IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 14, 2020, accepted May 30, 2020, date of publication June 8, 2020, date of current version June 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3000893

Learn to Schedule (LEASCH): A Deep Reinforcement Learning Approach for Radio Resource Scheduling in the 5G MAC Layer

FAROQ AL-TAM[1], NOÉLIA CORREIA[1], AND JONATHAN RODRIGUEZ[2], (Senior Member, IEEE)

[1]Center for Electronics, Optoelectronics, and Telecommunications (CEOT), Faculty of Science and Technology, University of Algarve, 8005-139 Faro, Portugal
[2]Institute of Telecommunications, University of Aveiro, 3810-193 Aveiro, Portugal

Corresponding author: Faroq Al-Tam (ftam@ualg.pt)

This work was supported in part by the European Regional Development Fund (FEDER), through the Competitiveness and Internationalization Operational Programme (COMPETE 2020), in part by the Fundação para a ciência e Tecnologia Regional, through the Operational Program of the Algarve (2020), in part by i-Five: Extensão do acesso de espectro dinâmico para rádio 5G under Grant POCI-01-0145-FEDER-030500, and in part by the Fundação para a ciência e Tecnologia, Portugal, within the Center for Electronics, Optoelectronics, and Telecommunications (CEOT), under Grant UID/MULTI/00631/2020.

ABSTRACT Network management tools are usually inherited from one generation to another. This was successful since these tools have been kept in check and updated regularly to fit new networking goals and service requirements. Unfortunately, new networking services will render this approach obsolete and handcrafting new tools or upgrading the current ones may lead to complicated systems that will be difficult to maintain and improve. Fortunately, recent advances in AI have provided new promising tools that can help solving many network management problems. Following this interesting trend, the current article presents LEASCH, a deep reinforcement learning model able to solve the radio resource scheduling problem in the MAC layer of 5G networks. LEASCH is developed and trained in a sand-box and then deployed in a 5G network. It has been evaluated under different numerology settings. The experimental results show that it is both numerology-agnostic and efficient when compared to conventional baseline methods in many key performance indicators.

INDEX TERMS 5G, MAC, deep reinforcement learning, scheduling, resource management.

I. INTRODUCTION

The rapid evolution of networking applications will continue to bring new challenges to communication technologies. In the fourth-generation (4G), also known as long term evolution (LTE), throughput and delay were the main foci. In 5G and beyond, services have reached completely new levels. This new era of communication is featured by new killer applications that will benefit from emergent technologies like Internet of things (IoT) and next generation media such as virtual reality (VR) and augmented reality (AR), to name a few.

Unlike LTE, 5G is a use-case driven technology. In addition, 5G is not only machine-centric but also user-centric, where the user notion has evolved to cover a wider range of entities other than a traditional human-on-handset notion. Small devices that use 5G infrastructure are basically clients/users [1].

The main use cases supported by 5G, for now, are enhanced mobile broadband (eMBB), ultra-reliable and low latency communications (URLLC) and massive machine-type communications (mMTC). eMMB supports high capacity and high mobility (up to 500 km/h) radio access with 4 ms user plane latency. URLCC provides urgent and reliable data exchange with sub 1 ms user plane latency. The new radio (NR) of 5G will also support massive number of small packet transmissions for mMTC with sub 10 ms latency. Furthermore, it is foreseen that more requirements will appear in the future, in particular with regard to energy [2], [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wang.

The main key-enablers to handle the requirements of this new era include flexible numerology, bandwidth parts (BWPs), service multiplexing and mini-slotting, optimized frame structure, massive MIMO, inter-networking between high and low bands, and ultra lean transmission [1], [6], [7]. In addition, emergent technologies like software-defined networking (SDN) [5], network function virtualization (NFV), and network slicing [4] will also be key technologies in paving the way for enhancements in 5G and beyond.

LTE and 5G both rely on the same multi-carrier modulation (OFDM) waveform. Nevertheless, the NR is more flexible. It supports a multi-numerology structure having different sub-carrier spacings (SCS), symbol durations and cyclic prefixes (CPs). This flexibility, on one hand, makes it possible to deliver data for all three main use-cases but, on the other hand, makes it difficult to manage resources efficiently. In addition, it is expected that more use cases will emerge and it is foreseen more flexibility into the NR frame in the future, making the resource management task even more complicated. For instance, current specifications of the physical layer supports only four BWPs for each user with only one BWP being active at a time. However, UEs in the future will be able to use multiple BWPs simultaneously [8].

The new service requirements [9], the significant diversity of the characteristics of the traffic [10], and the user stringent requirements, make 5G a complex system that can not be completely managed by tools inherited from ancestor networks [9]. Current radio resource scheduling tools, for instance, are designed to follow a single policy all the time. However, the 5G system will follow various policies to adapt to network configuration and traffic dynamics. On the other hand, artificial intelligence (AI) can provide model-based and model-free approaches that can learn (or select) the appropriate policy under current network conditions [32]. One of the main paths is to rely on new AI advancements like deep reinforcement learning (DRL). This path is featured by a new concept, learn-rather-than-design (LRTD), and it branches into two main research subpaths. One focuses on using AI to select an algorithm (or policy), among candidate policies, according to current network state. The other subpath focuses on developing AI-models that learn policies and apply them according to network conditions. These two subpaths are discussed further in Section III.

The current article is aligned with the second subpath. It focuses on a fundamental problem in 5G: the radio resource management (RRM) problem. In general, RRM can be seen as a large problem with many tasks. This article specifically studies the radio resource scheduling (RRS) task in the media access control (MAC) layer. The main contributions of this work are:

- A numerology-agnostic DRL model. The proposed model works under different numerologies with no modification to its architecture or retraining, unlike state of the art models requiring a different architecture whenever numerology changes [17].
- A clear pipeline for the development/training of DRL agents and their deployment into network simulators;
- A comparative analysis in several network settings between the proposed model and the baseline algorithms;
- A reward analysis of the model to inspect which policy the model has learned, which is rarely performed in the literature.

Our approach is novel compared to AI-based approaches which are still scarce. First, this work proposes off-simulator training scheme, which maximizes the flexibility of training the agents and minimizes the training time. In addition, deploying our model is as easy as deploying any other conventional scheduler. Second, our model is tested on an environment different from the one being trained in. From a generalization point of view, we think this should be the case for DRL agents. That is, the training and deployment tasks should be separated to suppress any dependency. Third, the designed model is new and includes novel state and reward components. Finally, our work is tested on a 5G system level simulator that uses all recent components and configurations of a 5G network. Up to our humble knowledge, all these components have not been jointly addressed in any previous work.

This article is organized as follows: The RRS problem and the system model are described in the reminder of this section. Section II presents a brief background about DRL theory. The related work is presented later in Section III. Sections IV and V present the proposed approach and the results, respectively. The article is then concluded in Section VI.

A. RADIO RESOURCE SCHEDULING PROBLEM

The continuous update of physical layers to handle new use cases in communications is the main surge behind the development of flexible MAC layers or components thereof. As new use cases emerge, handcrafted MAC layers become more complicated and prone to error. This is, in fact, one of the main problems in modern networks and resource management [11], [12]. Human-centered approaches lack flexibility and usually require continuous repairs and updates, which leads to a degradation in the level of service and compromise in the performance [13].

Improving the ability of communication systems to effectively share the available scarce spectrum among multiple users, has always been one of the main research targets of academia and big com industry. As the service stack continues to grow, more user requirements will be added to the system. The need to find better resource sharing approaches becomes inevitable. Therefore, RRS is an essential task in communications. The main objective of RRS task is to dynamically allocate spectrum resources to UEs while achieving an optimal trade-off between some key performance indicators (KPIs), like spectrum efficiency (SE), fairness, delay, and so on [14]. Achieving such trade-off is known to be a combinatorial problem.

