Deep Reinforcement Learning for QoS-Constrained Resource Allocation in Multiservice Networks

Juno V. Saraiva, Iran M. Braga Jr., Victor F. Monteiro, F. Rafael M. Lima,
Tarcisio F. Maciel, Walter C. Freitas Jr. and F. Rodrigo P. Cavalcanti

***Abstract*—In this article, we study a Radio Resource Allocation (RRA) that was formulated as a non-convex optimization problem whose main aim is to maximize the spectral efficiency subject to satisfaction guarantees in multiservice wireless systems. This problem has already been previously investigated in the literature and efficient heuristics have been proposed. However, in order to assess the performance of machine learning algorithms when solving optimization problems in the context of RRA, we revisit that problem and propose a solution based on a Reinforcement Learning (RL) framework. Specifically, a distributed optimization method based on multi-agent deep RL is developed, where each agent makes its decisions to find a policy by interacting with the local environment, until reaching convergence. Thus, this article focuses on an application of RL and our main proposal consists in a new deep RL based approach to jointly deal with RRA, satisfaction guarantees and quality of service constraints in multiservice celular networks. Lastly, through computational simulations we compare the state-of-art solutions of the literature with our proposal and we show a near optimal performance of the latter in terms of throughput and outage rate.**

***Index Terms*—Radio resource allocation, quality of service, satisfaction guarantees, reinforcement learning, deep Q-learning.**

I. Introduction

The overall performance of wireless telecommunications systems directly depends on how efficiently the available resources are managed, e.g., subcarriers, time slots, transmit power, antennas, among others. Consequently, optimal Radio Resource Allocation (RRA) is one of the fundamental challenges and a key requirement for the design of efficient mobile networks. RRA problems in general are formulated as optimization problems and different objective functions and constraints have been considered in the literature. Two examples of classical objective functions are: maximize the system throughput, as considered in [1] and [2] and guarantee system fairness, as done in [3]. However, with the advent of the Fifth Generation (5G) of mobile wireless telecommunications, which shall integrate new technology components, RRA problems can become even harder to tackle, with larger optimization domains and a series of practical considerations. Moreover, these problems need also to deal with the advanced RRA functionalities, the growing variety of scenarios and sophisticated types of services and/or Quality of Service (QoS) constraints of users [4].

Although it is possible to apply optimal methods to some of these problems, such as exhaustive search and Branch and Bound (BB) algorithm, their high computational complexities are prohibitive and, therefore, these methods are not appealing for large-scale mobile networks. Contrarily, techniques such as Lagrangian relaxations, iterative distributed optimization and heuristic algorithms normally have reduced computational costs, but they fail in achieving the maximum performance and are usually tailored to specific network configurations [4]. Furthermore, issues related to convergence and optimality gaps of these solutions can be unknown as well [5]. As a result, the solution of many problems in the context of RRA can be quite inadequate using conventional optimization methods. There already exists a very rich literature on this topic, as it can be seen in [6], where an extensive survey on these techniques is presented.

Machine Learning (ML) techniques are leading the advent of the fourth industrial revolution due to its capabilities of solving complex problems. In fact, this has aroused interest of many researchers in the mobile communications field. More specifically, in learning-based resource allocation, a branch of ML called deep learning has gained notoriety and shown its potential in this type of context. Moreover, in order to further improve the performance of this technique with regard to learning from high-dimensional raw input data and make intelligent decisions, in [7], it is proposed a sophisticated approach. Basically, the authors combined deep learning and Reinforcement Learning (RL), resulting in a promising and powerful technique known as deep RL. In summary, in deep RL, a Deep Neural Network (DNN) along with other techniques, e.g., replay memory, are used in order to carry out a stable and efficient training.

Applying deep RL to cellular mobile networks can lead to the following main advantages: (1) a DNN with a moderate size can quickly perform predictions as only a small number of simple operations are needed to obtain an output. This is interesting and also helps the deep RL agent to get to know his environment faster; (2) the fact that deep RL agent learns directly from the raw collected network data with high dimension in large environments is not a problem due to the powerful representation capabilities of DNNs; (3) by exploiting distributed and/or parallel computing employing multiple machines and multiple cores, the response time of deep RL-based schemes can be greatly reduced and its performance increased; (4) deep RL-based schemes can also

This work was supported by Ericsson Research, Sweden, and Ericsson Innovation Center, Brazil, under UFC.46 and UFC.47 Technical Cooperation Contracts Ericsson/UFC. Iran M. Braga Jr. would like to acknowledge CAPES for its financial support under the grant 88887.474363/2020-00.

The authors are with the Wireless Telecommunications Research Group (GTEL), Federal University of Ceará (UFC), Fortaleza, Ceará, Brazil.

Digital Object Identifier: 10.14209/jcis.2020.7

improve over time since the deep RL agent aims at optimizing a long-term performance, considering the impact of actions on future rewards. This makes deep RL efficient in dealing with imprecise input data such as the Channel State Information (CSI) and makes it capable of learning how to behave in an unknown environment [8]; (5) deep RL is a model-free approach, i.e., it does not rely on a specific system model and, therefore, it can be easily extended to different contexts [9].

Motivated by the benefits of deep RL, in this paper, we revisit the problem of maximizing system throughput subject to minimum satisfaction constraints per service, as in [1] and [2], and we propose a new near-optimal RRA solution based on multiple agent deep RL.

The remainder of this paper is organized as follows. In Section II previous related works are reviewed, and the main contributions of our work are highlighted. In Section III and Section IV, we present the system modeling and formulate the optimization problem to be solved, respectively. Section V presents a deep RL-based method to solve the problem formulated in the previous section. Section VI and Section VII show simulation results and the main conclusions of this study, respectively.

II. State-of-the-Art and Main Contributions

RL techniques have recently been used in a variety of wireless resource management problems such as, channel and power allocation, throughput maximization and spectrum sharing. In [10], for example, a self-organizing method to allocate power to millimeter Wave (mmW) Base Stations (BSs) is proposed based on Q-learning technique. Q-learning is the most popular RL algorithm, where an agent interacts over time with its environment based on trial-and-error in order to learn a policy to achieve a given goal [11]. Thus, based on this technique, in [10], each BS acts as an independent agent taking actions such as choosing a transmit power. In other words, each BS sees the others as part of the environment and do not communicate with each other. In this case, the environment, for a given BS, is seen as a source of interfering signals. The problem with this solution is that, in a non stationary environment as mobile wireless networks, it takes a long time to converge, due to the lack of cooperation between the BSs.

Q-learning based resource allocation models are also proposed in [12] and [13]. Specifically, in [12], Q-learning is used to select which network node a User Equipment (UE) should connect to in order to minimize the total transmit power. It is considered that a UE can either directly connect to a BS or to another UE, which relays the traffic from a BS. According to that solution, each UE autonomously selects the node to which it will be connected and keeps a record of its experience when using that node. The records, called rewards, reflect the degree of fulfillment of the optimization target, e.g., total transmit power used by BSs and also other constraints such as required bit rate. In [13], we have proposed a Q-learning based solution to the same problem that we address in the present paper, i.e., schedule frequency resources to UEs in order to maximize the system throughput subject to users' QoS requirements, in terms of UE throughput. However, in general, Q-learning based solutions, and especially the one proposed in [13], present a scalability problem, since the agent's experiences are stored in a look-up table, called Q-table. This becomes an issue when the set of possible experiences increases with the dimensions of the environment, which is the case in [13], where the size of the Q-table is proportional to the number of frequency resources and the number of UEs.

