the frequency prioritization. Given the uncertainty $\mathbf{v}[t] = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_P]$, it is likely that the controllable momentary state $\mathbf{c}[t] = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_P]$ will evolve according to the decided action $\mathbf{a}[t] = [p, d]$. Then, we denote by $\mathbf{c}'_{(p,d)} = [\mathbf{x}'_{(p,d)}, \underline{\mathbf{x}}'_{(p,d)}, \mathbf{q}'_{(p,d)}]$ the controllable state that evolves according to the selected action and scheduling transition function $f : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}^C$:

$$\mathbf{c}'_{(p,d)} = f(\mathbf{s}, p, d). \tag{3}$$

The reward is calculated according to $\mathbf{c}'_{(p,d)}[t+1]$ at each TTI.

C. Reward Function

The reward function measures the performance of applying a given action $\mathbf{a}[t] = [p, d] \in \mathcal{A}$ in state $\mathbf{s}[t] \in \mathcal{S}$ such as [25]:

$$r(\mathbf{s}, p, d) \overset{\mathbf{(def)}}{=} \mathbb{E}\Big[\mathcal{R}_{t+1} | \mathbf{s}[t] = \mathbf{s}, \mathbf{a}[t] = [p, d]\Big], \tag{4}$$

where $\mathcal{R}_{t+1}$ is the reward value at TTI t+1 and the expectation $\mathbb{E}[\cdot]$ is given due to the fact that $\mathbf{s}[t] \in \mathcal{S}$ is considered as random such that $\mathbb{P}(\mathbf{s}[t] = \mathbf{s}) > 0$ and $\mathbb{P}(\mathbf{a}[t] = [p, d]]) > 0$ holds for all $p \in \mathcal{P}$ and $d \in \mathcal{D}$. Under its original form, the reward function depends on both controllable and uncontrollable states. This makes the computation more difficult since the CQI reports for each user $u \in \mathcal{U}_p$ and class $p \in \mathcal{P}$ need to be taken into account at each TTI. This may actually involve some pre-processing stages to compress the CQI state for each traffic class that can increase the complexity of *5MART* framework.

Theorem 1: For any prioritized traffic $p \in \mathcal{P}$ and selected scheduling rule $d \in \mathcal{D}$ in state $\mathbf{s}[t] = \mathbf{s}$, the reward will be a function of consecutive controllable states, such as: $r(\mathbf{s}, p, d) = r(\mathbf{c}'_{(p,d)}, \mathbf{c}, p, d)$ (proof provided in Appendix A).

In real-time systems, the reward value for each applied action is not known in advance. We can only determine its value once the scheduling is performed and the entire system evolves in the next state, at TTI t+1, for example. Here, the reward function can be considered as: $r(\mathbf{c}'_{(p,d)}, \mathbf{c}, p, d) = r(\mathbf{c}', \mathbf{c})$. In the reward computation, we would like to know how much the QoS indicators $\mathbf{x}'[t+1]$ have been improved when compared to $\mathbf{x}[t]$ from the previous state. We are implementing our proposed reward function as follows:

$$r(\mathbf{c}', \mathbf{c}) = R(\mathbf{c}') - R(\mathbf{c}), \tag{5}$$

where, $R : \mathcal{S}^C \rightarrow \mathbb{R}_{[0,1]}$ is the reward value that evaluates the performance of meeting the QoS requirements for all traffic classes. However, in (5) when all QoS requirements are met in between two consecutive states, and $R(\mathbf{c}') = R(\mathbf{c}) = 1$, the reward is $r(\mathbf{c}', \mathbf{c}) = 1$. When computing $R(\mathbf{c})$, the standardized and static prioritization between traffic classes must be considered in respect of $\mathcal{P} = \{1, 2, ...p, .., P\}$. To this end, we propose the following computation:

$$R(\mathbf{c}) = \sum_{p=1}^{P} \frac{P + 1 - p}{\sum_{h=1}^{P} h} \cdot R_p(\mathbf{c}_p), \tag{6}$$

where R_p is the QoS revenue for traffic class $p \in \mathcal{P}$. The weight in (6) sets the importance on meeting the QoS requirements for each traffic class by respecting the order of $\mathcal{P}$. The *5MART* learning framework aims to always maximize $R(\mathbf{c})$ and avoids the starvation of some lower priority classes by properly selecting at each TTI the most suitable class to be prioritized. The intra-class reward $R_p(\mathbf{c}_p)$ is determined by weighting the QoS revenues for each active UE, such that:

$$R_p(\mathbf{c}_p) = \sum_{u=u_1}^{u_{U_p}} \frac{1}{U_p} \cdot R_{p,u}(\mathbf{c}_{p,u}), \tag{7}$$

where $\mathbf{c}_{p,u} = [\mathbf{x}_{p,u}, \underline{\mathbf{x}}_{p,u}, q_{p,u}]$. The QoS revenue of each user $u \in \mathcal{U}_p$ is calculated based on the following equation:

$$R_{p,u}(\mathbf{c}_{p,u}) = \frac{1}{3} \sum_{o=o_1}^{o_3} R_{o,p,u}(\mathbf{c}_{o,p,u}), \tag{8}$$

where, $\mathbf{c}_{o,p,u} = [x_{o,p,u}, \underline{x}_{o,p,u}, q_{p,u}]$. The sub-reward $R_{o,p,u}$ is determined according to GBR, PLR or delay objectives:

$$R_{o,p,u} = \begin{cases} 1 - \frac{\underline{x}_{o,p,u}}{\bar{x}_{o,p,u}}, & o = o_1, \underline{x}_{o,p,u} > 0, q_{p,u} \neq 0, \\ 1 - \frac{\underline{x}_{o,p,u}}{x_{o,p,u}}, & o = \{o_2, o_3\}, \underline{x}_{o,p,u} > 0, q_{p,u} \neq 0, \\ 1, & otherwise. \end{cases} \tag{9}$$

When the reward $R_{o,p,u} = 1$, the objective $o \in \mathcal{O}$ is met.

The purpose of the *5MART* framework is to increase the number of TTIs when all objectives $o \in \mathcal{O}$ are met for the entire set of active users $u \in \mathcal{U}_p$ of each traffic class $p \in \mathcal{P}$. However, if $R(\mathbf{c}') \geq R(\mathbf{c})$, then the applied action has a positive impact in improving the overall QoS provisioning.

An important objective to be considered when maximizing the global reward $r(\mathbf{c}', \mathbf{c})$ is the PLR reduction. In general, the PLR indicator for each user encapsulates two sub-components: *a)* packet loss due to congestion; *b)* packet loss due to channel errors. Both elements must be minimized in order to meet the PLR requirements at each TTI. Packet loss due to congestion can happen when failing to meet: *a)* the GBR objective, since the user decoder may drop certain packets if the rate requirements are not respected; *b)* the delay objective, since packets waiting in the scheduler queue longer than the given time bounds are dropped and declared lost. Consequently, the *5MART* meta-scheduler aims to minimize this packet loss by properly selecting the scheduling rule oriented on GBR or delay objectives. Moreover, lower delay requirements than the ones specified by standards are set in order to improve the delay budget and minimize the packet loss even more. To reduce the packet loss induced by channel errors, the PLR-based scheduling rule can be selected in certain states. By dynamically selecting the traffic to be prioritized at each TTI, the service class requesting higher number of re-transmissions can get more frames to meet the PLR objective.

D. Value Functions

We define the scheduling policy as a function of $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ that maps scheduler states distributions over discrete action space [25]. For the purpose of time and frequency prioritization decisions, we define the stochastic policy $\pi[(p, d) \mid \mathbf{s}]$ as the probability of an action $\mathbf{a}[t] = [p, d]$ to be selected in state $\mathbf{s}[t] = \mathbf{s}$, being written as follows:

$$\pi[(p, d) \mid \mathbf{s}] = \mathbb{P}[\mathbf{a}[t] = [p, d] | \mathbf{s}[t] = \mathbf{s}]. \tag{10}$$

According to the reward value received in each state and the scheduling policy defined in (10), we would like to find the function that can measure the sum of rewards from each state-to-state iteration that can eventually guarantee the best prioritization decisions in each state when starting with a given and random initial state $\mathbf{s}[0] = \mathbf{s}$. In this sense, we need a function that measures the value of accumulated rewards from the initial state $\mathbf{s}[0]$ and underlies the given scheduling policy $\pi[(p, d) \mid \mathbf{s}]$. In the literature, this function is known as *state-value* or simpler, *value function*, defined as follows [25]:

$$V^{\pi}(\mathbf{s}) \overset{(\mathbf{def})}{=} \mathbb{E}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} | \mathbf{s}[0] = \mathbf{s}\Big], \tag{11}$$

where, 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 $\gamma \in [0, 1]$. This function needs to be redefined in order to keep the same form as the computed reward function which depends strictly on two-consecutive controllable states only.

Theorem 2: For any policy π that aims to optimize the maximization problem in (1) and being constrained by (1.a)-(1.m), the new value function $H^{\pi} : \mathcal{S}^C \times \mathcal{S}^C \rightarrow \mathbb{R}$ becomes:

$$H^{\pi}(\mathbf{c}', \mathbf{c}) = \mathbb{E}_{\pi}\Big[\sum_{t=1}^{\infty} \gamma^{t-1} \mathcal{R}_t | \mathbf{c}[1] = \mathbf{c}', \mathbf{c}[0] = \mathbf{c}\Big], \tag{12}$$

where the scheduling policy takes the form of $\pi[(p, d) \mid (\mathbf{c}', \mathbf{c})]$ denoting the probability of prioritizing traffic class $p \in \mathcal{P}$ and selecting the scheduling rule $d \in \mathcal{D}$ when the current state is $(\mathbf{c}', \mathbf{c})$. The proof is provided in Appendix B.