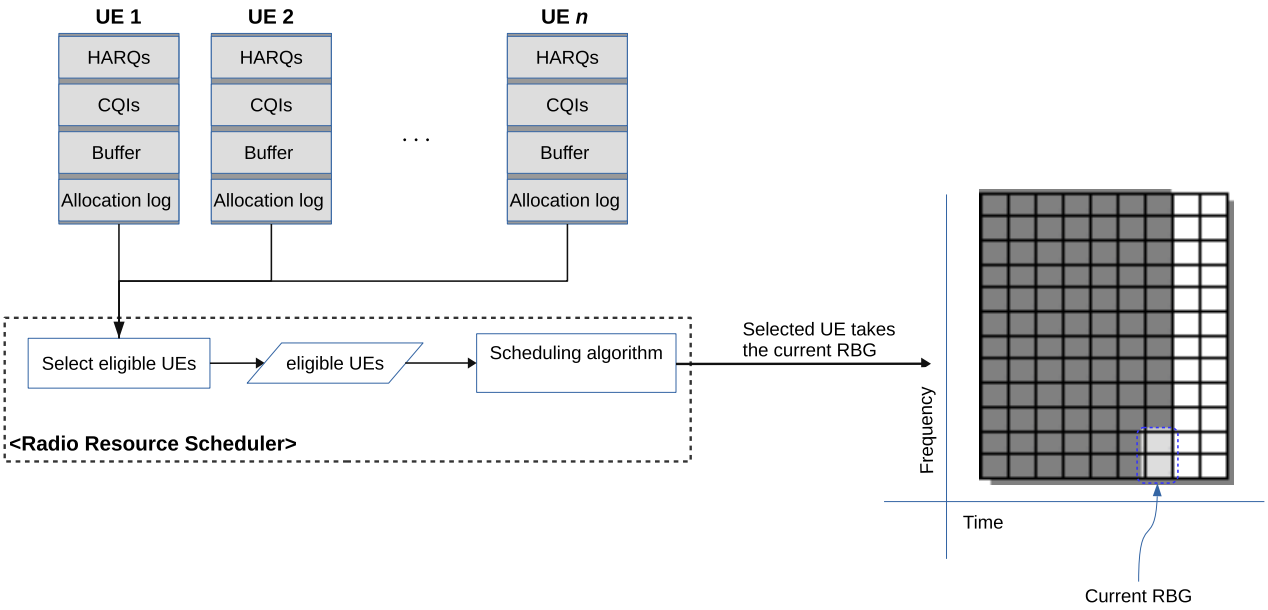


FIGURE 1. Radio resource scheduler.

TABLE 1. Numerology settings in 5G NR.

Index	SCS (kHz)	# slots	slot duration (ms)	RB bandwidth (kHz)
0	15	1	1	180
1	30	2	0.5	360
2	60	4	0.25	720
3	120	8	0.125	1440
4	240	16	0.0625	2880

B. SYSTEM MODEL

The available bandwidth in the frequency domain is divided into resource blocks (RBs). Each resource block is 12 subcarriers. A set of RBs can be aggregated to form a resource block group (RBG); see [15] for possible RBG sizes. An RBG will be the smallest scheduling unit.

The time domain is divided into frames, each frame is divided further into 10 subframes. The number of slots in each subframe, the duration of the slot, and the bandwidth of RB depend on the numerology index in use; see Table 1.

As shown in Figure 1, a set of UEs in the system compete for the available resources. The gNB is able to collect information from them that includes channel feedback information, buffer, HARQs, and allocation log. Additional information can also be obtained. The radio resource scheduler runs at the gNB at every (or kth) slot and uses this information to share the available RBGs between active UEs. Therefore, the problem boils down to filling the resource grid by deciding which UE will win the current RBG in the current slot. However, not all users can be considered for scheduling at the current RBGs. Only those that are eligible (active) will be considered and allowed to compete for the RBGs under consideration. A UE is eligible if it has data in the buffer and is not retransmitting in the current slot, i.e., if it is not associated with a HARQ process in progress.

C. WHY DRL IS SUITABLE FOR RRM PROBLEM?

In many cases, obtaining an optimal solution for the RRS problem is computationally prohibitive due to the size of the state-space and the partial observability of the states [16]. Moreover, surged by new requirements, the RRS task will continue to expand, in the future, both horizontally and vertically. Horizontally, regarding the number and diversity of users, and traffic patterns it should support, and vertically by having to consider new (and perhaps contradictory) KPIs. Therefore, RRS can easily become intractable even for small-scale scenarios.

Current RRS solutions are driven by conventional off-the-shelf designated tools. This includes variants of the proportional fairness (PF), round robin (RR), BestCQI, among others. This scheme has been successful but it will become difficult to maintain it in the future. A new RRS approach is inevitable due to:

- the rapid increase in network size;
- the breadth of control decisions space;
- the new perception from business-makers and end-users of the networking services and applications;
- modern networks are delayed return environments;
- lack of sufficient understanding of underlying network by conventional tools, i.e., they are myopic.

In this context, we share the same vision with [11] that research communities and industry have been focusing on developing services, protocols, and applications more than developing efficient management techniques. Fortunately, recent technologies in AI offer promising solutions and there is a consensus among many scholars of the need for AI-powered network management [18].

The notion of *self-driving* networks [11] is gaining more and more attention nowadays. The core vision of self-driving network engineering is to learn rather than to design the

network management solutions [12]. Such vision has radically changed some fields like computer vision via deep learning [25], by learning features rather than hand-crafting them. However, we did not witness such major progress in network management. The reason is that supervised learning is not suitable for some control and decision problems, since collecting and labeling networking data is not trivial, and network states are non-stationary [26], [27]. DRL, contrarily to supervised learning, can be quite suitable for such problems due to the following reasons:

- All information about RRM can be centralized in the gNB thus creating a network wide view (although not fully) of the network. In addition, new paradigms like knowledge defined networking (KDN) can be used [28];
- DRL agents can continue learning and improving while the network operates. They can interact with other conventional components in the system, and learn from them if necessary [29];
- Network dynamics are difficult to anticipate and exact mathematical models are not scalable. For 5G network management, it is difficult to model the network state and traffic due to the diversity of the applications and traffic it supports [10]. Therefore, DRL model-free agents can be the choice.
- After the new breakthroughs, DRL became an extremely hot research topic [30]. In networking, the popularity of DRL is increasing and some famous network simulators have been recently extended to support general DRL environments like gym [31].

II. DEEP REINFORCEMENT LEARNING

RL is a learning scheme for sequential decision problems and the goal is to maximize a cumulative future reward. An environment of such scheme can be modeled as a Markov decision process (MDP) represented by the tuple $(\mathcal{S}, \mathcal{A}, \boldsymbol{P}, r, \gamma)$. Where $\mathcal{S}$ is a compact space of states of the environment. $\mathcal{A}$ is a finite set of possible actions (action space), $\boldsymbol{P}$ is a predefined transition probability matrix such that each element $p_{ss'}$ determines the probability of transition from state s to s'. The reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ tells how much reward the agent will get when moving from state s to state s' due to taking action a. The γ is a discount factor used to trade-off the importance between immediate and long-term rewards.

In RL, an agent learns a policy π by interacting with environment. In each time step t, the agent observes a state s_t, takes an action (decision) a_t, observes a new state s_{t+1} and receives a reward signal $r(s_t, a_t)$. The learning scheme can be episodic or non-episodic and some states are terminal.

A policy π is a behavioral description of the agent and the policy for state s, $\pi(s)$, can be defined as a probability distribution over the action space $\mathcal{A}$, such that the policy for the pair (s, a), $\pi(a|s)$, defines the probability assigned to a in state s. Therefore, a policy simply tells us which action to take at state s.

The objective of training an agent is to find an optimal policy that will tell the agent which action to take when in a specific state. Therefore, the objective of an agent boils down to maximizing the expected reward for a long run. Starting from state s_t, the outcome (return) can be expressed as:

$$G_t = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}) | s_0 = s_t\right] \tag{1}$$

For a non-episodic learning scheme, we can see that $\gamma < 1$ is important not only to obtain a trade-off between immediate and long-term rewards but also for mathematical convenience.

When an agent arrives at a state it needs to know how good it is to be at state s and following the optimal policy afterwards. A function to measure that is called the value, aka state-value, function $V(s)$:

$$V(s) = \mathbb{E}\left[G_t | s_t = s\right] \tag{2}$$

Similarly, to measure how good it is to be at state s and take action a, a quality function Q, aka action-value function, can also be derived as:

$$Q(s, a) = \mathbb{E}\left[G_t | s_t = s, a_t = a\right] \tag{3}$$

Once we know Q and π we can calculate V using:

$$V(s) = \sum_{a \in \mathcal{A}} \pi(a|s) Q(s, a) \tag{4}$$

Therefore, V and Q can be related by:

$$V(s) = \mathbb{E}_{a \sim \pi(a|s)}[Q(s, a)]. \tag{5}$$

In addition, these two functions can also be related via an advantage function A [19]:

$$A(s, a) = Q(s, a) - V(s), \tag{6}$$

where A subtracts the value function from the quality function to obtain a relative importance of each action, and tell the agent if choosing an action a is better than the average performance of the policy.

In fact, we are interested in finding Q since we can easily derive the optimal policy π^* from the optimal Q^*. $Q(s, a)$ maps each (s, a) pair to a value, i.e., it measures how good it is to take action a when in state s and then following the optimal policy. Using the Bellman expectation function, we can rewrite $Q(s, a)$ as:

$$Q(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p_{ss'}(a) V(s') \tag{7}$$

Therefore, following the Bellman optimality equation for Q^* we have:

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p_{ss'}(a) \max_{a'} Q^*(s', a') \tag{8}$$

The optimal policy can be then derived from the optimal values $Q^*(s, a)$ by choosing the maximum action value in each state. This scheme is known as value-based (compared

to policy-based) learning since the policy is driven from the value function:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s, a), \quad \forall s \in \mathcal{S} \tag{9}$$

However, finding π^* is not easy since in may real world applications, the transition probability is not known. One algorithm to solve this Bellman optimality equation is the Q-learning algorithm [20]. This algorithm is off-policy critic-only (compared to on-policy and actor-critic algorithms). In this algorithm, Q is represented as a lookup table, which can be initialized by random guesses and gets updated in each iteration using the Bellman Equation:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \tag{10}$$

For terminal state this update comes down to:

$$Q(s_t, a_t) = r(s_t, a_t) \tag{11}$$

In order to balance between exploration and expedition, the agent, in Q-learning, adapts an ϵ-greedy algorithm. In ϵ-greedy, the agent selects an action a using $a = \arg\max_{a' \in \mathcal{A}} Q(s, a')$ with probability $1 - \epsilon$, otherwise selects a random action with probability ϵ. This randomness in decision making helps the agent to avoid local minimums. As the agent progresses in learning, it reduces ϵ via a decaying threshold δ_ϵ. With this annealing property of ϵ-greedy, in practice, an agent is expected to perform almost randomly in the beginning and matures with time.