As presented in the previous section, when a DNN is used by an agent, it is referred to as deep RL and this technique has shown to be efficient for RRA in large cellular networks. In [14], for example, a model-free deep RL method is applied to perform dynamic transmit power allocation. The solution presented in that work achieves near-optimal performance and is suitable for practical scenarios where the system model is inaccurate, thus overcoming some issues of classical and heuristic solutions. A decentralized band and power allocation problem for a Vehicle-to-Vehicle (V2V) communication system is solved based on deep RL in [15]. In details, the goal is to minimize interference under latency constraints, where each V2V link operates as an agent making its own allocation decisions. Moreover, many other papers also successfully address deep RL in several wireless telecommunications research areas, such as resource scheduling [16] and mobile edge computing/caching [17], [18].

The above mentioned works consider either a completely centralized solution [17], [18] or a decentralized solution [14]–[16]. On one hand, in a centralized solution, a deep RL agent is localized in a central node and it is responsible for the entire processing. Unfortunately, this processing consumes a lot of computational resources since the employed DNN size is proportional to the wireless network dimension. On the other hand, in a decentralized solution, multiple agents are considered each one running in a different node and taking actions independently. In this last option, the agents can either work independently or in cooperation with the burden of a longer convergence time or a higher signalling overhead and data updating procedures, respectively. In the present work, we propose another option, where we assume a centralized node, but with multiple agents running in parallel, each one related to a frequency resource. In addition, this structure is executed independently in a distributed way in different cores or machines in order to improve the performance of the system.

In summary, our main contributions are the following:

1) We revisit an important RRA problem originally studied with the help of optimization tools and heuristics and propose a new methodology that leverages decentralized deep RL. This methodology could be applied in other RRA problems to obtain alternative solutions. To the best of our knowledge, this strategy has not yet been applied to frequency resource scheduling problems.
2) By means of extensive computer simulations, we show that our proposed deep RL-based solution outperfoms the state-of-art solutions found in the literature. Indeed, this is interesting because it shows that our deep RL approach is capable of dealing with a challenging

scenario of mobile networks that includes satisfaction of users' QoS in different types of service plans, something that few papers consider in the literature.

III. System Modeling

We assume that resource allocation should be performed in a downlink of a cellular system composed by an Evolved Node B (eNB) serving a set $\mathcal{J} = \{1, \ldots, J\}$ of UEs distributed on its coverage area. Both users and BS employ single antenna transceivers. Moreover, we assume that there is no intracell interference due to the assignment of orthogonal resources with the use of Time Division Multiple Access (TDMA) and Orthogonal Frequency Division Multiple Access (OFDMA) multiple access schemes. Therefore, the smallest scheduling unit in our work is termed as Resource Block (RB) in which each RB consists of a group of one or more Orthogonal Frequency Division Multiplexing (OFDM) subcarriers in the frequency domain and a set of consecutive OFDM symbols in the time domain, whose total duration represents a Transmission Time Interval (TTI). We define N and $\mathcal{N}$ as the total number of RBs and the set of available RBs, respectively. Regarding inter-cell interference, we assume that it is added to the thermal noise in the Signal to Noise Ratio (SNR) expression, defined later.

We also assume a centralized approach where the eNB is responsible to make decisions about RB assignment and and it requires full channel state information of the links between the eNB and UEs. Nevertheless, note that channel state estimation schemes are out of the scope of our article. Moreover, the mobile radio channel coefficients are kept approximately constant during a TTI. In addition, we assume a multiservice scenario where the system operator supports L service plans contained in the set $\mathcal{L} = \{1, \ldots, L\}$. Thus, in each TTI, the J UEs compete for the available RBs in order to meet their throughput requirements, ξ_j, which are defined by their service plans and each service plan $l \in \mathcal{L}$ requires a minimum number of UEs, η_l, that should be satisfied. The set of all UEs from service $l \in \mathcal{L}$ is $\mathcal{J}_l$ with $|\mathcal{J}_l| = J_l$, where $|\cdot|$ denotes the cardinality of a set and $\mathcal{J}_l$ is the set of UEs from service $l \in \mathcal{L}$. Besides, each UE subscribes to only a single service plan, i.e., $\mathcal{J}_{l_1} \cap \mathcal{J}_{l_2} = \emptyset$, $\forall l_1, l_2 \in \mathcal{L}$ and $l_1 \neq l_2$.

Similar to [1], [2] and [13], power allocation is not optimized herein and we employ Equal Power Allocation (EPA) among RBs, which is the most basic and common power allocation scheme. Hence, the power p_n allocated to each RB n is fixed and equal to P/N, where P is the available power at the eNB. Therefore, the SNR $\Gamma_{j,n}$ of UE $j \in \mathcal{J}$ in RB $n \in \mathcal{N}$ is given by

$$\Gamma_{j,n} = \frac{p_n \cdot \varphi_j \cdot |h_{j,n}|^2}{\sigma^2}, \tag{1}$$

where φ_j models the joint effect of the path loss and shadowing of the link between the eNB and UE j; $|h_{j,n}|$ represents the magnitude of the complex channel frequency response of RB n when assigned to UE j; and, finally, σ^2 is the noise power at the receiver in the bandwidth of a given RB.

Finally, We assume $f(\cdot)$ as the link adaptation function responsible for mapping the achieved SNR to the transmit rate. It is a discrete and monotonic increasing function that models the Modulation and Coding Scheme (MCS) levels so that the transmission parameters at the physical layer are adapted according to the current channel state. Thus, we consider that the transmit rate when the RB n is assigned to UE j is $r_{j,n}$ such that $r_{j,n} = f(\Gamma_{j,n})$.

IV. Problem Formulation and Optimal Solution

As presented in Section II, the problem investigated herein aims to maximize the system throughput constrained by a per-service minimum number of satisfied UEs in a given TTI. For that problem, we define $x_{j,n}$ as the binary decision variable that assumes the value 1 when RB n is assigned to UE j and 0, otherwise. Furthermore, let R_j be the total throughput allocated to a UE j, i.e., $R_j = \sum_{n \in \mathcal{N}} r_{j,n} x_{j,n}$. According to the previous considerations, the resource assignment problem can be formulated as the following optimization problem:

$$\max_{\mathbf{X}} \sum_{j \in \mathcal{J}} R_j, \tag{2a}$$

$$\text{s.t.} \sum_{j \in \mathcal{J}} x_{j,n} = 1, \ \forall n \in \mathcal{N}, \tag{2b}$$

$$\sum_{j \in \mathcal{J}_l} u(R_j, \xi_j) \geq \eta_l, \ \forall l \in \mathcal{L}, \tag{2c}$$

$$x_{j,n} \in \{0, 1\}, \ \forall j \in \mathcal{J} \text{ and } \forall n \in \mathcal{N}, \tag{2d}$$

where $\mathbf{X}$ is the matrix of optimization variables composed of $x_{j,n}$ and $u(a, b)$ in (2c) denotes the Heaviside step function, which assumes the value 1 if $a \geq b$ and 0, otherwise. The constraints (2b) and (2d) guarantee that an RB can be shared within the cell, i.e., each RB can be assigned to a single UE. Finally, the constraint (2c) states that a minimum number of UEs should have their QoS requirements fulfilled in terms of throughput for each service plan.

