A Q-learning Based Approach to Spectral Efficiency Maximization in Multiservice Wireless Systems

Juno V. Saraiva*†, Victor F. Monteiro†, F. Rafael M. Lima*†, Tarcisio F. Maciel† and F. Rodrigo P. Cavalcanti†

***Abstract*— In this article, we study Radio Resource Allocation (RRA) as a non-convex optimization problem, aiming at maximizing the spectral efficiency subject to satisfaction guarantees in multiservice wireless systems. This problem has already been previously investigated and efficient heuristics have been proposed. However, in order to assess the performance of Machine Learning (ML) 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, our proposal is based on the Q-learning technique, where an agent gradually learns a policy by interacting with its local environment, until reaching convergence. Thus, in this article, the task of searching for an optimal solution in a combinatorial optimization problem is transformed into finding an optimal policy in Q-learning. Lastly, through computational simulations we compare the state-of-art proposals of the literature with our approach and we show a near optimal performance of the latter for a well-trained agent.**

***Keywords*— Radio resource allocation, satisfaction guarantees, machine learning, reinforcement learning, Q-learning.**

I. Introduction

Due to its importance for mobile networks, RRA has attracted much interest from industry and academy over past years [1]. Typically, RRA problems have been proposed in the literature as optimization problems where objective, constraints and resource type to be optimized are specified. In [2] and [3], for example, the authors addressed RRA in order to maximize the system throughput subject to user satisfaction constraints in a multiservice scenario. In order to solve that problem, the authors in [2] and [3] proposed optimal and heuristic solutions. However, not all optimization problems are liable to be optimally solved; in general, non-linear combinatorial problems as in the case of [2] and [3] are hard to solve. Furthermore, some optimal solutions can be found only for small or moderate input size due to their high computational complexity. On the other hand, heuristic solutions are practical methods/algorithms that can be derived from previous experience with similar problems and require much less computational effort than optimal solutions. Nevertheless, heuristics normally do not present any performance guarantee and are problem specific.

ML is a broad area from artificial intelligence and computer science where computer systems perform specific tasks such as solving problems. Successful applications of ML can be found in many areas such as in financial services, marketing, data security and robotics [4]. A branch of ML called RL has gained notoriety since the AlphaGo's victory in the Google DeepMind challenge match in 2016 [1]. In RL, an agent is capable of learning how to behave in an unknown environment so as to achieve a given objective that in general is modeled as the maximization of an expected reward. By taking advantage of the high computing and storage capacity available in current networks, RL is able to deal with many real-world problems due to its capacity to take decisions in complex environment.

Mobile networks are well known for their heterogeneity. Different networks have their own dynamic space-time features and performance requirements. As previously explained, conventional approaches for solving RRA have devised optimal and heuristic solutions based on optimization theory that on their turn are tailored to specific network conditions and user demands. In this sense, the use of ML, and more specifically of RL solutions, to deal with RRA in mobile networks has been considered a very fruitful area. According to [5], RL can efficiently deal with imprecise input data such as the Channel State Information (CSI) that cannot be accurately collected due to random fading. Moreover, RL is capable of processing a huge amount of information and taking good decisions that are of fundamental importance in RRA for modern networks.

One of the most popular RL algorithms is the Q-learning which is a powerful and efficient technique for agents to learn how to act optimally in controlled Markovian domains. This algorithm has been applied in RRA [6]–[8] due to its flexibility and adaptability which are essential features for resource management in current and future mobile networks. The works [6] and [7] addressed Device-to-Device (D2D) communications in heterogeneous networks where the basic purpose was to obtain efficient solutions from the perspective of energy consumption employing Q-learning. Particularly, in [6] the main objective was to minimize the total transmit power, optimizing the connection of the user to the base station. Meanwhile, in [7] Q-learning was used to reduce the total system transmission power by choosing the best user of the system to act as a relay. Both articles showed that their proposals are capable of achieving gains in performance and complexity reduction. Finally, in [8] a self-organizing algorithm was proposed to maximize the sum capacity in a dense mmWave network while providing users with their required Quality of Service (QoS). The results in [8] showed that the proposed algorithm reduces complexity by using a distributed clustering method, and provides adaptability in power allocation by using Q-learning.