One drawback of the original Q-learning algorithm is scalability. Keeping a tabular for such iterative update is feasible only for small problems. For larger problems, it is infeasible to keep track of each (s, a) pairs. Therefore, in practice it is more feasible to approximate Q.

A common way to approximate Q is to use a deep neural network (DNN). This cross-breeding between deep learning and Q-learning has yielded deep Q networks (DQN), more generally known as deep reinforcement learning (DRL), which is the main breakthrough behind recent advancements in RL that delivered a human-level performance in Atari games [21] and even more strategic games [22] where the agent learns directly from a sequence of image frames via convolutional neural networks (CNN) and DRL.

In DQN, the Q function is approximated by minimizing the squared error between the Bellman equation and the neural network estimation, aka mean-squared Bellman error (MSBE):

$$\text{loss} = \left(Q(s_t, a_t; \boldsymbol{\theta}) - Q^{\text{target}}\right)^2 \tag{12}$$

where Q^{target} is the target Q function, known as the target critic, and $\boldsymbol{\theta}$ is the set of DNN parameters. Q^{target} is calculated as:

$$Q^{\text{target}} = r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \boldsymbol{\theta}) \tag{13}$$

where $\boldsymbol{\theta}$ is the set of DNN's weights and is updated in a stochastic gradient descent (SGD) fashion. For a predefined learning rate α and a mini-batch size M, $\boldsymbol{\theta}_t$ is updated using:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_t + \frac{\alpha}{M} \left(Q(s, a; \boldsymbol{\theta}_t) - Q^{\text{target}}(\boldsymbol{\theta}_t)\right) \nabla_{\boldsymbol{\theta}_t} Q(s, a; \boldsymbol{\theta}_t) \tag{14}$$

where $\left(Q(s, a; \boldsymbol{\theta}_t) - Q^{\text{target}}(\boldsymbol{\theta}_t)\right)$ is known as the temporal difference (TD) error.

In DQN, the state and actions are represented by two separate networks and combined via an `Add` layer. The output is a single value (Q value) in a way similar to classical regression. However, a more efficient architecture is to have the state as input and let the network output be equal to the length of action space. This way, each output represents the likelihood of an action given the state. As in classical Q-learning, the action with maximum likelihood will be selected.

In order to stabilize the results, and to break any dependency between sequential states, DQN uses two tricks. First, two identical neural networks are used one for on-line learning and another to calculate the target Q^{target}. The target network is updated periodically, from the on-line network, every T steps. Therefore, the target is calculated from a more mature network, thus increasing the learning stability:

$$Q^{\text{target}} = r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \hat{\boldsymbol{\theta}}) \tag{15}$$

where $\hat{\boldsymbol{\theta}}$ is a delayed version of $\boldsymbol{\theta}$

Instead of copying the weights from the on-line to the target network at every T steps, it turns out that a smoothing (i.e., progressive) update approach can noticeably increase the learning stability:

$$\hat{\boldsymbol{\theta}} = \beta \boldsymbol{\theta} + (1 - \beta)\hat{\boldsymbol{\theta}} \tag{16}$$

where β is a small real-valued smoothing parameter.

The second trick is to use an experience replay memory $\mathcal{R}$, usually implemented as a cyclic queue. This memory is updated in every learning step, by appending the tuple (s, a, s_{t+1}, r_{t+1}) to the end of the queue. Therefore, when training Q, random mini-batches are sampled from $\mathcal{R}$ and fed to the Q on-line network. $\mathcal{R}$ reduces the dependency between consequence input and improve the data efficiency via re-utilizing the experience samples.

Q-learning and its variant DQN tend to be overoptimistic due to the noise in the Q estimates, and the use of the max operator in selecting the action and calculating the value of the action. A solution is the Double DQN (DDQN) [23], [24] model, which learns from two different samples. One is used to select the action and another one is used to calculate the action value. Therefore, in DDQN, the critic target Q^{target} is calculated as:

$$Q^{\text{target}} = r_{t+1}(s, a) + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a, \boldsymbol{\theta}); \hat{\boldsymbol{\theta}}) \tag{17}$$

In expression (17), the selection of the action is made from the on-line network, i.e., $\arg\max_a Q(s_{t+1}, a, \boldsymbol{\theta})$, and the evaluation and update is made from the target critic network $Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a, \boldsymbol{\theta}); \hat{\boldsymbol{\theta}})$.

III. RELATED WORK

DRL solutions for RRS are scarce in the literature but they can be divided according to the nature of the action space into two main categories: Coarse (high-level) and fine-grained (low-level) decisions. In the former, the DRL agent acts as a method/algorithm selector [17], [32] or protocol designer [12], [13]. For instance, for a given network state, the DRL agent selects which conventional algorithm is suitable to perform scheduling. In the latter, DRL decisions are hard-wired in the networking fabric. The DRL agent makes fine-grained decisions like filling the resource grid [29], airtime sharing between users and decide which user has rights to access the channel [16], [33], or select which coding scheme is suitable [34]. In addition, the fine-grained methods can be classified into distributed [16], [35] and centralized [29], [33]. In the distributed approaches, each UE acts as a DRL agent. This way the network is composed of multi-agents in a way similar to those in game theory. Such approach is scalable but sharing the network state among multiple entities makes it difficult to guarantee convergence. On the other hand, centralized approaches can benefit from a better computational power and better network state understanding.

Both coarse and fine-grained approaches have pros and cons. The coarse level scheme is more scalable, since the agent acts in almost a constant action space. On the other hand, such approach falls short in obtaining deep control of the network. Conventional algorithms are still the main working horses. In fine-grained approaches, the DRL agent deals with the finest decisions. Therefore, it can obtain a deep control of the network. However, these approaches require more sophisticated designs to be adaptive to networking dynamics. Our work belongs to the fine-grained centralized approaches.

A. COARSE APPROACHES

An algorithm selector approach can be found in [32]. At each slot, an actor-critic agent chooses a scheduling algorithm, among a set of available PF-variants algorithms, to maximize some QoS objectives. The state is the number of active users, the arrival rate, the CQIs, and the performance indicator with respect to the user requirements. The reward function measures the impact of choosing a rule on the QoS satisfaction of the users. A similar approach can be found in [17] for 5G networks but using a variant of actor-critic DRLs known as deep deterministic policy-gradient (DDPG) algorithm and with larger action space that controls more parameters. However, this approach is not numerology-agnostic. In [17], for instance, a distinct DRL design is required for each network setting.

In [13], AlphaMac is proposed which is a MAC designer framework that selects the basic building blocks to create a MAC protocol using a constructive design scheme. A building block is included in the protocol if its corresponding element in the state is 1, zero otherwise. As action, the agent chooses the next state that will increase the reward (which is the average throughput of the channel). Each selection by the agent is then simulated in an event-driven simulator that mimics the MAC protocol but with flexibility to allow adding and removing individual blocks of the protocol.

Physical layer self-driving radio is proposed in [12]. The user specifies the control knobs, and other requirements, and the system learns an algorithm that fits a predefined objective (reward) function. The action space is the control knobs and their possible settings. The system then holds a set of DNN and applies the appropriate one to the input scenario. In fact this work can be regarded as hybrid since it combines both coarse and fine-grained approaches in a hierarchal design.

B. FINE-GRAINED APPROACHES

A general resource management problem is handled in [36] by a policy gradient DRL agent. The objective is to schedule a set of jobs at a resource cluster at a given time step. On one hand, this work demonstrated the suitability of DRL agents, but on the other hand it can not be applied directly to 5G RRS problems.

A RRS agent for LTE networks can be found in [29]. A single RBG is considered and the authors have shown that DRL agent, trained by the DDPG algorithm, can achieve near PF results when it uses PF algorithm as an expert (guide) to learn from. This approach can ensure great stability since the agent learns from a well-established algorithm, but it diminishes the ability of agents to discover their own policies.

In [27] a high volume flexible time (HVFT) traffic driven by IoT is scheduled on radio network via a variant of DDPG algorithm, where the scheduler determines the fraction of IoT traffic on top of conventional traffic. To empower the agent with time notion, a temporal features extractor is used, and these features are then fed to the agent. The reward function is a linear combination of several KPIs, like IoT traffic served, traffic loss due to the introduction of IoT traffic and the amount of served bytes below as the system-wide desired limit.