It is worth noting that (2) is a combinatorial optimization problem with a non-convex constraint (2c), which has a prohibitive computational complexity depending on the problem dimensions. Moreover, since $u(\cdot)$ is neither convex nor concave, the optimal solution of (2) becomes harder to find. In order to simplify the optimal solution analyses, we linearize (2c) by new optimization variable (slack variable) and replace constraint (2c) by two sets of linear constraints as follows

$$R_j \geq \xi_j \cdot \rho_j, \ \forall j \in \mathcal{J}, \tag{3a}$$

$$\sum_{j \in \mathcal{J}_l} \rho_j \geq \eta_l, \ \forall l \in \mathcal{L}, \tag{3b}$$

where ρ_j is a binary selection variable that assumes the value 1 if UE j is selected to be satisfied and 0, otherwise. Thus, $\rho_j = 1$ in (3a) implies that UE j is satisfied whereas the constraint (3b) means that for all service l there are at least η_l satisfied UEs. In this way, problem (2) can be equivalently reformulated by substituting the constraint (2c) by two new constraints as follows:

$$\max_{\mathbf{x},\boldsymbol{\rho}} \sum_{j\in\mathcal{J}} R_j, \tag{4a}$$

$$\text{s.t.} \sum_{j\in\mathcal{J}} x_{j,n} = 1,\ \forall n \in \mathcal{N}, \tag{4b}$$

$$R_j \geq \xi_j \cdot \rho_j,\ \forall j \in \mathcal{J}, \tag{4c}$$

$$\sum_{j\in\mathcal{J}_l} \rho_j \geq \eta_l,\ \forall l \in \mathcal{L}, \tag{4d}$$

$$x_{j,n},\ \rho_j \in \{0,1\},\ \forall j \in \mathcal{J} \text{ and } \forall n \in \mathcal{N}. \tag{4e}$$

Thus, problem (2) is rewritten as an Integer Linear Problem (ILP) optimization problem, which can be solved by standard methods presented in the literature such as the BB and Branch and Cut (BC) [1], [2]. These methods can be directly applied and have much lower average complexity than the brute force solution, i.e., the complete enumeration of all possible assignments.

V. Proposed solution

In this section, in order to better understand our proposed solution, a brief review of RL, including the techniques Q-learning and deep RL, is first described.

A. An Overview of Reinforcement Learning

1) Q-Learning technique: In RL, an agent interacts with the surrounding environment in order to learn an optimal policy or an optimal path to a given goal. The learning is done by trial and error, where the agent gets a reward for each taken action [11]. Thus, we define $\mathcal{S}$ as the set composed by all states, which is responsible for characterizing the environment of the agent and $\mathcal{A}(s)$ is defined as the set of actions per state in which each action represents the changes that the agent applies to this environment.

The well-known Q-learning algorithm can be used to obtain an optimal policy that maximizes the long-term expected accumulated discounted rewards [11]. In this algorithm, the agent observes the surrounding environment and an action is taken by an agent according to a particular strategy or decision policy. This strategy or policy can be implemented using a variety of techniques such as the ϵ-greedy decision policy. Particularly, the ϵ-greedy policy consists in selecting a greedy action[1] with probability $1-\epsilon$ or a random action with probability ϵ. In addition, ϵ decreases with time. The advantage of this policy is that it allows agents to explore through random action selection in order to make better action selections in the future, while avoiding to get stuck at non-optimal policies by greedy action selections [7]. Indeed, one of the challenges that arise in RL techniques is the trade-off between *exploration* and *exploitation* and it should be carefully balanced so that the benefits of both can be properly harvested [11].

Once taken an action $a \in \mathcal{A}(s)$, the system state changes from s to s' and this change generates a signal or indicator that evaluates the effect of the taken action. This feedback or message from the environment is called *reward*, ϕ, which is a numerical score and it is used to estimate the expected value of taking an action a in a particular state s, also known as Q-value of a state/action (s,a). In detail, the Q-value is calculated by a Q-function such that $Q : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ and, for a given state/action pair (s,a), it can be estimated according to Bellman's equation [11]:

$$\hat{Q}(s,a) = (1-\alpha)\hat{Q}(s,a) + \alpha(\phi + \gamma \max_{a'} \hat{Q}(s',a')), \tag{5}$$

where $0 < \alpha \leq 1$, $0 \leq \gamma < 1$ are constants called learning rate and discount factor, respectively, and $\max_{a'} \hat{Q}(s',a')$ is the best estimated Q-value given the next state s' and all possible actions at s'. Basically, α determines how quickly the learning process occurs, while γ controls the value placed on future Q-values. Then, over several iterations the state/action pairs are defined and their respective Q-values are estimated and updated by Bellman's equation. A set of these iterations from an *initial state* s_o to a *final state* s_f is called an *episode*.

In its classical form, Q-learning algorithm constructs a lookup table, Q-table, as a surrogate of the optimal Q-function. At the beginning, the lookup table is initialized with arbitrary values. Next, it is iteratively updated, following the steps described before, in order to find the optimal policy. According to [11], the Q-learning algorithm converges to an optimal policy if each action in the action space is executed under each state for an infinite number of times on an infinite run and the learning rate α decays appropriately. However, the Q-learning algorithm is suitable only when the state-action space is small. The reason is that the storage of the lookup table related to (5) becomes impractical as the state-action space increases and several states will be rarely visited, consequently creating holes in the lookup table.

2) Deep Q-Learning technique: Although the Q-learning technique is very simple, it is a quite powerful algorithm to create an interesting set of experience or a kind of cheat-sheet for the agent. Indeed, this is fundamental and helps the agent to figure out exactly which action to perform until it converges to an optimal policy. Nevertheless, as highlighted in [13], Q-learning has two serious problems: (1) the amount of memory required to save and update the Q-table can increase exponentially as the number of states and actions increases, (2) many states are rarely visited and, consequently, the amount of time required to explore all these possibilities (state/action pairs) in order to create a good estimate for Q-table would be unrealistic or impractical in a real setting [14].

As a result, producing and updating a Q-table can become ineffective in large-sized environments, i.e., with a large number of states and actions. Notwithstanding, these limitations can be solved with the emerging deep RL, e.g., deep Q-learning, which is considered as a promising technique to solve the complex control issues, especially for the high-dimension solutions [16]. Basically, this technique can be cast as an extension of classical Q-learning algorithm that uses DNN to approximate the Q-function in lieu of a lookup table.

In specific, in the deep Q-learning algorithm, a DNN called Deep Q-Network (DQN) is defined as a parameterized value function $Q_{\boldsymbol{\theta}} : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ that is used to estimate the

[1] A greedy action on state s is given by $a = \underset{\forall a\in\mathcal{A}(s)}{\arg\max}\, Q(s,a)$

Q-function, where the state is given as its input, the Q-value of all possible actions is generated as its output and, finally, $\boldsymbol{\theta}$ represents its parameters that define the Q-values. Therefore, the key idea of deep Q-learning technique is that the function $Q_{\boldsymbol{\theta}}$ is completely determined by $\boldsymbol{\theta}$. Consequently, the task of finding the best Q-function is essentially limited to the search for these best parameters of finite dimensions [14].

Although the algorithmic and statistical properties as well as the performance of the classical Q-learning algorithm are well known and studied, the same is not true for deep Q-learning, which still remains less well-understood in theory. Thereby, the idea of simply approximating the Q-function by a DNN can often lead to learning instability, e.g, the Q-function may be over-optimistically estimated. Fortunately, these instabilities can be greatly reduced by the following two aspects [19], [20]. Firstly, similar to classical Q-learning, the agent interacts with its environment following an ϵ-greedy policy over $\mathcal{S} \times \mathcal{A}$. However, the experiences, composed each one of them of the current state (s), action (a), reward (ϕ), and the next state (s'), obtained by agent are gathered in a memory $\mathcal{D}$ with limited capacity D. In addition, the DQN training is performed on a mini-batch $\mathcal{B}$ of B tuples or experiences selected randomly from $\mathcal{D}$. Such method is known as *memory replay*. This strategy can reduce the correlation among the training examples, which ensures that the optimal policy cannot be driven to a local minimal [20].