*Computer Engineering Dep., Federal University of Ceará, Sobral, Brazil.
†Wireless Telecommunications Research Group (GTEL), Federal University of Ceará, Fortaleza, Brazil. Emails: {juno, victor, rafaelm, maciel, rodrigo}@gtel.ufc.br. T. F. Maciel was supported by CNPq under the grants 426385/2016-0 and 308621/2018-2. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. This study was also supported by the Ericsson Research, Sweden, and Ericsson Innovation Center, Brazil, under UFC.47 Technical Cooperation Contract Ericsson/UFC.

Motivated by the fact that ML based approaches have been employed successfully to solve optimization problems in order to achieve improved RRA in mobile networks and by the rapid advances in this area, we provide a Q-learning algorithm for RRA mobile networks and compare it with the algorithms in [2] and [3]. The RRA problem involves the optimization of Resource Block (RB) assignment under QoS and user satisfaction constraints.

II. System Modeling

We admit a Single Input Single Output (SISO) downlink cellular system composed of a number of sectored cells so that in a given sector there are J User Equipments (UEs) connected to an Evolved Node B (eNB). UEs are grouped in set $\mathcal{J} = \{1, \ldots, J\}$.

We combine Orthogonal Frequency Division Multiple Access (OFDMA) and Time Division Multiple Access (TDMA) where its resources are arranged in a frequency-time resource grid such that, in the frequency domain, a RB consists of a group of adjacent subcarriers, while, in the time domain, a slot consists in a number of consecutive Orthogonal Frequency Division Multiplexing (OFDM) symbols. The time duration of a slot corresponds to one Transmission Time Interval (TTI). We consider that an RB and a slot are the minimum scheduling units in the frequency and time domain, respectively. RBs grouped in set $\mathcal{N} = \{1, \ldots, N\}$.

Regarding intra-cell interference, in a given cell sector, each RB $n \in \mathcal{N}$ is assigned to only one UE, therefore, there is no intra-cell interference. Furthermore, regarding inter-cell interference, we assume that it is added to the thermal noise in the Signal to Noise Ratio (SNR) expression, defined later.

In this work, we also assume a multiservice scenario with L service plans contained in the set $\mathcal{L} = \{1, \ldots, L\}$ and supported by the system operator. In each TTI, the J UEs compete for the available RBs in order to meet their throughput requirements, which are defined by their service plans. Each service plan $l \in \mathcal{L}$ requires a minimum number of UEs 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., $\mathcal{J}_{l_1} \cap \mathcal{J}_{l_2} = \emptyset$, $\forall l_1, l_2 \in \mathcal{J}$ and $l_1 \neq l_2$.

The SNR $\gamma_{j,n}$ of UE $j \in \mathcal{J}$ in RB $n \in \mathcal{N}$ is $\gamma_{j,n} = (p_n \cdot \alpha_j \cdot |h_{j,n}|^2)/(\sigma^{\text{RB}})^2$, where p_n is the transmit power allocated to the UE j on RB n; α_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 frequency response (fast fading) of RB n when assigned to UE j; and, finally, $(\sigma^{\text{RB}})^2$ is the noise power at the receiver in the bandwidth of a given RB.

Similar to [2] and [3], 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.

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 $r_{j,n} = f(\gamma_{j,n})$.

III. Problem Formulation and Optimal Solution

As discussed previously, the problem investigated herein is the one from [2] and [3], whose aim is 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., $R_j = \sum_{n \in \mathcal{N}} r_{j,n} x_{j,n}, \ \forall j \in \mathcal{J}$. Therefore, the resource assignment problem can be formulated as:

$$\max_{\mathbf{x}} \sum_{j \in \mathcal{J}} R_j, \tag{1a}$$

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

$$\sum_{j \in \mathcal{J}_l} u(R_j, \xi_j) \geq \eta_l, \ \forall l \in \mathcal{L}, \tag{1c}$$

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

where $\mathbf{x}$ is the vector of optimization variables and $u(x, b)$ in (1c) denotes the Heaviside step function, which assumes the value 1 if $x \geq b$ and 0, otherwise. With this, η_l is the minimum number of UEs from service l that should be satisfied and ξ_j represents the required throughput for a UE to be considered satisfied, i.e., ξ_j consists of a QoS requirement for each UE j in terms of throughput. Regarding constraints (1b) and (1d), they guarantee that each RB is assigned to a single UE. Notice that (1) is a combinatorial optimization problem with a non-convex constraint (1c). In order to simplify the optimal solution analyses, we linearize equation (1c) by introducing some new variables. Let ρ_j be a binary selection variable that assumes the value 1 if UE j is selected to be satisfied and 0, otherwise. Thus, (1c) can be replaced by (2c) and (2d), where $\rho_j = 1$ in (2d) implies that UE j is satisfied and (2d) means that for all service l there are at least η_l satisfied UEs. Hence, problem (1) can be equivalently reformulated as shown in (2).