In [33] a policy gradient DRL is proposed to manage the resource access between LTE-LAA small base stations (SBS) and Wi-Fi access points. The goal is to determine the channel selection, carrier aggregation, and fractional spectrum access for SBS while considering airtime fairness between SBS and WI-FI APs. The state includes all network nodes states, and the reward is the total throughput over the selected channels. The scheduling problem is modeled as a non-cooperative Homo Equalis game model where, in this model, the achievement of a player is calculated by its performance while maintaining a certain fair equilibrium regarding other players. To solve this model and establish a mixed strategy, a deep learning approach is developed, where LSTM and MLP networks are used to encode the input data (from IBM Watson Wi-Fi data set) and the objective function of the model is solved via a REINFORCE-like algorithm. The work has shown throughput improvement when compared to

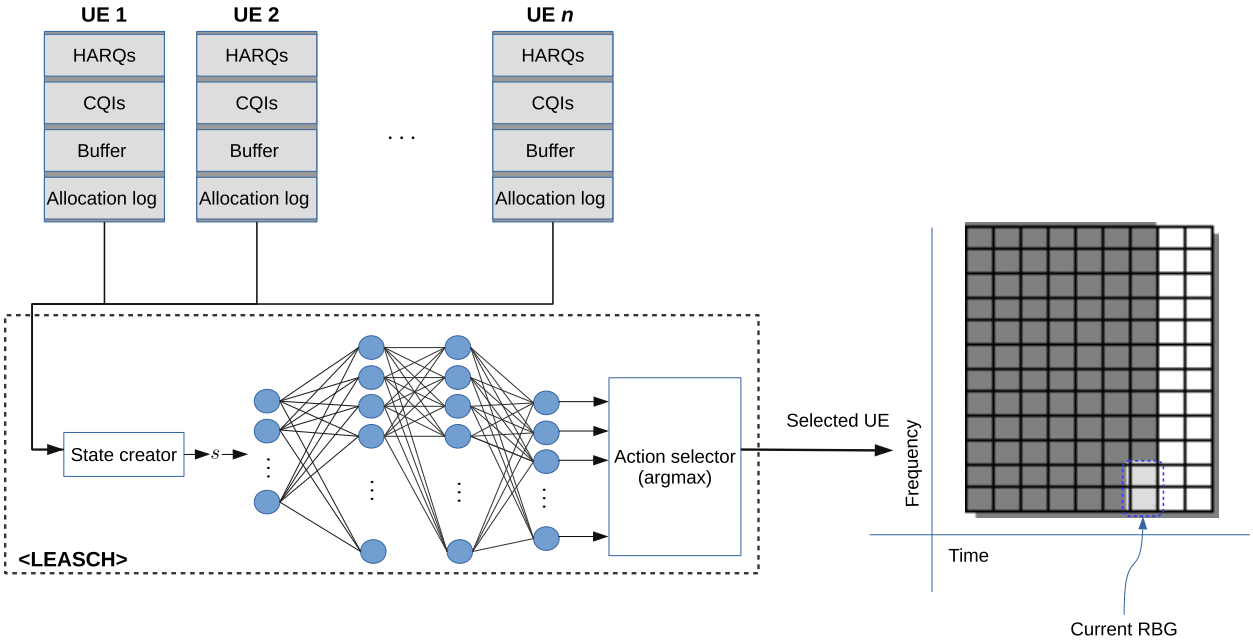


FIGURE 2. LEASCH deployment.

reactive RL, as the time horizon parameter increases. In addition, when compared to classical scheduling approaches like PF, the work shows enhancement in served network traffic but at the same time the average airtime allocation for Wi-Fi APs has degraded as the time horizon parameter increases. One disadvantage of this work is that it uses a heavy-weight architecture.

In [16] a lightweight multi-user deep RL approach is used to address the spectrum access problem, and a recurrent Q network (RQN) [37] with dueling [19] is used. At each time slot a user can only select a single channel to transmit, and if the transmission is successful then an ACK signal (observation) is sent back to the user, otherwise a collision has happened. When modeling this problem in an RL framework, the length of the action space of a user is a $|\mathcal{C}| + 1$ binary vector (one-hot), and $\mathcal{C}$ is the set of channels, indicating which channel was selected by the user. The first element of this vector is 1 if the user has decided to wait. The state is the history of actions and the observations made by a user u until time t. The reward is the achievable data rate. The training phase of this work is centralized, while the deployment phase is distributed, and the model weights are updated in each UE only when required, e.g., after substantial change in the UE behavior.

In [38], the duty cycle multiple access mechanism is used to divide the time frame between LTE and Wi-Fi users. A DLR approach is then used to find the splitting point based on the feedback averaged from the channel status for several previous frames. Information like idle slots, number of successful transmissions, action, reward are used to represent the state of the agent. The action is a splitting point in the time frame (i.e., an integer), and the reward is the transmission time given to the LTE users while not violating the Wi-Fi users minimum data rate limit.

In [35] a DQN model is developed where the agent learns by interacting with users that use other protocols, like TDMA and ALOHA, and learns to send its data in the slots where the other users are idle.

IV. THE PROPOSED DRL SCHEDULER (LEASCH)

The proposed scheduler (LEASCH) is shown in Figure 2. It is developed through two stages: Training and testing (deployment).

In the training phase, the scheduling task is transformed into an episodic DRL learning problem and LEASCH is trained until it converges. In the testing phase, a 5G system level simulator is used to deploy LEASCH. These two phases are described deeply in the following subsections. Each component of LEASCH is described from a DRL perspective first, and then the training and deployment algorithms are presented.

A. LEASCH'S DESIGN

1) STATE

Let us recall the objective of our agent as a scheduler. At a given RBG, it has to select an active (eligible) UE from a set of candidate UEs and assign that RBG to the selected UE (Figure 1). Our objective is to jointly optimize the throughput and fairness. Therefore, we can divide our state into three parts: **eligibility**, **data rate**, and **fairness**. We derive each part separately and then combine them in a single input vector representing the state.

a: ELIGIBILITY

At each RBG there is only a subset of eligible UEs, $\hat{\mathcal{U}} \subseteq \mathcal{U}$. A user is eligible for scheduling at a given RBG if the UE has data in the buffer and is not associated with a HARQ process. However, instead of feeding the buffer and the HARQ status

of each UE to LEASCH, and ask the agent to learn ''eligibility'', we simplify the task for the agent by calculating a binary vector $\boldsymbol{g}$ to act as an eligibility indicator:

$$g_u = \begin{cases} 1, & \text{if } u \text{ is eligible} \\ 0, & \text{Otherwise} \end{cases}, \quad \forall u \in \mathcal{U} \tag{18}$$

As we will see, $\boldsymbol{g}$ will help us designing a tangible reward function that allows the agent to effectively learn how to avoid scheduling inactive UEs.

b: DATA RATE

One way to represent this piece of information in the agent state is to use the data/bit rate directly. However, we use the valid entries of modulation and coding schemes (MCSs) in Table 5.1.3.1-2 in the 5G physical layer specification TS 38.214 [39] to model this information. We denote this information vector by $\boldsymbol{d}$.

c: FAIRNESS

We keep track each time a UE is admitted to an RBG. To that end, a vector with all-zero elements $\boldsymbol{f} = \boldsymbol{0}$ is created in the beginning of each episode, and $\boldsymbol{f}$ is updated each time an RBG is scheduled:

$$f_u = \begin{cases} \max(f_u - 1, 0), & \text{if } u \text{ is selected} \\ f_u + 1, & \text{if } u\text{'s buffer is not empty} \end{cases}, \quad \forall u \in \mathcal{U} \tag{19}$$

Therefore, $\boldsymbol{f}$ represents the *allocation-log* of the resources. In the best case scenario all entries of $\boldsymbol{f}$ are the same, meaning that all UEs are admitted to the resources with the same probability. In addition, $\boldsymbol{f}$ also represents the delay because the value in $\boldsymbol{f}$ will be large if the UE did not access the resources for too long.

Combining these three vectors $\boldsymbol{g}, \boldsymbol{d}$ and $\boldsymbol{f}$ yields the state. The size of the state can be further reduced by joining $\boldsymbol{g}$ and $\boldsymbol{d}$ via the Hadamard product:

$$\hat{\boldsymbol{d}} = \boldsymbol{d} \circ \boldsymbol{g}$$

making the final state vector defined by:

$$\boldsymbol{s} = \begin{bmatrix} \hat{\boldsymbol{d}} & \boldsymbol{f} \end{bmatrix}^\top \tag{20}$$

This way our state represents all pieces of information in a compact but descriptive manner. For a better learning stability we normalize $\hat{\boldsymbol{d}}$ and $\boldsymbol{f}$ to the range [0, 1].

2) ACTION

The action space $\mathcal{A}$ is $\mathcal{U}$. Each action is encoded in hot-one encoding. In this encoding, only the selected UE (i.e., action) will be 1 while the other elements will be 0.