We thus learn to apply the best prioritization decisions based on consecutive controllable states and we eliminate the direct dependency on the uncertainty given by the uncontrollable elements. The aim of the learning stage is to update the value function $H^{\pi}(\mathbf{c}', \mathbf{c})$ by following a given policy π at each iteration in order to represent each state as accurate as possible. We need a relation of this function within two consecutive new states such as: $(\mathbf{c}', \mathbf{c})$ and $(\mathbf{c}'', \mathbf{c}')$. By following (12) and considering the reasoning from the temporal difference learning [16], the value function can be rewritten as follows:

$$H^{\pi}(\mathbf{c}', \mathbf{c}) = r(\mathbf{c}', \mathbf{c}, p, d) + \gamma \cdot H^{\pi}(\mathbf{c}'', \mathbf{c}'), \tag{13}$$

where $\mathbf{c}'' = \mathbf{c}[t+2]$. Therefore, we are at TTI t+2 and we would like to update the value function based on the reward value $r(\mathbf{c}', \mathbf{c}, p, d)$ when class $p \in \mathcal{P}$ and scheduling rule $d \in \mathcal{D}$ are applied in state $(\mathbf{c}', \mathbf{c})$. The aim is to update the value function based on high number of state-to-state iterations until the optimality condition is met. The optimal value $H^*(\mathbf{c}', \mathbf{c})$ of state $(\mathbf{c}', \mathbf{c}) \in \mathcal{S}^C \times \mathcal{S}^C$ is the highest expected return when the entire scheduling process is started from state $(\mathbf{c}', \mathbf{c})$. The optimal function $H^* : \mathcal{S}^C \times \mathcal{S}^C \rightarrow \mathbb{R}$ is determined as: $H^*(\mathbf{c}', \mathbf{c}) = max_{\pi} H^{\pi}(\mathbf{c}', \mathbf{c})$ [25]. By reloading (13), we have:

$$H^*(\mathbf{c}', \mathbf{c}) = r(\mathbf{c}', \mathbf{c}, p, d) + \gamma \cdot H^*(\mathbf{c}'', \mathbf{c}'). \tag{14}$$

As the scheduler states $(\mathbf{c}', \mathbf{c})$ cannot be enumerated exhaustively, the optimality of $H^{\pi}(\mathbf{c}', \mathbf{c})$ cannot be guaranteed. Then, we use a neural network to approximate the optimal value function such that $\bar{H}^*(\mathbf{c}', \mathbf{c}) \approx H^*(\mathbf{c}', \mathbf{c})$. In the learning stage, the weights of the neural network $\bar{H}^*(\mathbf{c}', \mathbf{c})$ are refined according to the selected actions in such a way that the difference/error between the target and NN output is minimized.

IV. 5MART Scheduler Framework

The state $(\mathbf{c}', \mathbf{c}) \in \mathcal{S}^C \times \mathcal{S}^C$ has a variable dimension because the number of users U_p may change from one TTI to another for each class $p \in \mathcal{P}$. In order to avoid this drawback and reduce the complexity of *5MART* framework, we propose a compression scheme for the scheduler state space, where its components can be modeled as normally distributed variables.

A. State Compression

Let us regroup the controllable instantaneous state as follows: $\mathbf{c} = [\mathbf{x}_{o_1}, \mathbf{x}_{o_2}, \mathbf{x}_{o_3}, \underline{\mathbf{x}}_{o_1}, \underline{\mathbf{x}}_{o_2}, \underline{\mathbf{x}}_{o_3}, \mathbf{q}]$, where $\mathbf{c} = [\mathbf{c}_k]$ with $k = \{1, .., 7\}$. Each component is decomposed per traffic class such as: $\mathbf{c}_k = [\mathbf{c}_{k,p}]$ where $p = \{1, .., P\}$. Considering a fixed number of traffic classes, we apply statistical modeling on each component $\mathbf{c}_{k,p} = [\mathbf{c}_{k,p,u}]$, where $u \in \mathcal{U}_p$. Then, these elements are normalized using the formula:

$$\hat{c}_{k,p,u} = \frac{c_{k,p,u}}{\frac{1}{U_p} \cdot \sum_{u'} c_{n,p,u'}}. \tag{15}$$

In (15), each normalized component $\hat{c}_{k,p,u}$ can be considered as a product of random variables due to the uncertainty given by (3), in which the evolution of controllable state at TTI t+1 depends on $\mathbf{s}[t] \in \mathcal{S}$ that includes both $\mathbf{c}[t] \in \mathcal{S}^C$ and $\mathbf{v}[t] \in \mathcal{S}^U$. According to [17], each vector $\hat{c}_{k,p} = [\hat{c}_{k,p,u_1}, \hat{c}_{k,p,u_2}, ..., \hat{c}_{k,p,u_{U_p}}]$, $k = \{1, 2, ..., 7\}$, $p = \{1, 2, ..., P\}$, can be modeled as normally distributed variables. This allows the implementation of the mean and standard deviation functions based on the maximum likelihood estimators [17]. These functions take the following forms:

$$\mu(\hat{c}_{k,p}) = \frac{1}{U_p} \cdot \sum_{u=u_1}^{u_{U_p}} ln(\hat{c}_{k,p,u}), \tag{16.a}$$

$$\sigma(\hat{c}_{k,p}) = \sqrt{\frac{1}{U_p} \cdot \sum_{u=u_1}^{u_{U_p}} [ln(\hat{c}_{k,p,u}) - \mu(\hat{c}_{k,p})]^2}, \tag{16.b}$$

where $\{\mu(\hat{c}_{k,p}), \sigma(\hat{c}_{k,p})\} : \mathbb{R}^{U_p} \rightarrow [-1, 1]$ are the mean and standard deviation functions, respectively. These functions are able to provide general statistics on QoS provisioning for each traffic class. Using (16.a) and (16.b) for all controllable components, the current state dimension of $7 \times (U_1[t] + U_2[t] + ... + U_P[t]) \times 2$ will be reduced after compression to $7 \times 2P \times 2$. Thus, we define a representation for the controllable state at each TTI t, such as: $\hat{\mathbf{c}} = [\hat{\mathbf{x}}, \underline{\hat{\mathbf{x}}}, \hat{\mathbf{q}}]$, where: *(a)* $\hat{\mathbf{x}} = [\hat{\mathbf{x}}_{o_1}, \hat{\mathbf{x}}_{o_2}, \hat{\mathbf{x}}_{o_3}]$ is the vector of compressed QoS values and $\hat{\mathbf{x}}_o = [\mu(\hat{\mathbf{x}}_{o,1}), \sigma(\hat{\mathbf{x}}_{o,1}), ..., \mu(\hat{\mathbf{x}}_{o,P}), \sigma(\hat{\mathbf{x}}_{o,P})]$; *(b)* $\underline{\hat{\mathbf{x}}} = [\underline{\hat{\mathbf{x}}}_{o_1}, \underline{\hat{\mathbf{x}}}_{o_2}, \underline{\hat{\mathbf{x}}}_{o_3}]$ is the compressed vector of differences between the instantaneous QoS and the corresponding requirements; $\underline{\hat{\mathbf{x}}}_o = [\mu(\underline{\hat{\mathbf{x}}}_{o,1}), \sigma(\underline{\hat{\mathbf{x}}}_{o,1}), ..., \mu(\underline{\hat{\mathbf{x}}}_{o,P}), \sigma(\underline{\hat{\mathbf{x}}}_{o,P})]$; *(c)* $\hat{\mathbf{q}} = [\mu(\hat{\mathbf{q}}_1), \sigma(\hat{\mathbf{q}}_1), ..., \mu(\hat{\mathbf{q}}_P), \sigma(\hat{\mathbf{q}}_P)]$ is the compressed queue vector. For simplicity, we consider $\mathbf{z} = [\hat{\mathbf{c}}', \hat{\mathbf{c}}]$ as the previous controller state and $\mathbf{z}' = [\hat{\mathbf{c}}'', \hat{\mathbf{c}}']$ the current one. Thus, the *5MART* framework learns based on the tuple $\{\mathbf{z}', \mathbf{z}\}$ the approximations of the best action pairs $\mathbf{a}[t] = [p, d]$ to be applied in each current scheduler state.

B. Approximations of Scheduling Decisions

Fig. 2 illustrates the architecture of our proposed *5MART* controller. We approximate the optimal value function by $\bar{H}^* : \hat{\mathcal{S}} \times \hat{\mathcal{S}} \rightarrow [-1, 1]$, where $\hat{\mathbf{c}} \in \hat{\mathcal{S}}$. However, this function relates only the state value according to some applied actions. Hence, an additional function approximator that decides the action to be applied in each state is defined and labeled as action-value function: $A : \hat{\mathcal{S}} \times \hat{\mathcal{S}} \rightarrow [-1, 1]^N$, where N is the output dimension, set based on the number of traffic classes to be prioritized and the number of scheduling rules. Then, the non-linear representations of these functions are defined as follows:

$$\begin{aligned} \bar{H}^*(\mathbf{z}) &= g^v[\theta^v_t, \varphi(\mathbf{z})], \\ A(\mathbf{z}) &= g^a[\theta^a_t, \varphi(\mathbf{z})], \end{aligned} \tag{17}$$

where $\{g^v, g^a\}$ are the neural networks corresponding to state-value and action-value functions, respectively, $\varphi(\mathbf{z})$ is the feature vector and $\{\theta^v_t, \theta^a_t\}$ are vectors with NN weights that must be tuned at each TTI in order to get good approximations of optimal state-value and action-value functions, respectively.