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

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

$$R_j \geq \xi_j \cdot \rho_j, \ \forall j \in \mathcal{J}, \tag{2c}$$

$$\sum_{j \in \mathcal{J}_l} \rho_j \geq \eta_l, \ \forall l \in \mathcal{L}, \tag{2d}$$

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

Thus, we have transformed (1) into an Integer Linear Problem (ILP), which can be solved by standard methods such as the Branch and Bound (BB) algorithm [2], [3].

IV. Proposed Solution

In this section, we first present an overview of the Q-learning technique and, after that, we present our proposed solution to problem (1).

A. An Overview of Q-Learning

In general, the Q-learning model consists of an *agent*, a set of *states* $\mathcal{S}$ and a set of *actions* per state $\mathcal{A}(s)$, $s \in \mathcal{S}$. By performing actions and, consequently, transitions from state to state, the agent aims to learn an optimal policy or an optimal path to a given goal. Each of these states can be defined as a tuple of values that characterizes the environment for the agent, while each action represents the change that the agent applies to this environment. Thereby, the idea is that the agent perceives the environment state and selects an action according to a particular strategy or decision policy [4]. This strategy or policy can be implemented using a variety of techniques such as the ϵ-greedy decision policy. It is simple, but very efficient: the agent *explores* or *exploits* its environment taking random (non-greedy) or greedy actions according to a given probability distribution, respectively. Normally, for this decision policy, a random action can be chosen with probability $\epsilon \in (0, 1)$, while a greedy action is taken with a probability $1-\epsilon$. In this way, when greedy actions are taken, the objective is to exploit the acquired knowledge to improve performance. However, generally, for a (partially-)unknown environment, these actions lead to locally optimal solutions. On the other hand, when the agent decides to take random actions it explores its environment for the sake of acquiring experience and knowledge about the environment and, therefore, there is no concern with the immediate effects of these actions. However, random actions allow the agent to neglect the locally optimal policies, and to acheive the globally optimal one, instead. Consequently, 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 harnessed [4].

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 $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ and, for a given state/action pair (s, a), it can be estimated according to Bellman equation: $\hat{Q}(s, a) = (1 - \alpha)\hat{Q}(s, a) + \alpha(\phi + \gamma \max_a \hat{Q}(s', a))$, 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 equation. A set of these iterations from an *initial state* s_o to a *final state* s_f is called an *episode*. Thereby, each state/action pair and its respective performed action allow the agent to interact with the environment and this interaction after several episodes produces precious information about the consequences of actions, mainly about what to do or not in order to achieve goals. This information are precisely represented in the Q-values and the set of all of them is stored in the Q-table, which is where all the experience or knowledge acquired by the agent is concentrated.

Algorithm 1 Q-learning Based Resource Assignment

$\epsilon \leftarrow 1$; Q-table $\leftarrow \emptyset$; ▷ initialization
loop over the episodes ▷ learning process
  $s \leftarrow s_\text{o}$ and $\Omega \leftarrow \emptyset$; ▷ where s_o is an initial state
  while $s \neq s_\text{f}$ **do** ▷ where s_f is a final state
    Generate a random number υ between 0 and 1;
    if $\upsilon > \epsilon$ **then**
      choose action $a \leftarrow \arg\max_{a \in \mathcal{A}(s)} Q(s, a)$; ▷ exploitation
    else
      choose action $a \in \mathcal{A}(s)$ randomly; ▷ exploration
    end if
    Define the pair (s, a) (state/action);
    Execute a, i.e., $s \leftarrow s'$ (next state);
    if $(s, a) \notin Q$-table **then**
      $\Omega \leftarrow \Omega \cup \{(s, a)\}$; ▷ (s, a) is a new pair
    end if
  end while
  Set $Q(s_\text{f}, \cdot)$ using Alg. 2;
  for $(s, a) \in \Omega$ **do**
    if $(s, a) \notin Q$-table **then**
      $Q(s, a) \leftarrow Q(s_\text{f}, \cdot)$;
    else
      $Q(s, a) \leftarrow \max\{Q(s, a), Q(s_\text{f}, \cdot)\}$;
    end if
  end for
  Update ϵ-greedy decision policy; ▷ ϵ value is decreased
end loop; **return** s;