3) REWARD

Reward engineering is a key problem in RL. In general, the reward is treated similarly to an objective function to be maximized. However, we believe that it should be engineered as a signal such that each state-action pair represents a meaningful reward.

From our state design the goal is to encourage the agent to transmit at the RBGs with the highest MCS, i.e., highest bit-per-symbol, to increase the throughput in the system. At the same time, we would like the agent not to compromise the resource sharing between users. Therefore, the adopted reward is given by:

$$r(s, u; K) = \begin{cases} -K, & \text{if } u \text{ is none-eligible} \\ \hat{d}_u \times \dfrac{\min\limits_u f_u}{\max\limits_u f_u}, & \text{otherwise} \end{cases} \tag{21}$$

where K is a threshold to represent the negative penalization signal for scheduling an inactive UE, and $\boldsymbol{f}$ is updated using (19). We can easily see that, our reward is a variant of a discounted bestCQI function, where the data rate is discounted by the resource sharing fairness.

B. TRAINING PHASE

This phase is performed off-line. LEASCH is trained for a sequence of episodes. The training procedure of one episode is described in Algorithm 1. In the beginning of each episode, a random state is created. Then the agent is trained for a set of ℓ_{episode} steps. In each step the agent trains its on-line Q neural network, and transfers the learned parameters to the target critic neural network at every T steps. After an episode has finished, the experience replay memory $\mathcal{R}$ and the learned weights are transfered to the next episode, and so on. The state is reset in the beginning of each episode.

C. DEPLOYMENT PHASE

Once the training phase has finished, LEASCH is deployed in a 5G simulator for testing. In this phase, LEASCH performs a single forward step on its neural network and no retraining is required, Figure 2. The deployment algorithm is shown in Algorithm 2. In this algorithm, the agent is plugged in like any other conventional scheduling algorithm. Each time an RBG is ready for scheduling, it is admitted to LEASCH which first calculates the set of eligible UEs, $\hat{\mathcal{U}}$, and creates a state $\boldsymbol{s}$. Next, it decides which UE wins the RBG by performing a forward step on its neural network with weights $\boldsymbol{\theta}$ and chooses the action with the highest probability. If the selected UE, u, belongs to $\hat{\mathcal{U}}$ then LEASCH assigns the current RBG to u. According to LEASCH's decision, the simulator allocates the resources and records statistics.

V. RESULTS

In order to evaluate the proposed scheduler, a comparison with two baseline algorithms, proportional fairness (PF) and round robin (RR), is performed. These are widely used algorithms in literature and in practice. The main objective here is to assess LEASCH using different settings in order to: *i*) show its ability to solve the RRS problem; *ii*) try to understand which policy it was able to learn; and *iii*) to analyze the

Algorithm 1 Training Phase of LEASCH

// input: $\ell_{episode}$, K, M, T, ϵ, δ_ϵ, $\min_\epsilon$, $\boldsymbol{\theta}$, $\hat{\boldsymbol{\theta}}$, $\mathcal{R}$.
// output: updated $\{\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}, \mathcal{R}\}$.
initialize $\boldsymbol{s}$ randomly according to the ranges of $\hat{\boldsymbol{d}}$ and $\boldsymbol{f}$
for $i = 1 : \ell_{\text{episode}}$ **do**
  forward $\boldsymbol{s}$ to the on-line Q neural network and get the selected UE, u, via ϵ-greedy as:
  $u = \arg\max_{a \in \mathcal{A}} Q(\boldsymbol{s}, a; \boldsymbol{\theta})$
  anneal ϵ as: $\max\{\epsilon - \delta_\epsilon, \min_\epsilon\}$
  calculate the reward $r(\boldsymbol{s}, u; K)$ using (21).
  calculate new state s' using the equations (18) to (20)
  add the tuple $(\boldsymbol{s}, u, r, \boldsymbol{s'})$ to the experience replay $\mathcal{R}$
  sample M mini-batches from $\mathcal{R}$ and train the on-line Q neural network with $\boldsymbol{\theta}$ using (14) and (17)
  update the target critic Q neural network (with $\hat{\boldsymbol{\theta}}$) using $\boldsymbol{\theta}$ every T steps via smoothing (16).
  $\boldsymbol{s} \leftarrow \boldsymbol{s'}$
end for
return $\{\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}, \mathcal{R}\}$

Algorithm 2 Deployment Phase of LEASCH in 5G

// input: trained LEASCH.
for each time slot **do**
  for each RBG **do**
    calculate the set of eligible UEs $\hat{\mathcal{U}}$
    if $\hat{\mathcal{U}} \neq \emptyset$ **then**
      calculate state $\boldsymbol{s}$
      forward $\boldsymbol{s}$ to LEASCH
      calculate the action u as:
      $u = \arg\max_{a \in \mathcal{A}} Q(s, a; \boldsymbol{\theta})$
      if $u \in \hat{\mathcal{U}}$ **then**
        schedule u for the current RBG
      end if
    end if
    collect statistics from the simulator
  end for
end for

quality of its design. The collected results were analyzed from different perspectives in order to accomplish these goals.

A. EXPERIMENTAL SETUP

The parameters adopted for LEASCH and 5G simulator are depicted in Tables 2 and 3, respectively. As for LEASCH's architecture, its Q neural networks are DNNs with two fully connected hidden layers of 128 neurons each, and `relu` activation functions. The number of layers and neurons are selected empirically. The input layer size is $2 \times |\mathcal{U}|$ while the output layer is a layer of size $|\mathcal{U}|$.

All methods and algorithms presented/discussed here are implemented in Matlab 2019b in a PC running Linux with i7 2.6GHz, 32GB RAM, and GPU Nvidia RTX 2080Ti with 11 GB.

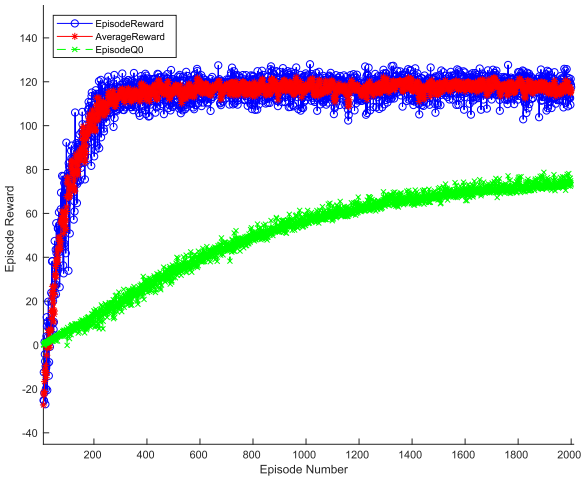


FIGURE 3. LEASCH learning curve for 2000 episodes.

TABLE 2. Adopted DRL parameters/hyper-parameters.

Parameter	Value	Description
α	$1e^{-4}$	DNN learning rate
Optimizer	Adam	
Gradient threshold	1	
ϵ	0.99	ϵ-greedy parameter
$\min_\epsilon$	0.01	Min. allowed ϵ
δ_ϵ	$1e^{-4}$	ϵ decaying factor
$\lvert\mathcal{R}\rvert$	$1e^{6}$	Experience replay memory size
M	64	Mini-batch size
T	20	Smoothing frequency
β	$1e^{-3}$	Smoothing threshold
ℓ_{episode}	150 RBG	Episode length
No. of episodes	500	Training episodes

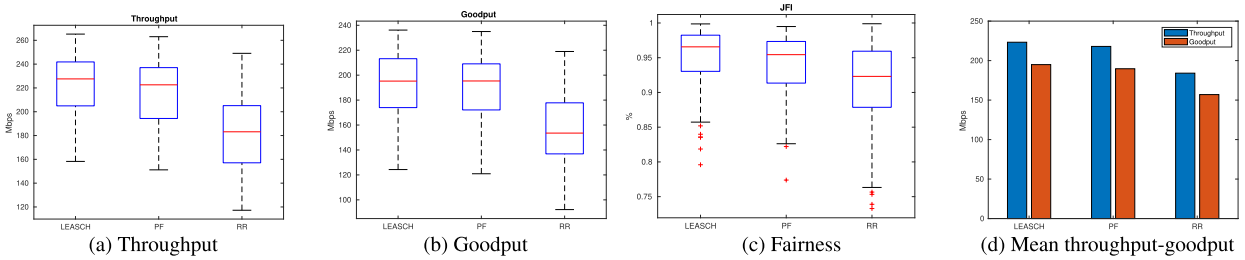
TABLE 3. Adopted parameters for LEASCH testing on 5G network.