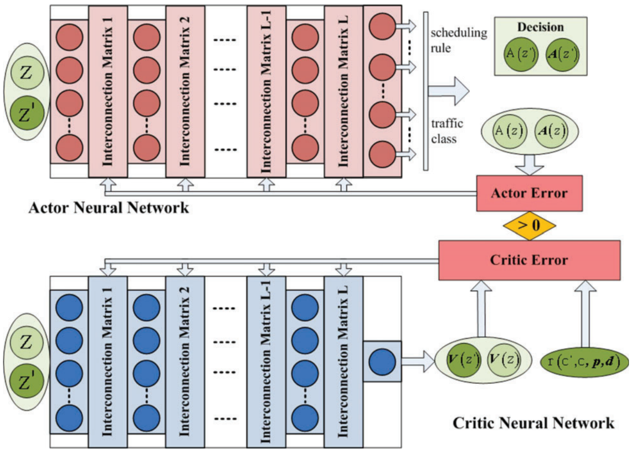


Fig. 2. Proposed 5MART controller.

As seen in Fig. 2, the architecture of neural networks contains a number of layers interconnected by weight matrices. For our *5MART* model, we consider the same topology for layers and interconnection matrices for both state-value and action-value functions. The only difference is that the output layer for the state-value function has one output pin, whereas the output layer of the action-value NN has a number of N pins. In general, a neural network is composed of L number of layers considering the hidden and output layers only, while the input layer is not counted. Then, the number of hidden layers for each neural network is $L_H = L - 1$. Each layer is composed of a number of N_l nodes or linear/nonlinear transformations, where $l \in \{1, 2, ..., L+1\}$. Given the fact that the number of nodes for the input and output layers are known for both state-value and action-value NNs ($N_1 = 7 \times 2P \times 2$, $N^v_{L+1} = 1$ and $N^a_{L+1} = N$), then the number of hidden layers (L_H) and number of neurons ($N_l, 2 \leq l \leq L$) are determined beforehand based on a priori tests. Furthermore, we consider the interconnection matrix of weights $\mathbf{W}_{l,l+1} = \{\mathbf{w}_y, y = 1, .., N_l\} = \{w_{y,i}; y = 1, .., N_l, i = 1, ..., N_{l+1}\}$ between layer l and l+1, where $l \in \{1, 2, ..., L\}$. In general, the functional form of a neural network when the momentary state $\mathbf{z} \in \hat{\mathcal{S}} \times \hat{\mathcal{S}}$ is Forward Propagated (FP) from the input to the output layer takes the following form:

$$FP(\overrightarrow{\mathbf{z}}) = \overrightarrow{\mathfrak{L}}_L ... \overrightarrow{\mathfrak{L}}_{l+1} \overrightarrow{\mathfrak{L}}_l ... \overrightarrow{\mathfrak{L}}_1(\overrightarrow{\mathbf{z}}), \tag{18}$$

where $\overrightarrow{\mathbf{z}} = \mathbf{z}^T$ and $\mathfrak{L}_l$ is the layer operator that takes as input the output vector of layer l, $\mathbf{z}^{(l)}$ (including the bias point) with a dimension of $N_l + 1$ and provides as output the vector $\mathbf{z}^{(l+1)}$ with the dimension of N_{l+1}. The operator for each layer is determined based on the following equation:

$$\overrightarrow{\mathfrak{L}}_l(\overrightarrow{\mathbf{z}}^{(l)}+) = \psi_{l+1}(\mathbf{W}^{+T}_{l,l+1} \cdot \overrightarrow{\mathbf{z}}^{(l)}+), \tag{19}$$

where $\overrightarrow{\mathbf{z}}^{(l)}+ = [z^{(l)}_1, ..., z^{(l)}_{N_l}, z^{(l)}_{N_l+1}]^T$ is the biased input vector meaning that a bias node $z^{(l)}_{N_l+1}$ is added to vector of nodes $\mathbf{z}^{(l)}$ of each layer l excepting the output layer. A bias node is always set to $z^{(l)}_{N_l+1} = 1$ aiming to increase the flexibility of the neural network in order to fit more generalized input data. A bias node is added at each layer by extending the original interconnection matrix of weights $\mathbf{W}_{l,l+1}$ between layers l and $l+1$ as follows: $\mathbf{W}^+_{l,l+1} = \{\mathbf{w}^+_y, y = 1, .., N_l, N_l + 1\}$, where $\mathbf{w}_{N_l+1} = [w_{N_l+1,1}, ..., w_{N_l+1,N_{l+1}}]$ is the additional vector of

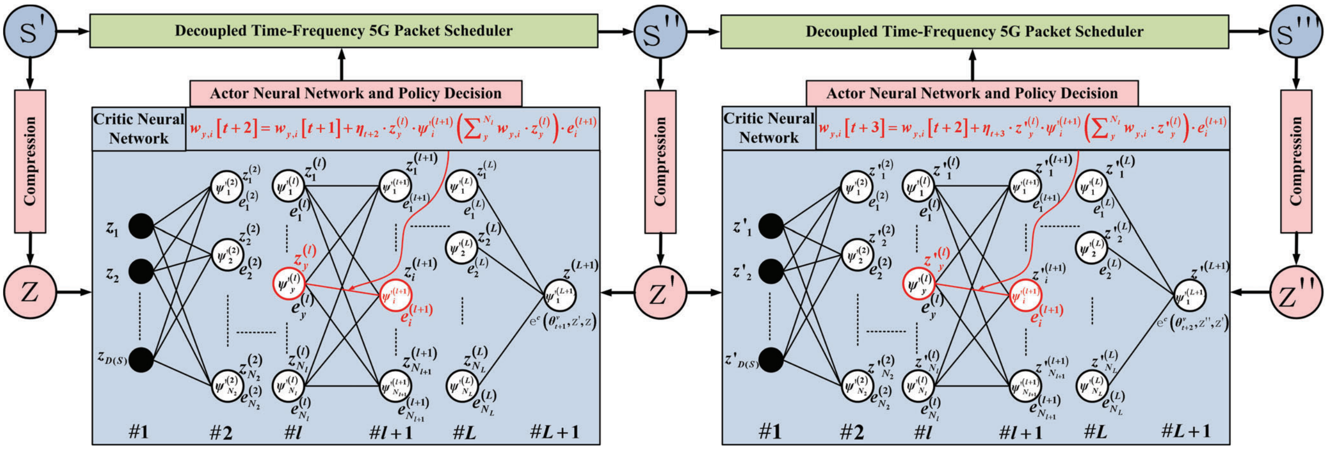

S'
Decoupled Time-Frequency 5G Packet Scheduler
S''
Decoupled Time-Frequency 5G Packet Scheduler
S'''
Compression
Actor Neural Network and Policy Decision
Critic Neural Network
$w_{y,i}[t+2]=w_{y,i}[t+1]+\eta_{t+2}\cdot z_y^{(l)}\cdot\psi_i'^{(l+1)}\left(\sum_y^{N_l} w_{y,i}\cdot z_y^{(l)}\right)\cdot e_i^{(l+1)}$
$w_{y,i}[t+3]=w_{y,i}[t+2]+\eta_{t+3}\cdot z'^{(l)}_y\cdot\psi_i'^{(l+1)}\left(\sum_y^{N_l} w_{y,i}\cdot z'^{(l)}_y\right)\cdot e_i^{(l+1)}$
Z
Z'
Z''
z_1
z_2
$z_{D(S)}$
z'_1
z'_2
$z'_{D(S)}$
$e^c(\theta^v_{t+1},z',z)$
$e^c(\theta^v_{t+2},z'',z')$
#1
#2
#l
#l+1
#L
#L+1

Fig. 3. State transition and refinement of critic NN weights.

weights connecting the bias point of layer l with the hidden nodes from layer $l+1$. The bias points could be beneficial especially when all input state elements are 0 (the output of neural networks would always be 0) and the tuned bias weights $\mathbf{w}_y^+$ can help the neural network to adjust its output to its corresponding target value given by (14). At each TTI t, a total number of $\sum_{l=1}^{L}(N_l+1)\cdot N_{l+1}$ weights for each neural network is tuned until a given convergence criterion is met. In (19), $\psi_l:\mathbb{R}^{N_l}\rightarrow\mathbb{R}^{N_l}$ is the vector of activation functions defined as $\psi_l=[\psi_1^{(l)},..,\psi_{N_l}^{(l)}]^T$. The same type of activation function is used by all nodes within one layer.

C. 5MART RL Algorithm

The proposed *5MART* framework uses actor-critic RL algorithm, in which the action-value or the actor neural network takes scheduling prioritization decisions and the state-value or the critic neural network criticizes the action taken at each current state. During the learning stage, the *5MART* framework takes scheduling decisions according to a given policy. We assume the scheduling process at TTI $t+2$, where the current controller state is $\mathbf{z}'=[\hat{\mathbf{c}}'',\hat{\mathbf{c}}']\in\hat{\mathcal{S}}\times\hat{\mathcal{S}}$. As shown in Fig. 2, the critic NN determines at each iteration the error between the target value at state $\mathbf{z}'$ based on (14) and the forwarded value at state $\mathbf{z}=[\hat{\mathbf{c}}',\hat{\mathbf{c}}]\in\hat{\mathcal{S}}\times\hat{\mathcal{S}}$. If this error is greater than zero, then the action taken in the previous state $\mathbf{z}$ is a good choice and the actor NN must be reinforced. We define the critic error function as $e^c:[-1,1]^{4Dim(\hat{\mathcal{S}})+1}\rightarrow[-1,1]$ being calculated at each TTI by following the equation [22]:

$$e^c_{t+2}(\theta^v_{t+1},\mathbf{z}',\mathbf{z})=r(\mathbf{z},p,d)+\gamma\cdot\bar{H}^*(\mathbf{z}')-\bar{H}^*(\mathbf{z}),\quad(20)$$

where the action taken in the previous state $\mathbf{a}[t+1]=[p,d]$ is decoded from the decision vector provided by the actor NN.