Moreover, to increase stability and reduce correlations in the system, the second aspect consists in using two DQNs with the same architecture. Therefore, we define the target DQN, $Q_{\boldsymbol{\theta}_{\text{target}}}$, with parameters $\boldsymbol{\theta}_{\text{target}}$ and the training DQN, $Q_{\boldsymbol{\theta}_{\text{train}}}$, with parameters $\boldsymbol{\theta}_{\text{train}}$. The training DQN is responsible for learning the values of $\boldsymbol{\theta}_{\text{train}}$, while the target DQN is used to take actions. The values of $\boldsymbol{\theta}_{\text{target}}$ are updated at every τ iterations and they are set to be equal to $\boldsymbol{\theta}_{\text{train}}$. Put another way, the weights $\boldsymbol{\theta}_{\text{target}}$ are fixed for a number of iterations while the weights $\boldsymbol{\theta}_{\text{train}}$ are constantly updated. This strategy is commonly called Double DQN (DDQN) and it has shown improvements in learning process compared to single DQN strategy [7]. Therefore, for each episode, the least squares loss of the training DQN for a random mini-batch $\mathcal{B} \subset \mathcal{D}$ with B samples is

$$\Omega(\boldsymbol{\theta}_{\text{train}}) = \sum_{(s,a,\phi,s') \in \mathcal{B}} (y - Q_{\boldsymbol{\theta}_{\text{train}}}(s,a))^2, \tag{6}$$

where the target is

$$y = \phi + \gamma \max_{\forall a'} Q_{\boldsymbol{\theta}_{\text{target}}}(s', a'). \tag{7}$$

Finally, the loss function (6) is minimized by a stochastic gradient algorithm in order to train the mini-batch $\mathcal{B}$. Then, the train DQN updates its parameters with the new parameters provided by training. According to [14], the convergence to a set of good parameters occurs quickly.

B. Proposed Multi-Agent Deep Q-learning Solution

In this section, we present a multi-agent DQN-based dynamic resource allocation framework to solve problem (2).

Algorithm 1 Set reward value

Require: R_j and ξ_j, $\forall j \in \mathcal{J}$;
1: $\phi \leftarrow \sum_{j \in \mathcal{J}} R_j$ and $\vartheta \leftarrow 0$;
2: **for** $l \in \mathcal{L}$ **do**
3: **if** $\sum_{j \in \mathcal{J}_l} u(R_j, \xi_j) < \eta_l$ **then**
4: **for** $j \in \mathcal{J}_l$ **do**
5: **if** $R_j < \xi_j$ **then**
6: $\vartheta \leftarrow \vartheta + (R_j - \xi_j)/\xi_j$;
7: **end if**
8: **end for**
9: **end if**
10: **end for**
11: **if** $\vartheta < 0$ **then**
12: $\phi \leftarrow \vartheta / \phi$;
13: **end if**;
14: **return** ϕ;

Firstly, we define the concept of agent, state, action, and reward for this approach.

- **Agents**: we propose a multi-agent deep reinforcement learning scheme with each RB as an agent. As a result, there are N agents in this approach.
- **Action of agent** n: consist of choosing a UE j, i.e., an action $a^{(n)} = j$ means that RB n is assigned to UE j. A tuple or vector $\mathbf{a}$, composed of elements $a^{(n)}$, therefore, means a given assignment pattern or association among UEs and RBs.
- **State of agent** n: we describe the state of agent n, $s^{(n)}$, as a composition of two important aspects for an agent. In the first part, we consider a piece of information common to all agents. This information consists in an N-tuple $\mathbf{a}$ which represents a possible assignment for the system. On the other hand, in the second part, we have specific information related to agent n. Thus, the second part of the state of agent n is composed by a J-tuple, $\mathbf{u}$, where $u_j^{(n)} = \gamma_{j,n}$, $\forall j$, i.e., each agent or RB n knows the SNR value for all users of the system.
- **Reward**: obviously, the reward function should be designed to maximize the objective (2a) of problem (2). Thus, to do that we use Alg. 1. The main idea of this algorithm is to define a reward value, ϕ, capable of reporting what is possible to achieve in terms of satisfaction and system throughput for a given assignment. This value tries to measure how close one is from meeting the requirements of problem (2), without disregarding its objective function. Note that if all constraints of problem (2) are met, meaning that the chosen assignment is a feasible solution of problem (2), then ϕ is equal to $\sum_{j \in \mathcal{J}} R_j$, which is the objective function (2a). Otherwise, ϕ is equal to $\vartheta / \sum_{j \in \mathcal{J}} R_j$. The variable ϑ is responsible for quantifying how close the chosen assignment is to a feasible solution of problem (2). Notice that if any constraint is not met, then ϑ is negative, and this represents a punishment or a negative reward.

Our proposed solution based on deep Q-learning to problem (2) is shown in Alg. 2. Basically, the idea of this algorithm is to use the concepts of DQN and those defined in Section V-A to approximate Q-table by a function and, therefore, avoid the main disadvantages of the proposal addressed in [13], such as high memory cost.

Regarding Alg. 2, in lines 1, 2 and 3 we define the main structures for our approach. More specifically, in line 1, we

Algorithm 2 Deep Q-learning based Resource Assignment

1: Initialize replay memory $\mathcal{D}$, γ and τ;
2: Initialize the training DQN, $Q_{\boldsymbol{\theta}_{\text{train}}}$, with random weights $\boldsymbol{\theta}_{\text{train}}$;
3: Initialize the target DQN, $Q_{\boldsymbol{\theta}_{\text{target}}}$, with weights $\boldsymbol{\theta}_{\text{target}} \leftarrow \boldsymbol{\theta}_{\text{train}}$;
4: **loop** over the episodes
5: Observe current state of all agents $s^{(n)}$;
6: Each agent n chooses an action $a^{(n)}$ using ϵ-greedy policy from $Q_{\boldsymbol{\theta}_{\text{target}}}$;
7: Execute the action of each agent, i.e., $\mathbf{a} \leftarrow [a^{(1)}, \cdots, a^{(N)}]$;
8: Obtain ϕ using Alg. 1;
9: Observe the next state of each agent ($s'^{(n)}$);
10: Store experience $(s^{(n)}, a^{(n)}, \phi, s'^{(n)})$ of each agent in $\mathcal{D}$;
11: Sample a set of random experiences, i.e., a mini-batch $\mathcal{B}$ from $\mathcal{D}$;
12: Perform the gradient descent step on (6) with respect to the weights $\boldsymbol{\theta}_{\text{train}}$;
13: At every τ episodes replaces target parameters, i.e., $\boldsymbol{\theta}_{\text{target}} \leftarrow \boldsymbol{\theta}_{\text{train}}$;
14: Update the ϵ-greedy decision policy;
15: $s^{(n)} \leftarrow s'^{(n)}, \quad \forall n \in \mathcal{N}$;
16: **end loop**;
17: **return** $\mathbf{a}$;

reserve a limited amount of memory, $\mathcal{D}$, and define a certain number of iterations, τ, that represent a period for updating target DQN weights. In lines 2 and 3, we randomly initialize the target and training DQNs responsible for the learning process, where both DQNs are fully-connected DNNs that consists of four layers: an input layer, an output layer and two hidden layers in between. The input layer is fed by the state vector of length $(N + J)$ and the output layer has dimensionality corresponding to the number of possible actions, in this case, J.