Parameter	Value
Radio access tech.	3GPP 5G NR
Test time	250 frames
Simulation runs	100 runs with different deployment scenarios
Numerology index μ	$\{0, 1, 2\}$ see Table 1
Bandwidth	{5MHz, 10MHz, 20MHz}
UEs	4
SCS	{15kHz, 30kHz, 60kHz}
No. of RBs	$\{25, 24, 24\}$ see [42]
Scheduling period	1 RGB
RBG size	2 RBs according to configuration 1 in [16]
Total tested RBGs	$250 \times 100 \times \{130, 240, 480\}$ RBGs
Channel development	Randomly changes each $\frac{1}{4}$ second
HARQ	True

In the training phase, LEASCH is trained in a pool of parallel threads in the GPU. As shown in Figure 3, LEASCH was able to converge in less than 300 episodes. The theoretical (long term) reward, the green line in the graph, has also shown a steady increase which indicates a stable learning of LEASCH with each episode. In addition, the average reward (averaged each 5 episodes) has revealed a stable experience by the agent.

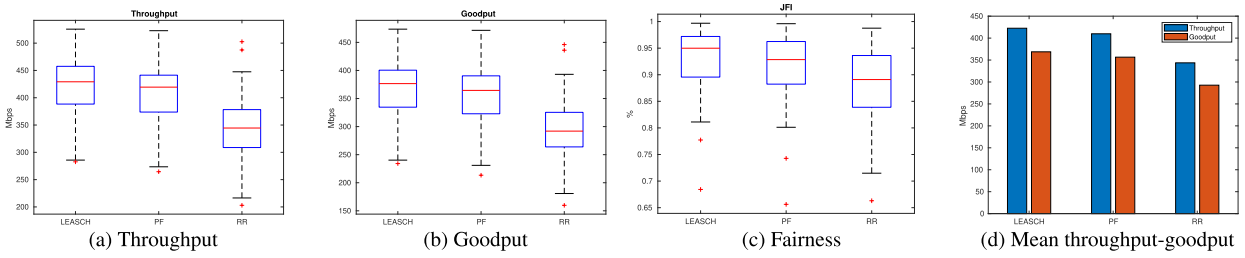
KEY PERFORMANCE INDICATORS

Throughput, goodput, and fairness are the main key performance indicators (KPIs) used for evaluation. For throughput, the sum of achievable data rate in the cell is reported.



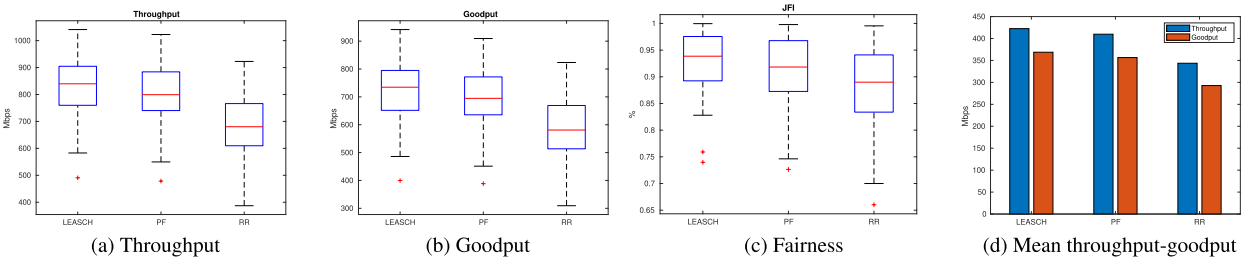
(a) Throughput (b) Goodput (c) Fairness (d) Mean throughput-goodput

FIGURE 4. KPIs for 250 frames of 15kHz SCS under 5MHz BW for 100 runs.



(a) Throughput (b) Goodput (c) Fairness (d) Mean throughput-goodput

FIGURE 5. KPIs for 250 frames of 30kHz SCS under 10MHz BW for 100 runs.



(a) Throughput (b) Goodput (c) Fairness (d) Mean throughput-goodput

FIGURE 6. KPIs for 250 frames of 60kHz SCS under 20MHz BW for 100 runs.

For goodput, the delivered data rate is measured at the receiver. For fairness, the popular Jain's fairness index (JFI) is used.

B. ABILITY TO SOLVE RRS

LEASCH and the baseline algorithms have been tested on different channel bandwidths: 5, 10 and 20 MHz; and different numerology indexes: 15, 30 and 60 kHz SCS. See Table 3.

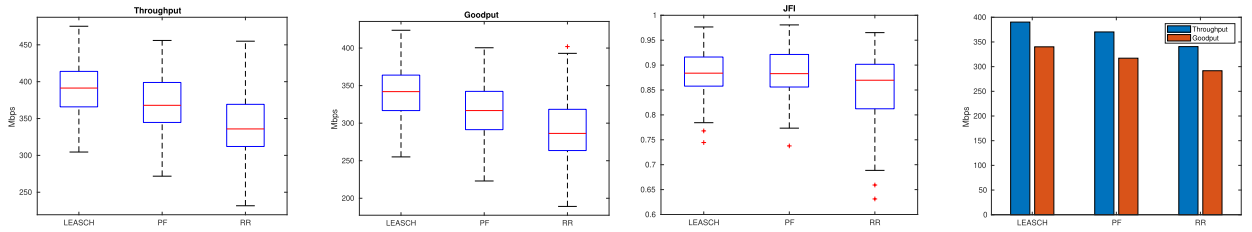
The results of the first group of settings, i.e., 5 MHz BW and 15 kHz SCS, are shown in Figure 4. These results clearly demonstrate that LEASCH is better than the baseline in all KPIs. LEASCH has improved the throughput by $\approx$ 2.4% and 18% compared to PF, and RR, respectively. In terms of goodput, LEASCH is better by $\approx$ 3% and 20 % compared to PF and RR, respectively, which indicates a better stability in LEASCH performance when compared to the baseline. For the JFI, LEASCH is $\approx$ 1% and 4.3% better than PF and RR, respectively.

For the second set of settings, i.e., 10MHz BW and 30kHz SCS, LEASCH has improved the throughput by $\approx$ 3% and 19% compared to PF and RR, respectively. In terms of goodput, LEASCH is $\approx$ 3.3% and 21% better than PF and RR, respectively. Regarding JFI, LEASCH is $\approx$ 2% and 5% better than PF and RR, respectively.

The third set of settings, i.e., 20MHz BW and 60Khz SCS, has also shown similar performance where LEASCH has improved the throughput by $\approx$ 3% and 18% compared to PF and RR, respectively. For goodput, LEASCH outperformed PF and RR by $\approx$ 4% and 20%, respectively. Regarding JFI, LEASCH improved the fairness compared to PF and RR by $\approx$ 2% and 5%, respectively.

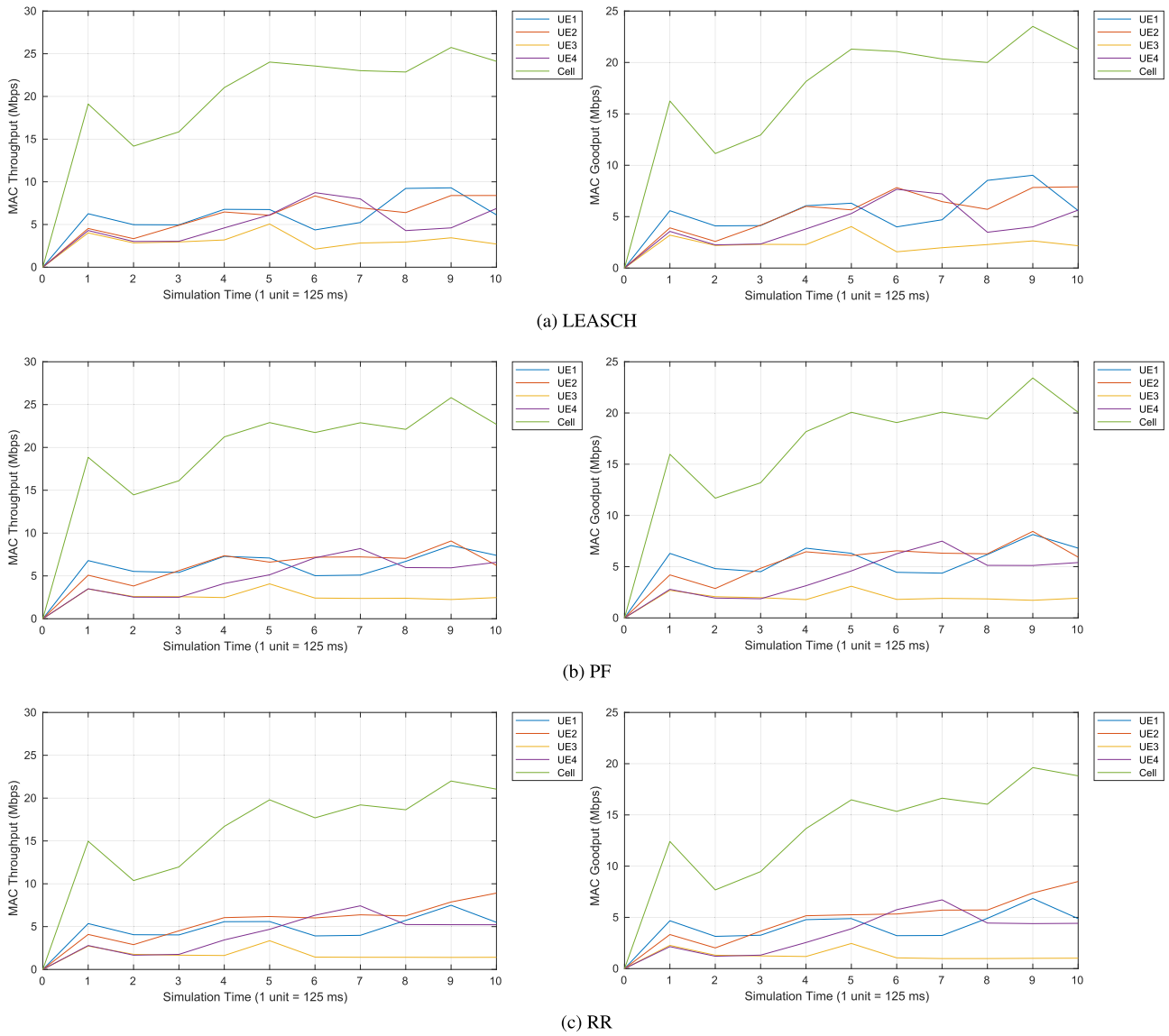
These results have clearly shown that LEASCH has a competitive and consistent performance compared to the baseline. LEASCH has shown improvement in all measurements, which is not an easy task given that LEASCH has a simple design and has been trained off-simulator. In addition, when choosing a setting with higher theoretical throughput (e.g., 10MHz with 30kHz SCS instead of 5MHz with 15kHz SCS), LEASCH was able to scale well and improve the performance even further. One nice property of LEASCH is that it is able to push all the KPIs without compromising any of them. More specifically, LEASCH was able to improve the throughput but at the same time without compromising the goodput. This is why the goodput is enhanced even more than the throughput in all tests compared to the baseline.