The actor decision can select the improvement step in which a vector with random elements is preferred instead of selecting the output vector of the actor neural network; or it can select directly the output of the actor network if the exploitation step is decided. If $\mathbf{A}$ is the actor decision, then the policy of selecting $\mathbf{A}[t+2]=\mathbf{A}$ in state $\mathbf{z}'[t+2]=\mathbf{z}'$ is:

$$\pi(\mathbf{A}\mid\mathbf{z}')=\begin{cases}[\epsilon^{(1)}_{t+2},\epsilon^{(2)}_{t+2},...,\epsilon^{(N)}_{t+2}] & \upsilon_{t+2}\geq\epsilon_{t+2},\\ [z_1'^{(L+1)},...,z_N'^{(L+1)}] & \upsilon_{t+2}<\epsilon_{t+2},\end{cases}\quad(21)$$

where $[\epsilon^{(1)}_{t+2},\epsilon^{(2)}_{t+2},...,\epsilon^{(N)}_{t+2}]$ is the decision vector with random variables, υ_t is a random variable at each TTI and ϵ_t is the decision parameter that aims to select at each TTI whether an improvement or exploitation step is decided. When ϵ_t is small, more improvements steps are decided, while when ϵ_t increases, the actor NN exploits more its output decisions.

In the learning stage, the critic neural network is updated at each TTI based on the error $e^c(\theta^v_{t+1},\mathbf{z}',\mathbf{z})$ which is calculated according to (20). If $e^c[t+2]\geq 0$, then the previous action taken based on (21) in state $\mathbf{z}$ is a good option and the actor NN must be updated by reinforcing the error vector:

$$\mathbf{e}^a_{t+2}(\theta^a_{t+1},\mathbf{z})=\mathbf{A}-A(\mathbf{z}),\quad(22)$$

where $\mathbf{e}^a_{t+2}(\theta^a_{t+1},\mathbf{z}):[-1,1]^{2Dim(\hat{\mathcal{S}})+1}\rightarrow[-1,1]^N$ is the actor error function. When compared to the critic NN where only one output error needs to be reinforced, the actor NN reinforces the error vector $\mathbf{e}^a=[e^a_1,e^a_2,...,e^a_N]$ each time when $e^c\geq 0$, where N depends on P and D.

The entire set of errors $\{e^c,e^a_1,e^a_2,...,e^a_N\}$ are back-propagated through the critic and actor neural networks, respectively, with the scope of updating the matrix of weights $\mathbf{W}^+_{l,l+1}$ from layer to layer. The back-propagation follows the same principle for both neural networks, with the only difference that the actor NN reinforces a number of N errors from the output layer. The functional form of the Back-Propagation (BP) procedure is given by:

$$BP(\overleftarrow{\mathbf{e}})=\overleftarrow{\mathfrak{L}}_1...\overleftarrow{\mathfrak{L}}_l\overleftarrow{\mathfrak{L}}_{l+1}...\overleftarrow{\mathfrak{L}}_L(\overleftarrow{\mathbf{e}}),\quad(23)$$

where $\overleftarrow{\mathbf{e}}=\mathbf{e}^T$ and operator $\overleftarrow{\mathfrak{L}}_l$ takes the errors $\overleftarrow{\mathbf{e}}^{(l+1)}$ as input from the output of layer $l+1$ and gives the vector of errors $\overleftarrow{\mathbf{e}}^{(l)}$ as output at the output of layer l [26]:

$$\overleftarrow{\mathfrak{L}}_l(\overleftarrow{\mathbf{e}}^{(l)})=\mathbf{W}^+_{l,l+1}\cdot\nabla\psi_{l+1}[\mathbf{W}^{+T}_{l,l+1}\cdot\overrightarrow{\mathbf{z}}^{(l)}+]\odot\overleftarrow{\mathbf{e}}^{(l+1)},\quad(24)$$

where $l=\{L,...,2,1\}$, $\nabla\psi_l=[\psi_1'^{(l)},\psi_2'^{(l)},...,\psi_{N_l}'^{(l)}]^T$, $\overleftarrow{\mathbf{e}}^{(l)}=[\mathbf{e}^{(l)}_1,...,\mathbf{e}^{(l)}_{N_l}]^T$ and $\odot$ is the Hadamard product.

Once the FP and BP computations are completed, the value of each weight must be updated. Fig. 3 depicts the transition between three consecutive states, where within each pair of states, the way how each weight is updated for the critic NN is detailed. On each node y of each layer l, the following computations are available: *(a)* $z_y^{(l)}$ as a result of FP computation of momentary state $\mathbf{z}[t+1]$ on each node; *(b)* $e_y^{(l)}$ as a result of the BP computation on each node of the critic error calculated based on (20); and *(c)* the derivative activation function $\psi_y'^{(l)}$ obtained by propagating $\mathbf{z}$ through each layer l and node y. Therefore, each weight $w_{y,i}$ that interconnects the node $y = 1, 2, ..., N_l + 1$ of layer l to node $i = 1, 2, ..., N_{l+1}$ of layer l+1 is updated at TTI t+2 such as [26]:

$$w_{y,i}[t+2] = w_{y,i}[t+1] + \eta_{t+2} \cdot z_y^{(l)} \cdot \psi_i'^{(l+1)} \cdot e_i^{(l+1)}, \quad (25)$$

where $\eta_t \in [0, 1]$ is the learning rate. The actor NN follows the same computations when the critic error $e^c \geq 0$.

Both learning rate η_t and decision parameter ϵ_t play important roles and must be updated at each TTI as the learning process evolves. Parameter ϵ_t is incremented with a given step in such a way that, at the beginning of the learning stage more improvements steps are chosen, whereas at the end of the leaning stage the controller experiences a higher number of exploitation steps. On the other hand, the learning rate η_t must be decremented as the learning process evolves from higher to lower values since it is expected that at the end of the learning stage the interconnection matrices will be less updated.

D. Decoding Actor's Decisions

In both learning and exploitation stages, the multi-dimensional and continuous actor decisions $\mathbf{A}(\mathbf{z})$ and $A(\mathbf{z})$, respectively, must be decoded in two-dimensional and discrete action space $\mathbf{a}[t] = [p, d] \in \mathcal{A}$ according to the maximum number of classes P and scheduling rules D. Let us assume that N_P is the number of output pins needed to decode the traffic class prioritization and N_D the number of pins used to decode the scheduling rule to be applied each TTI for the frequency prioritization, where $N = N_P + N_D$. If we set $N_P = P$ and $N_D = D$, then the scheduling decisions are taken as follows: *a)* at each TTI, in the time domain, the traffic class with the highest output pin value gets the highest priority to be scheduled; *b)* similarly, the scheduling rule with the highest pin value is selected to perform the frequency prioritization for the traffic class selected in *a)*. Therefore, part of the decision vector of the actor NN will be unused and consequently, the errors reinforced on those pins are noisy. This can lead to a very poor generalization of the input state space and can affect the learning performance of both NNs.

Our proposed decoding technique aims to set a threshold for each output pin. Then, the number of traffic classes can be decomposed as follows:

$$P = \sum\nolimits_x^X 2^{p_x} + p_y, \ \{p_x, p_y\} \in \mathbb{N}, \ p_{x_1} \neq p_{x_2}, \forall \{x_1, x_2\} \in \mathbb{N}. \quad (26)$$

Using this decomposition, the output pins are clustered into X groups, where a group of p_x pins are decoded together. Conversely, a number of p_y pins are decoded independently. Therefore, the number of pins needed to decode the traffic class prioritization at each TTI becomes:

$$N_P = p_y + \sum\nolimits_x^X p_x. \quad (27)$$

The same principle is applied when deciding the number of output pins N_D necessary to decode the scheduling rule index in each current state. However, by using (27), all output pins of actor NN are used when taking the scheduling decisions.

Algorithm 1: 5MART Scheduling Framework based on Actor-Critic RL Algorithm

```
for each TTI $t$+2
    observe current state $(\mathbf{c}'', \mathbf{c}')$, recall the previous state
    $(\mathbf{c}', \mathbf{c})$ and previous action $\mathbf{a}[t+1] = [p, d]$
    calculate the reward $r(\mathbf{c}', \mathbf{c}, p, d)$ based on (4)-(9)
    compress both consecutive states and get $[\mathbf{z}', \mathbf{z}]$
    according to (15), (16.a) and (16.b)
    forward propagate compressed states $[\mathbf{z}, \mathbf{z}']$ through
    $g^v[\theta_{t+1}^v, \varphi]$ based on (18) and (19)
    calculate the critic error $\mathbf{e}^c(\theta_{t+1}^v, \mathbf{z}', \mathbf{z})$ based on (20)
    // criticize previous action $\mathbf{a}[t+1] = [p, d]$
    if $\mathbf{e}^c(\theta_{t+1}^v, \mathbf{z}', \mathbf{z}) \geq 0$
        forward propagate the compressed state $\mathbf{z} \in \hat{\mathcal{S}} \times \hat{\mathcal{S}}$
        through the actor NN $g^a[\theta_{t+1}^a, \varphi]$ based on (18)-(19)
        calculate the multi-dimensional error $\mathbf{e}^a(\theta_{t+1}^a, \mathbf{z})$
        based on (22)
        back propagate error vector $\mathbf{e}^a = [e_1^a, e_2^a, ..., e_N^a]$
        based on (23)-(24)
        update weights $\theta_{t+1}^a$ according to (25)
    end if
    back propagate error $\mathbf{e}^c(\theta_{t+1}^v, \mathbf{z}', \mathbf{z})$ based on (23)-(24)
    update weights $\theta_{t+1}^v$ according to (25)
    // act based on the learned policy
    determine the decision vector to follow in current state
    $\mathbf{z}'$ based on (21)
    decode the decision vector $\mathbf{A}(\mathbf{z}')$ into discrete version
    $\mathbf{a}[t+2] = [p', d']$ according to (26) and (27)
end for
```