The *loop* between lines 4 and 16 represents the learning process, which is responsible for adjusting the weights of training and target DQNs, where each iteration is defined as an episode [2] . We assume an approach in which the actions are taken in parallel by each agent while the training is performed by a central module according to Fig. 1. In this figure, on one hand, it can be seen that the decisions of the N agents can be chosen in parallel, i.e., at the same time, using a DQN whose input and output depend on the current states and possible actions of each agent, respectively. However, on the other hand, the training phase is centralized and, therefore, the experiences of all agents are constantly collected to adjust the weights of another DQN. This framework eases implementation and improves stability. Moreover, this strategy can also significantly reduce the amount of memory and computational resources required by training [14]. Therefore, each agent has the same copy of $Q_{\boldsymbol{\theta}_{\text{target}}}$, while $Q_{\boldsymbol{\theta}_{\text{train}}}$ is localized at the central module. Thus, in each episode, all agents observe their respective states and are synchronized to take their actions at the same time based on ϵ-greedy policy from $Q_{\boldsymbol{\theta}_{\text{target}}}$, according to lines 5 and 6. Next, an assignment pattern, $\mathbf{a}$, is defined from the agent's actions, the reward, ϕ, is calculated and each agent n observes its next state according to lines 7, 8 and 9, respectively. Observe that a user can be assigned to one or more RBs. Therefore, there is no problem when two or more RBs are simultaneously associated to the same user. However, if the minimum guarantees of satisfaction are not fulfilled, the agents' actions receive a negative reward in order to improve their actions. In addition,

[2]In this article, we assume that the terms "episode" and "iteration" are interchangeable in the text.

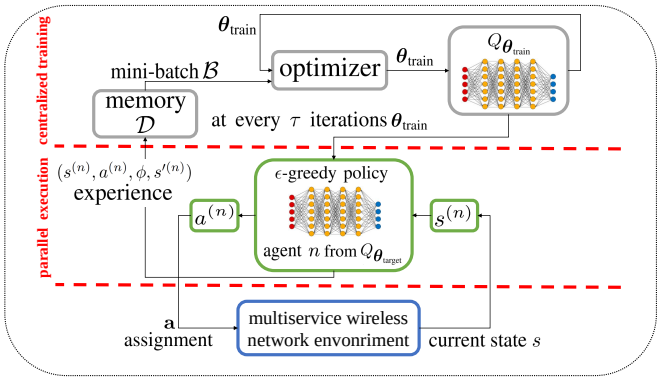


Fig. 1. Illustration of the proposed multi-agent deep RL algorithm for resource allocation [14], adapted.

the reward is a common value for all agents, which is obtained by Alg. 1 at each episode, so that they can benefit from each other's experiences to try to learn the optimal policy. This is interesting because it directs the agents to take actions that are good for the entire system rather than good actions individually. Consequently, the result over time is an efficient assignment in terms of system throughput. In other words, the agents work collaboratively to maximize the obtained reward and, therefore, the objective in (2a). Moreover, the reward can be computed by a central module and the calculated value is used for each agent in its respective tuple of experience.

After that, in line 10, we define an experience sample as a tuple, $(s^{(n)}, a^{(n)}, \phi, s'^{(n)})$, consisting in current state $s^{(n)}$, chosen action $a^{(n)}$, reward ϕ and next state, $s'^{(n)}$ of each agent. In addition, in order to avoid oscillations and divergence in the parameters, we use the concept of memory replay so that the tuples of experiences of all agents are stored in memory $\mathcal{D}$. We consider that this memory is a First In First Out (FIFO) queue where a new experience replaces the oldest experience in the queue when the number of experiences exceeds the capacity, D. In order to train the parameters $\boldsymbol{\theta}_{\text{train}}$, a mini-batch of experiences, $\mathcal{B}$, is sampled randomly from $\mathcal{D}$ and the stochastic gradient descent method is performed by central module to minimize the cost function in (6) as shown in lines 11 and 12, respectively. Furthermore, the process of updating the parameters $\boldsymbol{\theta}_{\text{target}}$ is periodic and, therefore, in line 13, only at every τ episodes, the new parameters $\boldsymbol{\theta}_{\text{train}}$ are available for target DQN. Finally, in line 14, the ϵ-greedy policy is updated, the current state of each agent changes to the next state (line 15) and another episode starts.

Mathematically, the complexity of Alg. 2 can be evaluated by quantifying the complexity to obtain the Q-function from the DQN and to train the weights of the DQN since this is the main idea of this algorithm. Obviously, it highly depends on the structure of the employed DQN and its parameters. As discussed, in our case, the DQN is composed by fully-connected layers and, thereby, the complexity of the algorithm is given by $\mathcal{O}(wm \log m)$ where w is the number of layers and m is the number of units per layer [21].

Something interesting about Alg. 2 is that depending on the initialization of the DQNs weights or parameters, the algorithm can converge to a solution more or less accurately relative to the optimal solution of problem (2), given a fixed

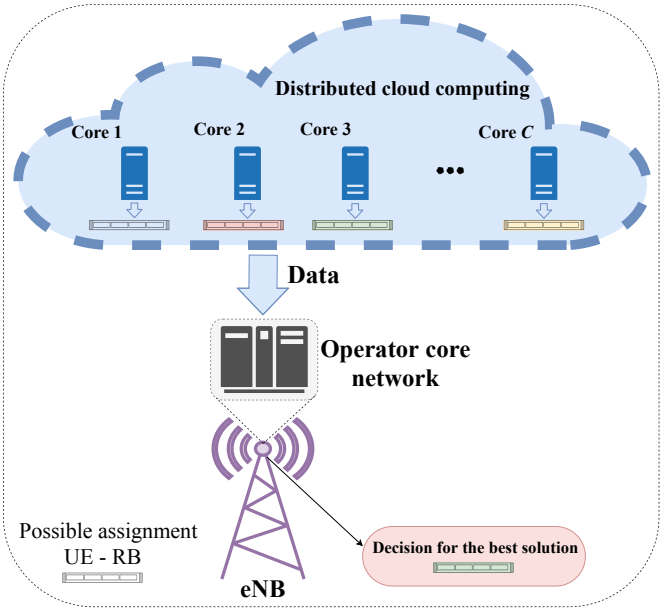


Fig. 2. Proposed solution to problem (2) using parallel execution of Alg. 2 on different cores.

number of episodes. Indeed, this can be exploited by letting Alg. 2 run multiple times on different cores and, therefore, with totally independent weights initialization. As a result, since the runtime of each core is the same, the idea is to choose the best output as a solution to problem (2) as depicted in Fig. 2. Note that parallel execution of multiple cores does not necessarily need to be computed on the eNB itself. Due to possible limitations of this infrastructure such as overhead, small storage space and low computing ability, the data storage and processing can be moved to decentralized and powerful computing platforms located in a cloud. In terms of performance, the cloud utilizes distributed system architectures and can offer excellent computation speeds. Besides, cloud computing provides many other advantages as quick deployment, easy integration, resiliency, redundancy, backup, disaster recovery, among others [22]. Thus, the eNB is limited to deciding the best solution after data processing.

VI. Performance Evaluation

In this section, we evaluate our proposed solution and compare it with the optimal solution and with the solutions of [1], [2] and [13]. We firstly present the main simulations parameters and, after that, the results and their discussion.