In addition, we have also doubled the number of UEs and selected the second settings, i.e., 10MHz BW and 30kHz SCS to retest all methods. The results are shown in Figure 7.



(a) Throughput (b) Goodput (c) Fairness (d) Mean throughput-goodput

FIGURE 7. KPIs for 8 UEs simulated for 250 frames using 30kHz SCS under 10MHz BW for 100 runs.



(a) LEASCH

(b) PF

(c) RR

FIGURE 8. A random testing run for the 10MHz BW and 30kHz SCS setting. Left column: throughput; right column: goodput.

From these figure, we can see that the proposed model still able to produce efficient results under larger set of UEs. In terms of throughput it is 5% and 13% better than PF and RR, respectively. Regarding goodput it is 7% and 14% better than PF and RR. For JFI, it has shown similar results as PF and is 2% better than RR.

C. WHICH POLICY DID LEASCH LEARN?

This section tries to analyze and figure out which policy did LEASCH learn. This task is not trivial, not only for LEASCH but for almost every DRL agent. Here it is more difficult not only because of the stochastic nature of LEASCH, but also due to the complexity of the RRS problem. Therefore,

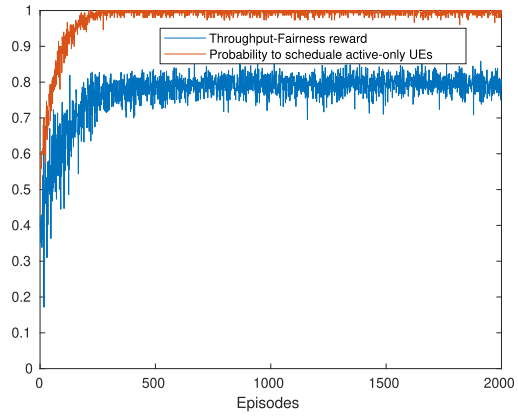


FIGURE 9. Learning different objectives by LEASCH.

the visual inspection approach of LEASCH behavior will be followed.

To that end, a testing run is sampled for a set of settings and the throughput and goodput curves are quantized into 10 time units (see Figure 8). These curves are then visually inspected with regard to those of PF and RR. By comparing these curves, both for each UE and for the cell, it is possible to construct an idea about which policy LEASCH has learned. In this figure, the second set of settings with 10MHz BW and 30kHz SCS is chosen. Since 30kHz SCS is used, the simulation time is only 1250 ms. For 15kHz this would be 2500 ms. This is due to the reduction in symbol duration as the numerology index increases.

According to the simulation settings in Table 3, the channel changes (and consequentially new CQI feedbacks are signaled) each 125 ms, i.e., each 1 time unit in Figure 8. In this Figure, first it is interesting to see that the trends of the cell curves are similar in all approaches. However, at each time unit each method makes different scheduling decisions. Second, LEASCH outperforms PF and RR since it reaches higher throughput-goodput, especially from the period 5 to 8 time units where major changes have occurred in the channel. Before time unit 5 (i.e., from 1 to 4) LEASCH performed almost identical to PF in terms of throughput-goodput but, at the same time, the UEs' curves are more compact in LEASCH which indicates a better fairness. After the period 5 to 8 time units, LEASCH continued to maintain high throughput without sacrificing UEs that have bad CQIs (e.g., compare the curve of UE3 in all approaches). Although our discussion here lacks analytical bases, due to the complexity of the problem, it is clear that LEASCH has nicely realized the intuitions we have designed it for. LEASCH tries to improve all KPIs without sacrificing UEs with bad CQIs, by wisely distributing the spectrum among all UEs.

D. LEARNING PERFORMANCE

Here the learning performance of LEASCH is analyzed. The main objective is to asses its design quality given that it has to learn two different goals: avoid scheduling inactive UEs, and jointly optimize throughput and fairness. Using only theoretical foundation of DRL, it is not easy to see how LEASCH learned these different (and perhaps contradictorily) goals. The reason is that, the learned weights of the Q networks can not easily be interpreted to assess the learning performance and the quality of LEASCH's design. Therefore, a reward analysis is performed by separating both goals outcomes.

To that end, the learning curve in Figure 3 is decomposed into two curves as shown in Figure 9. In addition, instead of calculating the average total reward of the episode (as in Figure 3), the average reward of each episode is used. This allows us to study how LEASCH learns both parts of expression (21) separately. From this figure, the red curve represents the probability of scheduling active-only UEs while the blue curve is the throughput-fairness reward, i.e., $\left\{\hat{d}_u \times \frac{\min_u f_u}{\max_u f_u}\right\}$ in (21). These two curves show that LEASCH was able to jointly learn these two objectives and, around episode 300, LEASCH was able to converge for both objectives which clearly indicates the effectiveness of LEASCH's design. In addition, this also shows the suitability of DRL to handle the scheduling problem, which is usually a multi-objective problem.

VI. CONCLUSIONS

This article presents LEASCH, a deep reinforcement learning agent able to solve the radio resource scheduling problem in 5G. LEASCH is a breed of DDQN critic-only agents that learns discrete actions from a sequence of states. It does so by adapting its neural networks, known as DQNs, weights according to the reward signal it receives from the environment. What makes LEASCH different from conventional schedulers is that; it is able to learn the scheduling task from scratch with zero knowledge about the RRS. LEASCH is different from the extremely scarce and new AI-schedulers in many things. First LEASCH is trained off-simulator to break any dependency between learning and deployment phases, making LEASCH a generic tool in any networking AI-ecosystem. Second, LEASCH has novel design not addressed in earlier approaches. Finally, LEASCH was designed as numerology-agnostic which makes it suitable for 5G deployments.

Concerning LEASCH performance, it has been compared to the well-established approaches PF and RR. Despite LEASCH's simple design it has shown clear improvement and stability in throughput, goodput, and fairness KPIs. Further analysis has also shown that LEASCH is able to learn not only how to enhance the classical throughput-fairness tradeoff, but also not to schedule inactive users. It was able to learn both objectives at the same time as the learning curves depicted. Another interesting property of LEASCH is that it avoids to penalize users with bad CQIs and tries to keep all KPIs high at the same time. Such property can be improved in the future. In addition, more interesting properties, which can not be easily obtained by conventional approaches, can be learned by LEASCH.

As a future work, a more advanced version of LEASCH will be developed to serve larger set of users. It will be developed and deployed under larger 5G network with a mixture of numerologies and more complex rewarding systems that include different type of services.

REFERENCES

[1] S.-Y. Lien, S.-L. Shieh, Y. Huang, B. Su, Y.-L. Hsu, and H.-Y. Wei, "5G new radio: Waveform, frame structure, multiple access, and initial access," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 64–71, Jun. 2017.

[2] C. G. Tsinos, S. Maleki, S. Chatzinotas, and B. Ottersten, "On the energy-efficiency of hybrid analog-digital transceivers for Single- and multi-carrier large antenna array systems," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 9, pp. 1980–1995, Sep. 2017.

[3] K. David and H. Berndt, "6G vision and requirements: Is there any need for beyond 5G?" *IEEE Veh. Technol. Mag.*, vol. 13, no. 3, pp. 72–80, Sep. 2018.