Algorithm I summarizes the proposed *5MART* scheduling framework. At TTI t+2, a new state $(\mathbf{c}'', \mathbf{c}')$ is observed and the reward $r(\mathbf{c}', \mathbf{c}, p, d)$ is determined based on (4)-(9). According to Sub-section IV.A, the perceived state $(\mathbf{c}'', \mathbf{c}')$ is compressed onto $\mathbf{z}' = [\hat{\mathbf{c}}'', \hat{\mathbf{c}}']$ and the previous state $\mathbf{z}$ is recalled. Both compressed states are propagated through the critic neural network and the error $\mathbf{e}_{t+2}^c(\theta_{t+1}^v, \mathbf{z}', \mathbf{z})$ is calculated according to (20). If the critic error is $\mathbf{e}_{t+2}^c(\theta_{t+1}^v, \mathbf{z}', \mathbf{z}) \geq 0$, then the actor NN is updated as follows: *a)* the previous compressed state $\mathbf{z}$ is propagated through the actor NN and the previous decision vector is recalled; *b)* the multi-dimensional actor error $\mathbf{e}_{t+2}^a(\theta_{t+1}^a, \mathbf{z})$ is determined based on (22); this error is back-propagated by using (23), (24), and the weights are updated by using (25). Whether the actor NN is updated or not, the critic error is reinforced at each TTI by following the same reasoning of (23) and (24) and the critic weights are updated based on (25). According to (21), a new decision vector is determined in the learning stage and decoded into a new action $\mathbf{a}[t+2] = [p', d']$ based on (26) and (27).

V. Simulation Results

The proposed *5MART* framework was implemented in a RRM-Scheduler C/C++ object oriented simulator [27] that inherits the LTE-Sim [28] by incorporating new features such as: decoupled time-frequency scheduler based on carrier aggregation, compression techniques for the scheduler states, scheduler controllers (RL algorithms and neural network approximations), etc. To evaluate the performance of the proposed framework, we have used 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. The performance of the *5MART* framework was compared with state-of-the-art schedulers in the same networking conditions.

The aim of this section is threefold: *a)* to identify the simulated scenario and parameter settings for network, scheduler

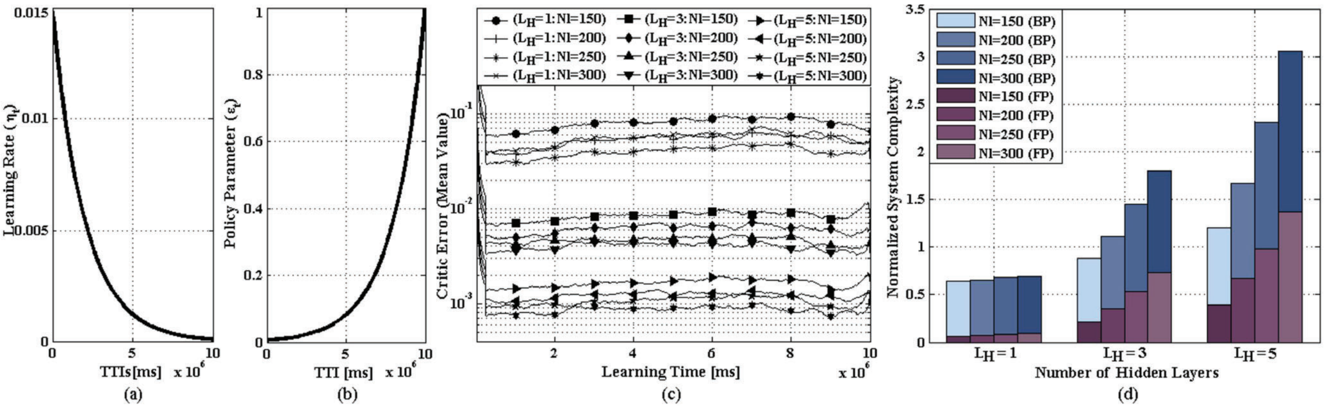


Fig. 4. (a) Learning rate function; (b) Policy parameter function; (c) Critic error (mean value) under different configurations of hidden layers and nodes; (d) Normalized system complexity over the training stage.

Table II Parameter Settings

Parameter	Value/Description
System Bandwidth/Cell Radius	100 MHz (500 RBs), intra-band contiguous CA)/200m
Channel Model	Jakes Model
Path Loss/Penetration Loss	Urban Micro Cell/10dB
Carrier Frequency/DL Power	2GHz/43dBm
Frame Structure/Interfered Cells	FDD/(6 for each CC)
User Speed/Mobility Model	3 Kmph/Random Walk
CQI Reporting Mode	Full-band, periodic at each TTI
PUCCH Model	Errorless
RLC ARQ	Acknowledged Mode (5 retransmissions)
AMC Levels	QPSK, 16-QAM, 64-QAM
Target BLER	10%
Traffic Type	Heterogeneous (20% 360° video, 60% 2D video, 15% VoIP, 5% FTP)
QoS Requirements	$[20Mbps, 10^{-3}, 10ms]$ (360° video) $[1Mbps, 10^{-3}, 150ms]$ (2D video) $[32kbps, 10^{-2}, 50ms]$ (VoIP) $[256kbps, 10^{-6}, 300ms]$ (FTP)
Max. No. of Users	Variable: 12 (360° video), 36 (2D video), 9 (VoIP), 3 (FTP)
Time-Frequency Schedulers	5MART, RADS [10], FLS [11]
Frequency Schedulers	PF [4], BF [7], EXP [5], OPLF [6]
RL Algorithm	Actor-Critic
Learning/Exploitation Duration	10000s/500s
RL Discount Factor (γ)	0.99
NN Configuration (Optimum)	$(L = 2, L_H = 1, N_l = 150)$.

and intelligent controller; *b)* to study different configurations of neural networks and find the best trade-off between the decision accuracy and system complexity; *c)* to compare the performance of *5MART* with RADS, FLS and SP schedulers.

A. Parameters Settings

We consider the OFDMA downlink scenario with a total system bandwidth of 100MHz (500 RBs) consisting of $C = 5$ carrier components with 20MHz bandwidth each. Each carrier is represented by an urban micro-cell with a radius of 200m and FDD transmission mode. We consider the intra-cell interference negligible while the inter-cell interference model considers a cluster with 7 cells for each component carrier. We run the *5MART* framework only on the central cell of each cluster, while other cells provide the inter-cell interference. Jakes fading is used to model the downlink channels. More details on the scenario settings can be found in Table II.

The packet scheduler works on the carrier component basis, which means that separate Radio Link Control entities, retransmission schemes, TB and MCS calculations are considered for each carrier. Each RLC works in the acknowledged mode and considers a maximum of 5 re-transmissions for each erroneous packet. Packets failing to get successfully transmitted within this period are declared lost. When computing the QoS indicators such as PLR and average user throughput, we use a moving time window of 1000 TTIs to collect the lost packets and instantaneous user throughput, respectively. The *5MART* framework takes decisions for both domains:

a) Time Domain Scheduling: we simulate $P = 4$ traffic classes with the following load ratio [2]: 20% 360° video, 60% conventional video, 15% VoIP and 5% FTP traffic. The QoS requirements for each traffic class are presented in Table II [23]. We aim to increase the diversity of user channel conditions to improve the generalization of the learnt feed forward neural networks. Instead of considering each user being characterized by all traffic types, we assume that each user requests only one traffic type all the time. Then, the total number of active users will be multiplied by four. In the learning stage, we randomly switch UEs from idle to active states and vice-versa to create a more dynamic environment, where the maximum ranges are: $U_1 = 12$, $U_2 = 36$, $U_3 = 9$ and $U_4 = 3$. In the exploitation stage, the total number of users is varied based on the predefined traffic load ratio.

b) Frequency Domain Scheduling: at each TTI, one of the following scheduling rules is applied: PF oriented on fairness; OPLF with the main focus on PLR; EXP rule that minimizes the delay; BF that aims to respect the GBR requirements.