A. Simulation Assumptions

We consider 6 RBs ($N = 6$), 4 UEs ($J = 4$), 2 service plans ($L = 2$) and we admit that UEs from service plan 2 demand a throughput of 150 kbps higher than the UEs from the service plan 1. In both services, we consider only two UEs, where $\eta_1 = 2$ and $\eta_2 = 1$. We assume 11 QoS levels in kbps such that $\xi_{j\in\mathcal{J}_1} = (150, 220, \ldots, 850)$, i.e., the required data rates for service plan 1 vary between 150 kbps and 850 kbps at the step of 70 kbps. Consequently, the requirements for service plan 2 vary between 300 and 1000 with the same step. In our simulations, we also assume a full buffer model that is

TABLE I
DQN Parameters

Parameter	Value
Memory size (D)	1000
Mini-batch size (B)	256
Number of neurons per hidden layer	64
Initial value for ϵ	0.8
Decay rate	0.001
Learning rate (α)	0.0001
Discount factor (γ)	0
Period for updating target DQN weights (τ)	5

TABLE II
Network Parameters [1]

Parameter	Value
Cell radius	334 m
Transmit power per RB	0.35 W
Number of subcarriers per RB	12
Shadowing standard deviation	8 dB
Path loss	$35.3 + 37.6 \cdot \log(d)$ [dB]
Noise spectral density	$3.16 \cdot 10^{-20}$ W/Hz

characterized by two facts: the number of UEs in the cell is constant and the UEs' transmit buffers always have unlimited amount of data to transmit. Due to its simplicity, this type of traffic model has been widely adopted in OFDMA-based simulations [23].With respect to DQN, it was implemented using Tensorflow [24], assuming two hidden layers. We use the rectifier linear unit (ReLU) as DQN's activation function and we use Adam's algorithm [25] for the optimization. Moreover, we consider that ϵ-greedy policy varies over the episodes following an exponential decay. In general, all the important simulation parameters are shown in Table I and Table II.

To perform qualitative comparisons with our proposed algorithm (deep Q-RA), we simulate the optimal solution of problem (2) as well as the algorithms Reallocation-based Assignment for Improved Spectral Efficiency and Satisfaction (RAISES) [1], Rate Maximization under Experience Constraints (RMEC) [2] and Q-learning based Resource Assignment (Q-RA) [13]. On one hand, RAISES and RMEC are traditional rule-based algorithms, which use resource reallocation strategies to define the best assignment pattern for the system. On the other hand, Q-RA, as its name suggests, is an algorithm based on Q-learning technique for resource allocation. Therefore, Q-RA algorithm is a tabular learning method, where a single agent accumulates all its experience in a Q-table over several episodes.

Regarding the performance metrics, we consider the outage rate and the system throughput. An outage event happens when an algorithm cannot manage to find a feasible solution, i.e., the algorithm does not find a solution fulfilling the constraints of problem (2). Then, outage rate is defined as the ratio between the number of instances with outage events and the total number of simulated instances. The system throughput is the sum of the data rates obtained by all the UEs in a given instance. The results were obtained by running $1,000$ feasible instances of problem (2) in order to get valid results in a statistical sense and the channel realizations were the same for all the simulated algorithms to get fair comparisons.

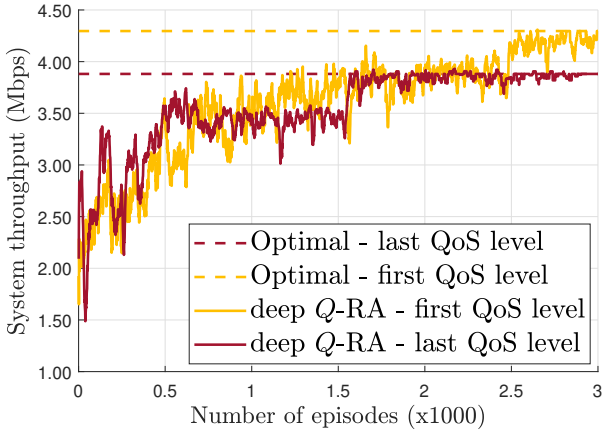


Fig. 3. System throughput versus number of episodes in a particular instance for optimal and deep Q-RA algorithms.

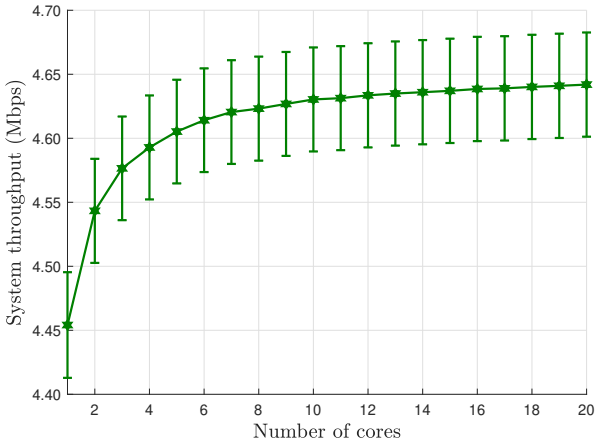


Fig. 4. System throughput of deep Q-RA algorithm versus the number of parallel cores in the system.

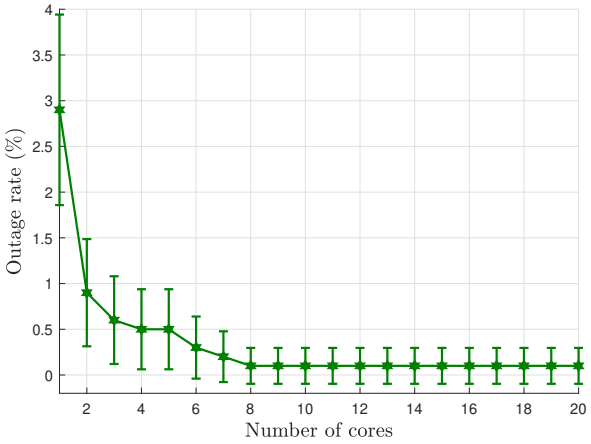


Fig. 5. Outage rate of deep Q-RA algorithm versus the number of parallel cores in the system.

B. Numerical Results

Fig. 3 shows the system throughput versus the number of episodes for the algorithms optimal and deep Q-RA, condesidering the first and the last QoS levels described in Section VI-A. Looking at the performance of the deep Q-RA solution, we can observe that it converges to the optimal solution as the number of episodes increases for both investigated QoS levels. This is an expected result since the more episodes we have, the more accurate the estimation of Q-function is and, consequently, the more favorable it is for the agents to converge to the optimal solution of problem (2). Moreover, note that in Fig. 3 the convergence time to the optimal solution may vary depending on the required QoS level. This is because at low QoS levels there are several possible solutions and, as a result, it can be more difficult to converge to the optimal solution of problem (2). For scenarios with high QoS levels required, possible solutions are rarer but once found means near optimal solutions, consequently the deep Q-RA algorithm tends to focus on them. Indeed, this can lead to faster convergence.

In Fig. 4 and Fig. 5, we plot the system throughput and outage rate versus the number of parallel cores in the system, respectively, in order to show the advantages of the structure illustrated in Fig. 2. Also, from here, we assume for all the following results a confidence interval with a 95% confidence level. Firsty, in Fig. 4 and Fig. 5, note that as the number of cores in the system increases, there is a considerable increase in the performance of the proposed solution. In addition, due to the characteristics of the deep Q-learning technique, this structure does not require a high memory consumption and, as shown in the last figures, a relatively low number of cores is enough to ensure excellent performance. Note, for example, that with less than 10 cores there is practically no outage in the system for the investigated scenario.

Now we compare our approach with other proposals from the literature. In Fig. 6 and Fig. 7, we plot the system throughput and the outage rate in the considered scenario versus the QoS level for the algorithms optimal, RAISES, RMEC, Q-RA and deep Q-RA, respectively. For the Q-RA and deep Q-RA algorithms, we consider $3,000$ episodes in the plots of these figures. Besides, for deep Q-RA algorithm we use 10 cores. In this way, we firstly observe a near optimal performance of our proposed solution both in terms of outage rate and system throughput to problem (2). In fact, we highlight Fig. 7 that shows the outage curve that is considerably better for the solutions based on RL, with even better performance for deep Q-RA solution. In this figure, notice that the outage rate for these solutions are smaller than 1% and 0.1% for Q-RA and deep Q-RA, respectively.