[4] F. Alvarez, D. Breitgand, D. Griffin, P. Andriani, S. Rizou, N. Zioulis, F. Moscatelli, J. Serrano, M. Keltsch, P. Trakadas, T. K. Phan, A. Weit, U. Acar, O. Prieto, F. Iadanza, G. Carrozzo, H. Koumaras, D. Zarpalas, and D. Jimenez, "An edge-to-cloud virtualized multimedia service platform for 5G networks," *IEEE Trans. Broadcast.*, vol. 65, no. 2, pp. 369–380, Jun. 2019.

[5] F. Al-Tam and N. Correia, "On load balancing via switch migration in software-defined networking," *IEEE Access*, vol. 7, pp. 95998–96010, 2019.

[6] C. G. Tsinos, S. Chatzinotas, and B. Ottersten, "Hybrid analog-digital transceiver designs for multi-user MIMO mmWave cognitive radio systems," *IEEE Trans. Cognit. Commun. Netw.*, vol. 6, no. 1, pp. 310–324, Mar. 2020.

[7] *System Architecture for the 5G System*, Standard 3GPP TS 23.501, Technical Report, ETSI, 2018.

[8] J. Jeon, "NR wide bandwidth operations," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 42–46, Mar. 2018.

[9] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5G wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1617–1655, 3rd Quart., 2016.

[10] Y. Fu, S. Wang, C.-X. Wang, X. Hong, and S. McLaughlin, "Artificial intelligence to manage network traffic of 5G wireless networks," *IEEE Netw.*, vol. 32, no. 6, pp. 58–64, Nov. 2018.

[11] N. Feamster and J. Rexford, "Why (and How) networks should run themselves," in *Proc. Appl. Netw. Res. Workshop*, Jul. 2018, p. 20.

[12] S. Joseph, R. Misra, and S. Katti, "Towards self-driving radios: Physical-layer control using deep reinforcement learning," in *Proc. 20th Int. Workshop Mobile Comput. Syst. Appl.*, Feb. 2019, pp. 69–74.

[13] H. B. Pasandi and T. Nadeem, "Challenges and limitations in automating the design of MAC protocols using machine-learning," in *Proc. Int. Conf. Artif. Intell. Inf. Commun. (ICAIIC)*, Feb. 2019, pp. 107–112.

[14] F. Capozzi, G. Piro, L. A. Grieco, G. Boggia, and P. Camarda, "Downlink packet scheduling in LTE cellular networks: Key design issues and a survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 678–700, 2nd Quart., 2013.

[15] *Physical Layer Procedures for Data*, Standard 3GPP TS 38.214, Technical Report, ETSI, 2019.

[16] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic spectrum access," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 310–323, Jan. 2019.

[17] S.-C. Tseng, Z.-W. Liu, Y.-C. Chou, and C.-W. Huang, "Radio resource scheduling for 5G NR via deep deterministic policy gradient," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2019, pp. 1–6.

[18] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, 3rd Quart., 2019.

[19] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," 2015, *arXiv:1511.06581*. [Online]. Available: http://arxiv.org/abs/1511.06581

[20] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[22] O. Vinyals *et al.*, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.

[23] V. Hado Hasselt, "Double q-learning," *Proc. Advances Neural Inf. Process. Syst.*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, A. Culotta, eds., 2010, pp. 2613–2621.

[24] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 30th Assoc. Advancement Artif.-Intell. (AAAI) Conf. Artif. Intell.*, Phoenix, AZ, USA, Feb. 2016, pp. 1–13.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[26] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th Annu. Conf. Internet Meas. IMC*, 2010, pp. 267–280.

[27] S. Chinchali, P. Hu, T. Chu, M. Sharma, M. Bansal, R. Misra, M. Pavone, and S. Katti, "Cellular network traffic scheduling with deep reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell. /30th Innov. Appl. Artif. Intell. Conf. / 8th AAAI Symp. Educ. Adv. Artif. Intell.*, New Orleans, LA, USA, Feb. 2018, pp. 1–9.

[28] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcon, M. Sole, V. Muntes-Mulero, D. Meyer, S. Barkai, J. Mike Hibbett, G. Estrada, K. Ma'ruf, F. Coras, V. Ermagan, H. Latapie, C. Cassar, J. Evans, F. Maino, J. Walrand, and A. Cabellos, "Knowledge-defined networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 47, no. 3, pp. 2–10, Jul. 2017.

[29] J. Wang, C. Xu, Y. Huangfu, R. Li, Y. Ge, and J. Wang, "Deep reinforcement learning for scheduling in cellular networks," 2019, *arXiv:1905.05914*. [Online]. Available: https://arxiv.org/abs/1905.05914

[30] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proc. 32nd AAAI Conf. Artif. Intell. /30th Innov. Appl. Artif. Intell. Conf. /8th AAAI Symp. Educ. Adv. Artif. Intell.*, New Orleans, LA, USA, Feb. 2018, pp. 3207–3214.

[31] P. Gawlowicz and A. Zubow, "Ns-3 meets OpenAI gym: The playground for machine learning in networking research," in *Proc. 22nd Int. ACM Conf. Modeling, Anal. Simulation Wireless Mobile Syst. MSWIM*, 2019, pp. 113–120.

[32] I.-S. Comsa, A. De-Domenico, and D. Ktenas, "QoS-driven scheduling in 5G radio access Networks–A reinforcement learning approach," in *Proc. GLOBECOM Global Commun. Conf.*, Dec. 2017, pp. 1–7.

[33] U. Challita, L. Dong, and W. Saad, "Proactive resource management for LTE in unlicensed spectrum: A deep learning perspective," *IEEE Trans. Wireless Commun.*, vol. 17, no. 7, pp. 4674–4689, Jul. 2018.

[34] L. Zhang, J. Tan, Y.-C. Liang, G. Feng, and D. Niyato, "Deep reinforcement learning-based modulation and coding scheme selection in cognitive heterogeneous networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 6, pp. 3281–3294, Jun. 2019.

[35] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1277–1290, Jun. 2019.

[36] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proc. 15th ACM SIGCOMM Workshop Hot Topics Netw. (HotNets)*, Atlanta, GA, USA, Nov. 2016, pp. 50–56.

[37] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," in *Proc. 2015 AAAI Fall Symp. Ser.*, Sep. 2015, pp. 1–52.

[38] J. Tan, L. Zhang, Y.-C. Liang, and D. Niyato, "Deep reinforcement learning for the coexistence of LAA-LTE and WiFi systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[39] *Physical Channels and Modulation*, Standard 3GPP TS 38.211, Technical Report, ETSI, Jul. 2018.

[40] *User Equipment (UE) Radio Transmission and Reception;—Part 1: Range 1 Standalone (Release 16)*, Standard 3GPP TS 38.101-1-NR, Technical Report, ETSI, Sep. 2019.

FAROQ AL-TAM received the degree in computer science (four years) from the University of Thamar (THU), in 2004, and the master's and Ph.D. degrees in computer science from the University of Algarve (UAlg), Portugal, in 2012 and 2016, respectively. He is currently a full-time Researcher with the Networks and Systems Group, Center for Electronics, Optoelectronics, and Telecommunications (CEOT), a research center supported by the Portuguese Foundation for Science and Technology (FCT), University of Algarve. His major interests are modeling and optimization problems in image processing and computer networks.

NOÉLIA CORREIA received the B.Sc. and M.Sc. degrees in computer science from the University of Algarve, Faro, Portugal, in 1995 and 1998, respectively, and the Ph.D. degree in optical networks (computer science) from the University of Algarve, in 2005, in collaboration with University College London, U.K. She is a Lecturer with the Faculty of Sciences and Technology, University of Algarve. She is a Founding Member of the Center for Electronics, Optoelectronics, and Telecommunications, a research center supported by the Portuguese Foundation for Science and Technology, University of Algarve. She is also the Networks and Systems Group Coordinator. Her research interests include the applications of optimization techniques to several network design problems in optical, wireless, and sensor networks fields, and the development of algorithms.

JONATHAN RODRIGUEZ (Senior Member, IEEE) received the master's degree in electronics and electrical engineering and the Ph.D. degree from the University of Surrey, U.K., in 1998 and 2004, respectively. In 2005, he became a Researcher of the wireless communications scientific area with the Instituto de Telecomunicações (IT), Portugal, where he was a member. In 2008, he became a Senior Researcher, where he established the 4TELL Research Group targeting next-generation mobile systems. Since 2009, he has been serving as an Invited Assistant Professor with the University of Aveiro, Portugal, and attained the Associate Level, in 2015. He is currently the Coordinator of the H2020-SECRET Innovative Training Network. In 2017, he was appointed as a Professor of mobile communications with the University of South Wales, U.K. He has served as a Project Coordinator of major international research projects, including Eureka LOOP and FP7 C2POWER whilst serving as the Technical Manager of FP7 COGEU and FP7 SALUS. He has authored more than 500 scientific works, including ten editorial books. His professional affiliations include a chartered engineer (C.Eng.) (since 2013) and a Fellow of the IET, in 2015.