For the scheduler controller, the discount factor is $\gamma = 0.99$ to provide nearly the same importance when approximating the value of two consecutive states. In Fig. 4.a, the learning rate η_t is adjusted as the learning stage evolves in the sense that, we learn more at the beginning of the learning stage by setting higher η_t, and we learn less at the end of this stage by decreasing η_t. As depicted in Fig. 4.b, parameter ϵ_t gets lower values at the beginning of the learning stage to increase the number of improvement steps and gets higher as the learning stage progresses to increase the amount of exploitation steps. Since a number of $P = 4$ traffic classes and a pool with $D = 4$ scheduling rules are used, the actor neural network needs 2 pins for each decision, where $N_P = N_D = 2$ and $(p_x = 2, p_y = 0)$. Therefore, the traffic class to be prioritized is decided based on the following decoding principle: *a)*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/tnsm.2019.2960849, IEEE Transactions on Network and Service Management

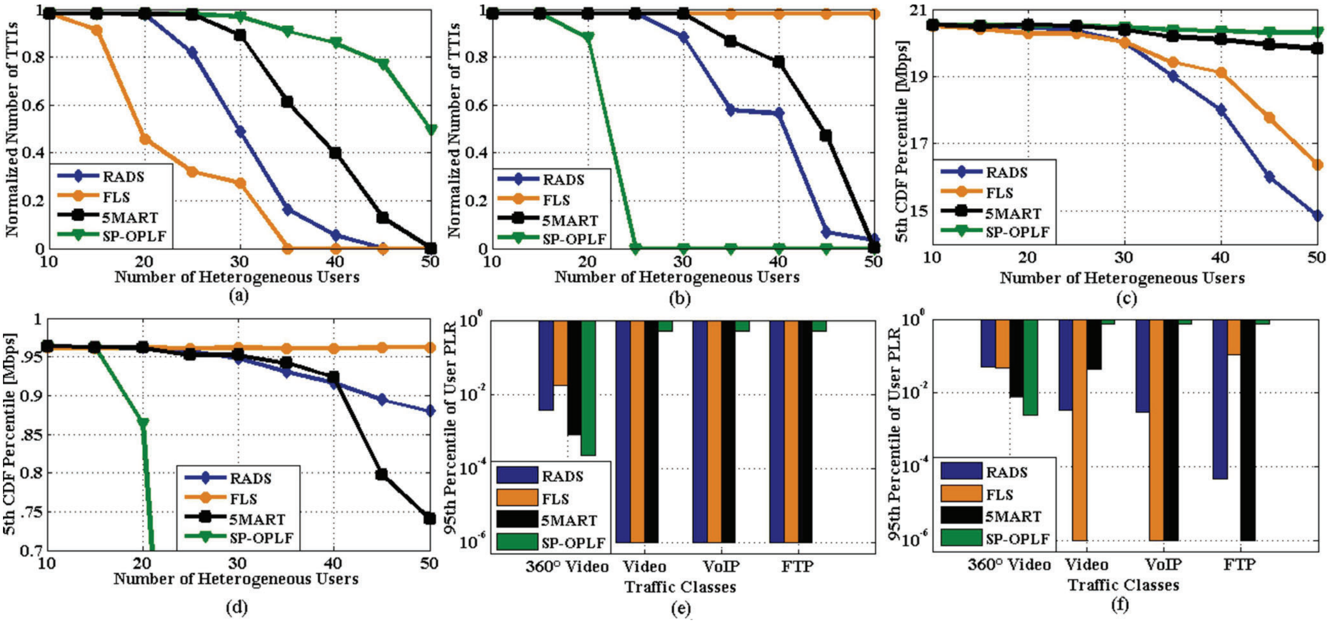


Fig. 5. Normalized number of TTIs when the QoS requirements are met by all users for: (a) 360° video traffic; (b) Conventional video traffic. The 5^{th} percentile of user throughput for: (c) 360° video traffic; (d) Conventional video traffic. Intra-class PLR in terms of mean 95^{th} percentile of packet loss rate in the range of: (e) [10,31] aggregate users; (f) [32,50] aggregate users.

prioritize class $p = 1$ if $(z_1^{(L+1)} \leq 0, z_2^{(L+1)} \leq 0)$; *b)* prioritize class $p = 2$ if $(z_1^{(L+1)} \leq 0, z_2^{(L+1)} > 0)$; *c)* prioritize class $p = 3$ if $(z_1^{(L+1)} > 0, z_2^{(L+1)} \leq 0)$; *d)* prioritize class $p = 4$ if $(z_1^{(L+1)} > 0, z_2^{(L+1)} > 0)$. Similarly, the other output pins $\{z_3^{(L+1)}, z_4^{(L+1)}\}$ are decoded into scheduling rule decision. The rest of the parameter setting is presented in Table II.

B. Learning Stage

Unfortunately, there is no measurable way to determine in advance the internal structure of the neural networks in terms of hidden layers and number of nodes for each hidden layer. The optimal configuration of neural networks differs from one optimization problem to another. In general, when the neural network is too flexible (involving a high number of hidden layers and nodes), the framework complexity is higher and the obtained policy can overfit the input data if the state space is not explored properly. On the other hand, when the configuration is too inflexible (involving a low number of hidden layers and nodes), the framework complexity is lower but the learnt policy can underfit the data and gives poor generalization of the scheduler state space. In both cases, the critic error is continuously increasing given a certain point in the learning stage. To minimize as much as possible the overfitting and underfitting problems, the *5MART* learning framework aims to dynamically change the number of active users within each traffic class at each 1000 TTIs. As it can be observed from Fig. 4.c, the critic errors are generally stable for the following configurations: $L_H = \{1, 3, 5\}$ and $N_l = \{150, 200, 250, 300\}$, where $l = 2, ..., L$. The critic mean error is much lower in the case of $(L_H = 5, N_l = 300)$ due to the increased flexibility of the *5MART* framework to approximate the values of the previous states $\mathbf{z} \in \hat{\mathcal{S}} \times \hat{\mathcal{S}}$ in the current states $\mathbf{z}' \in \hat{\mathcal{S}} \times \hat{\mathcal{S}}$. The actor errors follow the same trend for each considered configuration but at much lower values. This is explainable since lower critic errors involve a higher number of updates for the actor NN and consequently, lower error values than other actor NN configurations.

In this context of learning in parallel different configurations of neural networks, we also measure the accumulated reward in order to get the error impact of different configurations. Unfortunately, the highest reward gain between configuration $(L_H = 5, N_l = 300)$ and $(L_H = 1, N_l = 150)$ is only 0.5%. Thus, we cannot get a much better performance from the network by increasing the neural network topology. This is because, when the network conditions are unfavorable (high number of users, poor channel conditions), the reward is still negative no matter what action is applied. Fig. 4.d plots the normalized system complexity for the entire set of configurations when considering the forward and backward propagation procedures in the learning stages for both critic and actor neural networks. In real-time scheduling systems, it is not recommended to use the following configurations for the *5MART* scheduling framework: $(L_H = 3, N_l = \{200, 250, 300\})$ and $(L_H = 5, N_l = \{150, 200, 250, 300\})$. For these configurations, the learning stage requires a TTI duration larger than 1ms, which is inappropriate even for the LTE systems. However, when performing the offline learning and exploiting the learned non-linear function in real-time scheduling, we recommend the use of the neural network configuration lower than $(L_H = 3, N_l = 200)$. By considering all these aspects, we compare our *5MART* scheduling framework with other state-of-the-art schedulers by taking into consideration a minimal configuration of $(L_H = 1, N_l = 150)$.

C. Exploitation Stage

Figures (5.a)-(5.d) present the simulation results for 360° and conventional video traffic in the exploitation stage. First, we monitor the normalized duration when all QoS requirements are met by all 360° video users under different configurations, as depicted in Fig. 5.a. As expected, the SP scheme

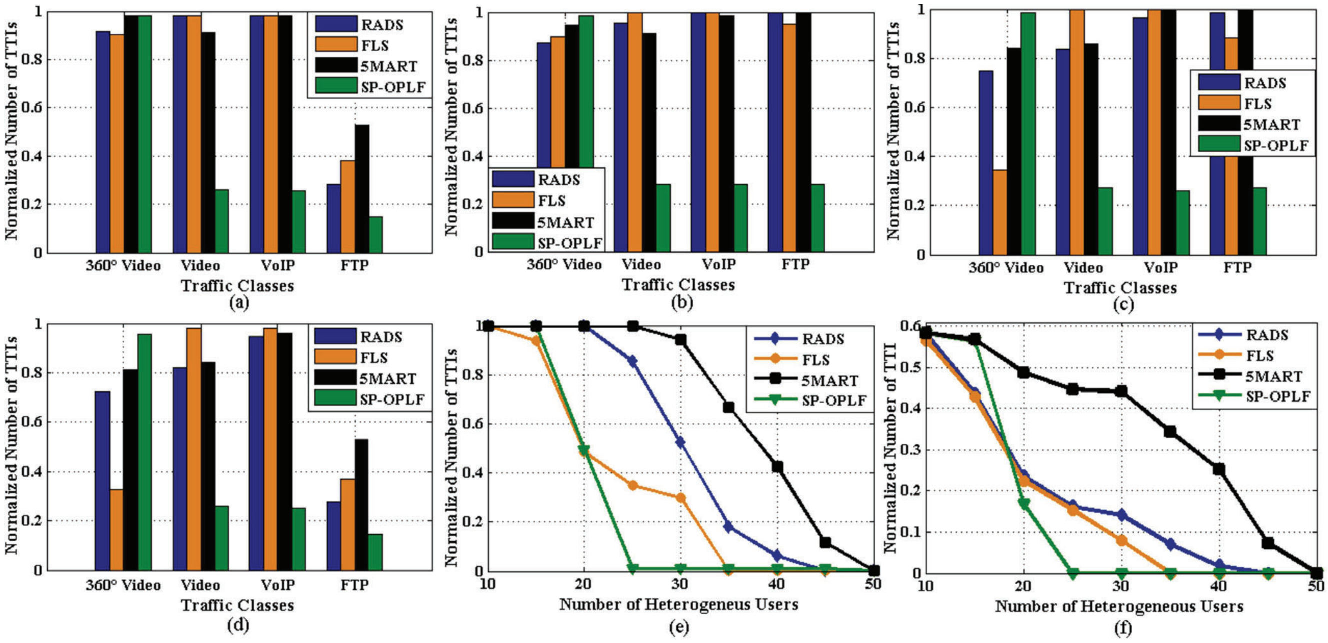


Fig. 6. Intra-class QoS provisioning in terms of: (a) GBR requirements; (b) Delay requirements; (c) PLR requirements; (d) GBR, delay and PLR requirements. Inter-class QoS provisioning in terms of: (e) heterogeneous PLR requirements; (f) heterogeneous GBR, PLR and delay requirements.

that makes use of OPLF scheduling rule in the frequency domain obtains the highest possible outcome in terms of the time period when all QoS objectives are fulfilled. The *5MART* scheduler provides a trade-off between SP-OPLF and FLS/RADS scheduling schemes. Fig. 5.b illustrates the normalized scheduling time when all QoS requirements are met by all users belonging to the conventional video traffic class. In this case, the FLS framework is over-provisioning this second prioritized traffic class while the SP-OPLF scheme is not able to provide video services when the number of heterogeneous users goes above 20. The *5MART* scheduler represents a good trade-off between FLS and other schedulers when the total number of users is $U \geq 30$.