On the other hand, RAISES and RMEC solutions have much higher outage rates, with approximately 7% and 10% for the highest QoS level, respectively. This shows that solutions based on ML algorithms may perform better than traditional heuristics and, therefore, they can be considered as a promising tool to solve resource allocation problems in modern networks. However, as highlighted in [13], Q-RA solution may require a high memory cost to build and store Q-table because it directly depends on space $\mathcal{S} \times \mathcal{A}$. Therefore, this makes its use more difficult in interesting and realistic scenarios. As discussed earlier, this is not a problem for deep Q-RA solution, which may in fact become a more attractive and less problematic solution in larger scenarios.

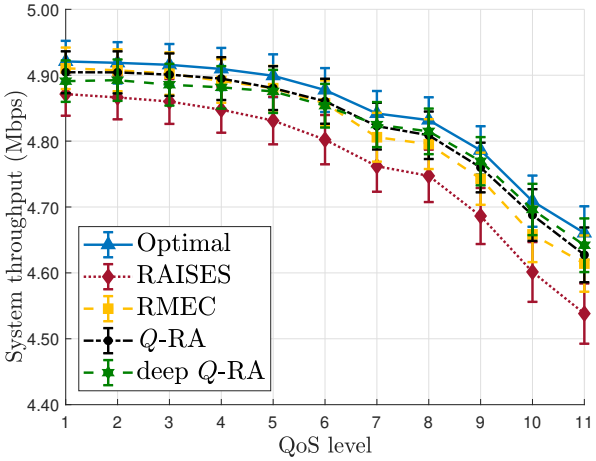


Fig. 6. System throughput versus QoS level for optimal, RAISES, RMEC, Q-RA and deep Q-RA algorithms in the considered scenario.

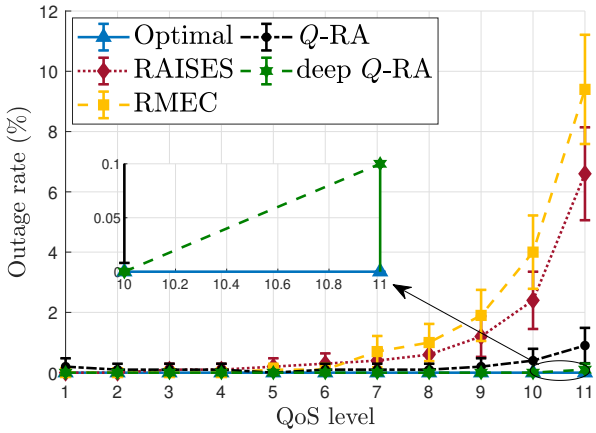


Fig. 7. Outage rate versus QoS level for optimal, RAISES, RMEC, Q-RA and deep Q-RA algorithms in the considered scenario.

VII. Conclusions and Perspectives

In this paper, we have investigated the problem of maximizing the system throughput subject to user satisfaction ratio constraints in a multiservice scenario. This problem was previously studied in [1], [2] and [13], where tradicional heuristics or machine learning based methods were proposed. However, to tackle this problem we have proposed a new decentralized radio resource allocation mechanism employing multi-agent deep reinforcement learning. From the simulation results, we have shown that each agent can learn how to jointly deal with resource allocation and QoS garantees while maximizing the system throughput. As a result, our proposed can provide better performance than the other benchmark approaches simulated in this article.

Regarding future works, we believe that the proposed framework in this paper can be improved by taking into consideration the channel correlation along the time and redefining the system state in order to considerably decrease the need for training when applied in dynamic contexts. Finally, other approaches where learning-based techniques are jointly responsible for allocating power and resource can also be analyzed in the future.

References

[1] F. R. M. Lima, T. F. Maciel, W. C. Freitas, and F. R. P. Cavalcanti, "Resource assignment for rate maximization with QoS guarantees in multiservice wireless systems," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 3, pp. 1318–1332, Mar. 2012, ISSN: 0018-9545. DOI: 10.1109/TVT.2012.2183905.

[2] D. A. Sousa, V. F. Monteiro, T. F. Maciel, F. R. M. Lima, and F. R. P. Cavalcanti, "Resource management for rate maximization with QoE provisioning in wireless networks," *Journal of Communication and Information Systems*, vol. 31, no. 1, Nov. 2016. DOI: 10.14209/jcis.2016.25.

[3] V. F. Monteiro, D. A. Sousa, T. F. Maciel, F. R. P. Cavalcanti, C. F. M. e Silva, and E. B. Rodrigues, "Distributed RRM for 5G multi-rat multiconnectivity networks," *IEEE Systems Journal*, vol. 13, no. 1, pp. 192–203, 2019. DOI: 10.1109/JSYST.2018.2838335.

[4] F. D. Calabrese, L. Wang, E. Ghadimi, G. Peters, L. Hanzo, and P. Soldati, "Learning radio resource management in RANs: Framework, opportunities, and challenges," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 138–145, Sep. 2018, ISSN: 2169-3536. DOI: 10.1109/MCOM.2018.1701031.

[5] K. I. Ahmed, H. Tabassum, and E. Hossain, "Deep learning for radio resource allocation in multi-cell networks," *IEEE Network*, vol. 33, no. 6, pp. 188–195, Nov. 2019, ISSN: 1558-156X. DOI: 10.1109/MNET.2019.1900029.

[6] E. Castaneda, A. Silva, A. Gameiro, and M. Kountouris, "An overview on resource allocation techniques for multi-user mimo systems," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 1, pp. 239–284, 2017. DOI: 10.1109/COMST.2016.2618870.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. DOI: 10.1038/nature14236.

[8] Q. Mao, F. Hu, and Q. H. and, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 4, pp. 2595–2621, 2018, ISSN: 1553-877X. DOI: 10.1109/COMST.2018.2846401.

[9] Y. Sun, M. Peng, and S. Mao, "Deep reinforcement learning-based mode selection and resource management for green fog radio access networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1960–1971, Apr. 2019, ISSN: 2372-2541. DOI: 10.1109/JIOT.2018.2871020.

[10] R. Amiri and H. Mehrpouyan, "Self-organizing mm wave networks: A power allocation scheme based on machine learning," in *Proceedings of the Global Symposium on Millimeter Waves (GSMM)*, May 2018, pp. 1–4. DOI: 10.1109/GSMM.2018.8439323.

[11] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018, ISBN: 978-0-262-91398-6.

[12] J. Pérez-Romero, J. Sánchez-González, R. Agustí, B. Lorenzo, and S. Glisic, "Power-efficient resource allocation in a heterogeneous network with cellular and D2D capabilities," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 11, pp. 9272–9286, Nov. 2016, ISSN: 0018-9545. DOI: 10.1109/TVT.2016.2517700.

[13] J. V. Saraiva, V. F. Monteiro, F. R. M. Lima, T. F. Maciel, and F. R. P. Cavalcanti, "A Q-learning based approach to spectral efficiency maximization in multiservice wireless systems," in *Proceedings of the Brazilian Telecommunications Symposium (SBrT)*, Sep. 2019.

[14] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2239–2250, 2019. DOI: 10.1109/JSAC.2019.2933973.

[15] H. Ye, G. Y. Li, and B. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019, ISSN: 1939-9359.

[16] N. Zhao, Y.-C. Liang, D. Niyato, Y. Pei, M. Wu, and Y. Jiang, "Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 11, pp. 5141–5152, Nov. 2019. DOI: 10.1109/TWC.2019.2933417.

[17] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 31–37, Dec. 2017. DOI: 10.1109/MCOM.2017.1700246.

[18] Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, V. C. M. Leung, and Y. Zhang, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10 433–10 445, Nov. 2017. DOI: 10.1109/TVT.2017.2751641.

[19] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, J. Pineau, *et al.*, "An introduction to deep reinforcement learning," *Foundations and Trends® in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, 2018. DOI: 10.1561/2200000071.