To quantify the gains from Figs. 5.a and 5.b, we represent in Figs. 5.c and 5.d the 5th percentile of the Cumulative Distribution Function (CDF) for the user's mean throughput belonging to 360° and conventional video classes, respectively, over the exploitation stage. In Fig. 5.c, only a slight difference can be noticed in terms of user throughput between *5MART* and SP-OPLF when there are more than 30 aggregate users. However, a degradation higher than 1Mbps can be observed for other two conventional scheduling schemes (RADS and FLS), meaning that, other objectives are also affected such as PLR and packet delay. For conventional video (Fig. 5.d), we notice that *5MART* has a higher degradation when compared to FLS and RADS. This is because the *5MART* scheduler gives more resources to the first prioritized traffic class under a higher number of aggregate users while still maintaining a good trade-off between all traffic classes.

As noticed in Figs. (5.a)-(5.d), for up to 30 heterogeneous users, *5MART* provides near similar results when compared to the best schedulers for each traffic class such as SP-OPLF and FLS, respectively. Above this threshold, *5MART* offers a performance trade-off between 360° and conventional video classes. Fig. 5.e illustrates the intra-class 95th CDF percentile for the user PLR when the number of heterogeneous users belongs to the range of $[10, 31]$. For this scenario, *5MART* respects the PLR requirements for all traffic types meaning that, it can afford a higher number of aggregate users to be served at the same time when compared to other schemes. Fig. 5.f plots the same performance metric when the range of aggregate users is $[32, 50]$. As expected, we observe that *5MART* can accommodate more users. Even though there is a PLR degradation for the conventional video traffic class, *5MART* can still achieve a good trade-off between all traffic classes when compared to the other schemes.

The intra-class provisioning of different QoS objectives for each traffic type is highlighted in Figs. (6.a)-(6.d). For the GBR and delay requirements (see Figs. 6.a and 6.b), *5MART* provides the best trade-off between the traffic classes. When optimizing the PLR and all QoS objectives (Figs. 6.c and 6.d), for 360° video, conventional video and VoIP traffic classes, the *5MART* meta-scheduler is the second best alternative while providing the best performance for the last prioritized class, the FTP traffic. It can be concluded that: FLS is over-provisioning the video and VoIP traffic classes at the expense of degrading its performance for 360° video and FTP traffic types; SP-OPLF is over-provisioning the first prioritized traffic class while harming the others; RADS aims to find a good balance between the four traffic classes; finally, the *5MART* scheduler outperforms other candidates by indicating the best trade-off between the traffic classes in terms of average scheduling time when the per-class QoS requirements are met.

To quantify the trade-off when measuring the intra-class QoS, Figs. 6.e and 6.f analyze the inter-class QoS performance of all traffic classes at the same time. Fig. 6.e indicates that the *5MART* scheduler provides the best performance among the other candidates in terms of the normalized scheduling

$$\begin{aligned}
V^{\boldsymbol{\pi}}(\mathbf{s}) &\overset{(11)}{=} \sum_{p\in\mathcal{P}}\sum_{d\in\mathcal{D}}\Big\{\mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=0}^{\infty}\gamma^{t}\mathcal{R}_{t+1}|\mathbf{s}[0]=\mathbf{s},\mathbf{a}[0]=[p,d]\Big]\cdot\pi[(p,d)\mid\mathbf{s}]\Big\} \overset{(3)}{=} \sum_{p\in\mathcal{P}}\sum_{d\in\mathcal{D}}\Big\{\mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=0}^{\infty}\gamma^{t}\mathcal{R}_{t+1}|\mathbf{c}[1]=f(\mathbf{s},p,d),\\
&\mathbf{s}[0]=\mathbf{s},\mathbf{a}[0]=[p,d]\Big]\cdot\pi[(p,d)\mid\mathbf{s}]\Big\} = \sum_{p\in\mathcal{P}}\sum_{d\in\mathcal{D}}\Big\{\mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=0}^{\infty}\gamma^{t}\mathcal{R}_{t+1}|\mathbf{c}[1]=f(\mathbf{s},p,d),\mathbf{c}[0]=\mathbf{c},\mathbf{v}[0]=\mathbf{v},\mathbf{a}[0]=[p,d]\Big]\cdot\\
&\pi[(p,d)\mid\mathbf{s}]\Big\} \overset{(*)}{=} \sum_{p\in\mathcal{P}}\sum_{d\in\mathcal{D}}\Big\{\mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_{t=0}^{\infty}\gamma^{t}\mathcal{R}_{t+1}|\mathbf{c}[1]=\mathbf{c}'_{(p,d)},\mathbf{c}[0]=\mathbf{c},\mathbf{a}[0]=[p,d]\Big]\cdot\pi[(p,d)\mid(\mathbf{c}'_{(p,d)},\mathbf{c})]\Big\}\\
&= \mathbb{E}_{\boldsymbol{\pi}}\Big[\textstyle\sum_{t=0}^{\infty}\gamma^{t}\mathcal{R}_{t+1}|\mathbf{c}[1]=\mathbf{c}'_{(p,d)},\mathbf{c}[0]=\mathbf{c}\Big] = H^{\boldsymbol{\pi}}(\mathbf{c}'_{(p,d)},\mathbf{c}).
\end{aligned}\tag{29}$$

time when the heterogeneous PLR requirements are met. When compared to *5MART* meta-scheduler and RADS, FLS gets the worst PLR performance. By correlating Fig. 6.e with Figs. 5.e, 5.f, and 6.c, this is explainable since FLS is over-provisioning certain traffic classes while failing to met the PLR requirements of other ones. By considering the normalized fraction of time (in TTIs) when all heterogeneous QoS requirements are met (Fig. 6.f), the *5MART* scheduler provides gains up to 50% when compared to other schedulers for a range of $U \in [20, 40]$ aggregate active users.

D. Considerations of 5MART Framework on 5G Technology

The proposed *5MART* meta-scheduler presents high flexibility and scalability when considering the 5G characteristics, such as: numerology, 5G New Radio (NR), higher frequencies, non-orthogonal access, mmWave communications. As a result of the 5G numerology increase, the time slot duration is much tighter. The *5MART* controller can be learnt to provide the time-frequency decisions for much shorter TTI duration. Alongside the traffic prioritization and scheduling rule selection, the *5MART* controller can also dynamically decide the TTI duration at each iteration based on the networking conditions. With the development of 5G NR, the *5MART* meta-scheduler can be used on any TDD or FDD configuration and frequency band. The 5G NR CQI table does not influence the functionality of *5MART* controller since the consecutive controllable states are perceived, while ignoring the uncontrollable scheduler sub-states, such as CQI reports. For non-orthogonal access schemes, only the frequency-based schedulers must be adapted to support mixed frequency-power allocation. In mmWave communications, the *5MART* meta-scheduler can employ an additional functionality for the beam selection management.

VI. Conclusions

This paper proposes *5MART*, a scheduling framework which maximizes the average scheduling time when heterogeneous QoS requirements are met for diverse traffic classes. To make the *5MART* framework suitable for real-time processes, we use an actor-critic RL framework to learn in each scheduler state about the traffic class to be prioritized in time domain and the scheduling rule to be performed in frequency domain. Also, we employ a neural network to approximate the best scheduling decisions at each TTI. Additionally, an innovative scheme for the scheduler state space compression is provided in order to reduce framework complexity and speed-up learning. Through extensive simulation results, we show that the proposed *5MART* framework provides gains in excess of 50% when compared to other state-of-the-art schedulers when using the neural network approximator with the lowest complexity.

Acknowledgements

The support of the European Union Horizon 2020 Research and Innovation Programme for the NEWTON project (Grant Agreement no. 688503) is gratefully acknowledged.

Appendix A: Proof of Theorem 1

By starting with the reward definition from (4), this equation can be developed as follows:

$$\begin{aligned}
r(\mathbf{s},p,d) &\overset{(4)}{=} \mathbb{E}\Big[\mathcal{R}_{t+1}|\mathbf{s}[t]=\mathbf{s},\mathbf{a}[t]=[p,d]\Big]\\
&\overset{(3)}{=} \mathbb{E}\Big[\mathcal{R}_{t+1}|\mathbf{c}[t+1]=f(\mathbf{s},p,d),\mathbf{s}[t]=\mathbf{s},\mathbf{a}[t]=[p,d]\Big]\\
&= \mathbb{E}\Big[\mathcal{R}_{t+1}|\mathbf{c}[t+1]=\mathbf{c}'_{(p,d)},\mathbf{c}[t]=\mathbf{c},\mathbf{v}[t]=\mathbf{v},\mathbf{a}[t]=[p,d]\Big]\\
&\overset{(*)}{=} \mathbb{E}\Big[\mathcal{R}_{t+1}|\mathbf{c}[t+1]=\mathbf{c}'_{(p,d)},\mathbf{c}[t]=\mathbf{c},\mathbf{a}[t]=[p,d]\Big]\\
&= r(\mathbf{c}'_{(p,d)},\mathbf{c},p,d),
\end{aligned}\tag{28}$$

where (*) indicates that the uncontrollable state $\mathbf{v}[t] \in \mathcal{S}^U$ can be reproduced at TTI t+1, when both controllable states $\{\mathbf{c},\mathbf{c}'\} \in \mathcal{S}^C$ are known. The arrival rate of user $u \in \mathcal{U}_p$ can be determined knowing the elements $\{q_u, q'_u, x_{o_1,p,u}\}$ at TTI t+1. Also, the SINR levels for the allocated RBs can be estimated when knowing the set of user throughput $\{x_{o_1,p,u}, x'_{o_1,p,u}\}$ within TTIs t and t+1, respectively. Consequently, the CQI reports can also be approximated and reproduced for each user $u \in \mathcal{U}_p$ of each traffic class $p \in \mathcal{P}$.