[20] Z. Yang, Y. Xie, and Z. Wang, "A theoretical analysis of deep Q-learning," *arXiv preprint arXiv:1901.00137*, 2019.

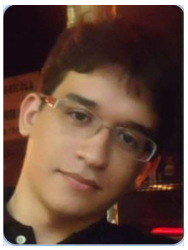
[21] H.-S. Lee, J.-Y. Kim, and J.-W. Lee, "Resource allocation in wireless networks with deep reinforcement learning: A circumstance-independent approach," *IEEE Systems Journal*, pp. 1–04, 2019, ISSN: 2373-7816. DOI: 10.1109/JSYST.2019.2933536.

[22] A. Alzahrani, N. Alalwan, and M. Sarrab, "Mobile cloud computing: Advantage, disadvantage and open challenge," in *Proceedings of the 7th Euro American Conference on Telematics and Information Systems*, New York, NY, USA: Association for Computing Machinery, 2014, ISBN: 9781450324359. DOI: 10.1145/2590651.2590670.

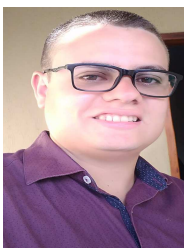
[23] P. Ameigeiras, Y. Wang, J. Navarro-Ortiz, P. E. Mogensen, and J. M. Lopez-Soler, "Traffic models impact on OFDMA scheduling design," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 61, pp. 1–13, Feb. 2012. DOI: 10.1186/1687-1499-2012-61.

[24] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. USENIX Symposium on Operating Systems Design and Implementation*, Nov. 2016, pp. 265–283, ISBN: 978-1-931971-33-1.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/ 1412.6980, 2014. arXiv: 1412 . 6980. [Online]. Available: https://arxiv.org/abs/1412.6980v9.



Juno Vitorino Saraiva received the B.Sc. degree in Computer Engineering, from the Federal University of Ceará (UFC), Sobral, Brazil, in 2017. He received his M.Sc. degree in Electrical and Computer Engineering with emphasis in Telecommunications from the UFC, Sobral, Brazil, in 2019. Since 2018, he has been a researcher at the Wireless Telecom Research Group (GTEL), UFC, participating of projects in a technical and scientific cooperation with Ericsson Research. His research interests include radio resource allocation algorithms and learning-based techniques for QoS guarantees in scenarios with multiple services, resources, antennas and users.



Iran Mesquita Braga Junior received the B.Sc. degree in Computer Engineering and M.Sc. degree in Electrical and Computer Engineering, from the Federal University of Ceará (UFC), Sobral, Brazil, in 2017 and 2019, respectively. He is currently pursuing D.Sc.degree in Telecommunications Engineering at UFC, Fortaleza, Brazil and, since 2018, he has been a researcher at the Wireless Telecom Research Group (GTEL), UFC, participating of projects in a technical and scientific cooperation with Ericsson Research. His research interests include radio resource management, machine-learning techniques, numerical optimization and multiuser/multiantenna communications.

Victor Farias Monteiro received the double B.Sc. degree in General Engineering, from the École Centrale Lyon, France, and in Telecommunications Engineering (magna cum laude), from the Federal University of Ceará (UFC), Fortaleza, Brazil, in 2013. In 2015 and 2019, he received the M.Sc. and PhD degrees in Telecommunications Engineering from UFC, respectively. He is currently a Postdoctoral researcher at the Wireless Telecommunications Research Group (GTEL), UFC, where he works in projects in cooperation with Ericsson Research. In 2016, he was an invited researcher at Ericsson Research in Lulea, Sweden, where he worked for 6 months on the topic of LTE-NR Dual Connectivity. Besides, in 2017/2018, he spent one year at Ericsson Research in Stockholm, Sweden, where he worked on the topics of mobility management and channel hardening. From 2010 to 2012, he took part, in France, of the Eiffel Excellence Scholarship Programme, established by the French Ministry of Foreign Affairs. His research interests include machine learning, 5G architecture and protocols, 5G measurement and reporting procedures, mobility management and radio resource allocation.

Francisco Rafael Marques Lima received the B.Sc. degree with honors in Electrical Engineering in 2005, and M.Sc. and D.Sc. degrees in Telecommunications Engineering from the Federal University of Ceará, Fortaleza, Brazil, in 2008 and 2012, re-spectively. In 2008, he has been in an internship at Ericsson Research in Lulea, Sweden, where he studied scheduling algorithms for LTE system. Since 2010, he has been a Professor of Computer Engineering Department of Federal University of Ceará, Sobral, Brazil. Prof. Lima is also a researcher at theWireless Telecom Research Group (GTEL), Fortaleza, Brazil, where he worksin projects in cooperation with Ericsson Research. He has published severalconference and journal articles as well as patents in the wireless telecommunications field. His research interests include radio resource allocation algorithms for QoS guarantees in scenarios with multiple services, resources,antennas and users.

Tarcisio Ferreira Maciel received his B.Sc. and M.Sc. degrees in Electrical Engineering from the Federal University of Ceará (UFC) in 2002 and 2004, respectively, and his Dr.-Ing. degree from the Technische Universität Darmstadt (TUD), Germany, in 2008, also in Electrical Engineering. Since 2001, he has actively participated in several projects in a technical and scientific cooperation between the Wireless Telecom Research Group (GTEL), UFC, and Ericsson Research. From 2005 to 2008, he was a research assistant with the Communications Engineering Laboratory, TUD. Since 2008, he has been a member of the Post-Graduation Program in Teleinformatics Enginnering, UFC. In 2009, he was a Professor of computer engineering with UFC-Sobral and since 2010, he has been a Professor with the Center of Technology, UFC. His research interests include radio resource management, numerical optimization, and multi-user/multi-antenna communications.

Walter C. Freitas Jr. received his PhD degree in Teleinformatic Engineering from Federal University of Ceará (UFC), Brazil in 2006 and his B.S. and M.S. degrees in Electrical Engineering from the same university. During his studies, he was supported by the Brazilian agency FUNCAP and Ericsson. During Q3 of 2015 up to Q2 of 2016, he was a Post-doc Researcher at I3S/CNRS Laboratory, from the University of Nice, Sophia Antipolis, France. During 2005 Walter Freitas Jr. was a senior research of Nokia Technology Institute, He is currently an Assistant Professor with the Department of Teleinformatics Engineering of the Federal University of Ceará and researcher of Wireless Telecom Research Group (GTEL) one of the most important research groups in telecommunication in Brazil. His main area of interest concerns features development to improve the performance of the wireless communication systems, interference avoidance tools, multilinear algebra, and tensor-based signal processing applied to communications.

Francisco Rodrigo Porto Cavalcanti received the D.Sc. degree in Electrical Engineering from the State University of Campinas, São Paulo,Brazil, in 1999. Upon graduation, he joined the Federal University of Ceara (UFC), where he is currently an Associate Professor and holds the Wireless Communications Chair at the Department of Teleinformatics Engineering. In 2000, he founded, and since then has directed, the Wireless Telecom Research Group (GTEL), which is a research laboratory based on Fortaleza, Brazil, which focuses on the advancement of wireless telecommunications technologies. At GTEL, he manages a 19-year long program of research projects in wireless communications sponsored by Ericsson Research. In 2017 he was a visiting researcher to Ericsson´s main site at Stockholm, Sweden. Prof. Cavalcanti has produced a varied body of work including books, papers, patents and software dealing with wireless access networks and technologies. Prof. Cavalcanti is a distinguished researcher of the Brazilian Scientific and Technological Development Council for his technology development and innovation record. He also holds a Leadership and Management professional certificate from the Massachusetts Institute of Technology, Cambridge, USA.