Appendix B: Proof of Theorem 2

Based on initial definition from (11), the state value function can be developed as shown in (29). The state value function is a sum of expectations being given the set of discrete actions $\mathbf{a}[t] = [p,d] \in \mathcal{A}$ [16]. Then, this sum of expectations keeps exactly the same form when applying the transition function from (3). The property $(*)$ has the same meaning as in (28) in which the dependency on uncertainty $\mathbf{v}[t] \in \mathcal{S}^U$ can be eliminated when knowing the controllable set $\{\mathbf{c},\mathbf{c}'_{(p,d)}\}$ at TTI t and t+1, respectively. Finally, the state-value function can be written as a function that depends on the consecutive controllable elements in each state.

References

[1] R. Trestian, I.-S. Comșa, and M. F. Tuysuz, "Seamless Multimedia Delivery within a Heterogeneous Wireless Networks Environment: Are We There Yet?" *in IEEE Communications Surveys and Tutorials*, vol. 20, no. 2, pp. 945 – 977, Second Quarter, 2018.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/tnsm.2019.2960849, IEEE Transactions on Network and Service Management

[2] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016-2021," Cisco, Tech. Rep., San Jose, USA, Feb. 2017, Accessed: Dec. 2018. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.

[3] J. G. Andrews, B. S., W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What Will 5G Be?" *in IEEE Journal on Selected Areas in Communications*, vol. 3, no. 6, pp. 1065 – 1082, 2014.

[4] A. Jalali, R. Padovani, and R. Pankaj, "Data Throughput of CDMA-HDR a High Efficiency-High Data Rate Personal Communication Wireless System," in *2000 IEEE 51st Vehicular Technology Conference Proceedings VTC2000-Spring*, vol. 3, May 2000, pp. 1854 – 1858.

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

[6] N. Khan, M. Martini, Z. Bharucha, and G. Auer, "Opportunistic Packet Loss Fair Scheduling for Delay-Sensitive Applications over LTE Systems," in *IEEE Wireless Communications and Networking Conference*, vol. 1, April 2012, pp. 1456 – 1461.

[7] M. Lundevall, B. Olin, J. Olsson, N. Wiberg, S. Wanstedt, J. Eriksson, and F. Eng, "Streaming Applications Over HSDPA in Mixed Service scenarios," in *IEEE Vehicular Technology Conference VTC2004-Fall*, vol. 1, April 2005, pp. 841 – 845.

[8] B. Bojovic and N. Baldo, "A New Channel and QoS Aware Scheduler to Enhance the Capacity of Voice over LTE Systems," in *IEEE 11th International Multi-Conference on Systems, Signals and Devices (SSD14)*, February 2014, pp. 1 – 6.

[9] F. Avocanh, M. Abdennebi, and J. Ben-Othman, "An Enhanced Two Level Scheduler to Increase Multimedia Services Performance in LTE Networks," in *IEEE International Conference on Communications (ICC)*, June 2014, pp. 2351 – 2356.

[10] G. Monghal, D. Laselva, P.-H. Michaelsen, and J. Wigard, "Dynamic Packet Scheduling for Traffic Mixes of Best Effort and VoIP Users in E-UTRAN Downlink," in *IEEE Vehicular Technology Conference (VTC-Spring)*, May 2010, pp. 1 – 5.

[11] G. Piro, L. Grieco, G. Boggia, R. Fortuna, and P. Camarda, "Two-Level Downlink Scheduling for Real-Time Multimedia Services in LTE Networks," *in IEEE Transactions on Multimedia*, vol. 13, pp. 1052 – 1065, 2011.

[12] K. Wang, X. Li, H. Ji, and X. Zhang, "Heterogeneous Traffic Scheduling in Downlink High Speed Railway LTE Systems," in *IEEE Global Communications Conference (GLOBECOM)*, December 2013, pp. 1452 – 1457.

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

[14] C. Desogus, M. Anedda, M. Murroni, and G.-M. Muntean, "A Traffic Type-Based Differentiated Reputation Algorithm for Radio Allocation During Multi-Service Content Delivery in 5G Heterogeneous Scenarios," *in IEEE Access*, vol. 7, pp. 27 720 – 27 735, 2019.

[15] O. Grndalen, A. Zanella, K. Mahmood, M. Carpin, J. Rasool, and O. N. Sterb, "Scheduling Policies in Time and Frequency Domains for LTE Downlink Channel: A Performance Comparison," *in IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3345 – 3360, April 2017.

[16] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. IMT Press Cambridge, 2017.

[17] I.-S. Comşa, S. Zhang, M. Aydin, J. Chen, P. Kuonen, and J.-F. Wagen, "Adaptive Proportional Fair Parameterization Based LTE Scheduling Using Continuous Actor-Critic Reinforcement Learning," in *IEEE Global Communications Conference (GLOBECOM)*, December 2014, pp. 4387–4393.

[18] I.-S. Comşa, A. De Domenico, and D. Ktenas, "QoS-Driven Scheduling in 5G Radio Access Networks - A Reinforcement Learning Approach," in *IEEE Global Communications Conference (GLOBECOM)*, December 2017, pp. 1–7.

[19] I.-S. Comşa, S. Zhang, M. Aydin, P. Kuonen, L. Yao, R. Trestian, and G. Ghinea, "Towards 5G: A Reinforcement Learning-based Scheduling Solution for Data Traffic Management," *in IEEE Transactions on Network and Service Management*, vol. 1, no. 1, pp. 1 – 15, August 2018.

[20] I. S. Comşa, A. De Domenico, and D. Ktenas, "Method for Allocating Transmission Resources Using Reinforcement Learning," U.S. Patent US20 190 124 667A1, April 25, 2019.

[21] N. Pratas and N. H. Mahmood, "D4.1: Technical Results for Service Specific Multi-Node/MultiAntenna Solutions," Aalborg University, Tech. Rep., 2016. [Online]. Available: http://fantastic5g.com/wp-content/uploads/2016/06/FANTASTIC-5G˙D4.1˙Final.pdf

[22] H. Van Hasselt and M. Wiering, "Using Continuous Action Spaces to Solve Discrete Problems," in *International Joint Conference on Neural Networks*, April 2009, pp. 1149 – 1156.

[23] 3GPP, "Technical Specification Group Services and System Aspects; Policy and charging control architecture Release 12, v.12.2.0," 3GPP, Tech. Rep., 2013.

[24] G. Song and Y. Li, "Utility-based Resource Allocation and Scheduling in OFDM-based Wireless Broadband Networks," *in IEEE Communications Magazine*, vol. 43, no. 12, pp. 127 – 134, 2005.

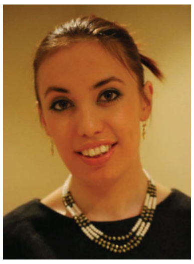
[25] C. Szepesvri, *Algorithms for Reinforcement Learning: Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan and Claypool Publishers, 2010.

[26] H. P. van Hasselt, *Insights in Reinforcement Learning Formal Analysis and Empirical Evaluation of Temporal-Difference Learning Algorithms*. University of Utrecht, 2011.

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

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

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



Ramona Trestian (S'08-M'12) is a Senior Lecturer with the Design Engineering and Mathematics Dept., Middlesex Univ., UK. She received her Ph.D. degree from Dublin City Univ., Ireland in 2012. She was with Dublin City Univ. as an IBM/IRCSET Exascale Postdoctoral Researcher. She published in prestigious international conferences and journals and has five edited books. Her research interests include mobile and wireless communications, quality of experience, multimedia streaming, SDN, handover and network selection strategies.

Gabriel-Miro Muntean (M'04–SM'17) is an Associate Professor with the School of Electronic Engineering, Dublin City Univ. (DCU), Ireland, and co-Director of the DCU Performance Engineering Laboratory. He has published over 300 papers in top-level international journals and conferences, authored three books and 18 book chapters, and edited seven books. His research interests include quality, performance, and energy issues related to rich media delivery, technology-enhanced learning, and other data communications over heterogeneous networks. He is an Associate Editor of the IEEE Transactions on Broadcasting, the Multimedia Communications Area Editor of the IEEE Communications Surveys and Tutorials, and a reviewer for important international journals, conferences, and funding agencies. He is the coordinator of the EU-funded project NEWTON http://www.newtonproject.eu.

Gheorghiţă Ghinea is a Professor in the Computer Science Department at Brunel Univ., UK. He received the B.Sc. and B.Sc. (Hons) degrees in Computer Science and Mathematics, in 1993 and 1994, respectively, and the M.Sc. degree in Computer Science, in 1996, from the Univ. of the Witwatersrand, Johannesburg, South Africa; he then received the Ph.D. degree in Computer Science from the Univ. of Reading, UK, in 2000. His work focuses on building adaptable cross-layer end-to-end communication systems incorporating user perceptual requirements. He is a member of the IEEE and the British Computer Society.