Algorithm 2 Set Q-value

Require: (s, a) and $R_j, \forall j \in \mathcal{J}$;
$Q(s, a) \leftarrow \sum_{j \in \mathcal{J}} R_j$ and $\vartheta \leftarrow 0$;
for $l \in \mathcal{L}$ **do**
  if $\sum_{j \in \mathcal{J}_l} u(R_j, \xi_j) < \eta_l$ **then**
    for $j \in \mathcal{J}_l$ **do**
      if $R_j < \xi_j$ **then**
        $\vartheta \leftarrow \vartheta + (R_j - \xi_j)/\xi_j$;
      end if
    end for
  end if
end for
if $\vartheta < 0$ **then**
  $Q(s, a) \leftarrow \vartheta / Q(s, a)$;
end if; **return** $Q(s, a)$;

Therefore, the basic idea in Q-learning is that the agent finds and learns an optimal policy for the desired problem by benefiting from the experience gathered in the Q-table. However, when the space of states and actions is large, building an efficient Q-table would require many iterations or a lot of training time, as well as a high amount of memory. Nevertheless, learning from interaction is a paramount concept and it is present in almost all the techniques of learning [4].

B. Proposed Q-Learning Solution

Our proposed solution based on Q-learning for problem (1) is depicted in Alg. 1. The agent, the states and the actions associated with our Q-learning model are defined as:

- **Agent:** defined as an entity located in the eNB which is able to sense the state of its environment and to take actions that affect this state.
- **State:** defined as an N-tuple where the n-th element, s_n, defines the UE $j \in \mathcal{J}$ that will get assigned RB $n \in \mathcal{N}$.

In other words, if $s_n = j$ and $j \neq 0$, then $x_{j,n} = 1$ and $x_{i,n} = 0, \forall j \neq i$. We assume that if $s_n = 0$ it means that RB n is available to be allocated. Our initial state is defined as $s_n = 0, \forall n \in \mathcal{N}$. Meanwhile, we assume as a final state s_f any state where all RBs are allocated, i.e., $s_n \in \mathcal{J}, \forall n \in \mathcal{N}$.

- **Action:** consists of choosing a UE j and a not yet allocated RB n to be assigned to the selected UE, i.e., to set $s_n = j$.

Thereby, in Alg. 1 the basic idea is that through the main loop from lines 2 to 26 the agent learns an optimal policy for RRA problem in (1). This loop corresponds to the learning process and each iteration of it we define as our episode. Therefore, in each of these episodes, the *while* loop from lines 4 to 16 runs until the initial state s_o becomes any state that is considered a final state s_f, i.e., the *while* loop runs until all RBs are allocated. This change of state is iteratively done by taking actions. The actions are chosen in lines 5 to 10 following an ϵ-greed policy.

Once decided which RB should be allocated to a UE or given that an action was chosen, the state/action pair is defined in line 11 and that action is executed in line 12, enjoining the change of state. Thereafter, if the state/action pair in line 11 has been defined for the first time, i.e., if it is not stored in Q-table yet, then it is stored in Ω, which stores all of these new pairs for each episode. Next, in line 17, the Q-value for the last state/action pair of the current episode is calculated. This procedure is executed by Alg. 2.

The main idea in Alg. 2 is to define a Q-function capable of reporting what is possible to achieve in terms of satisfaction and system throughput for a given pair (s, a). The Q-function tries to measure how close one is from meeting the requirements of (1), without forgetting its objective function. If all constraints of problem (1) are met, meaning that s is a feasible solution of problem (1), then the Q-function is equal to $\sum_{j\in\mathcal{J}} R_j$, which is the objective function (1a). Otherwise, the Q-function is equal to $\vartheta / \sum_{j\in\mathcal{J}} R_j$. The variable ϑ is responsible for quantifying how close the current state s is to a feasible solution of problem (1). Notice that if any contraint is not met, then ϑ is negative, and the Q-function as well.

Continuing Alg. 1 from line 18, before ending the current episode, we run the *for* loop from lines 18 to 24, where back-propagation is used to set the Q-values of all new state/action pairs defined in the current episode. Its main idea is to inform the agent, when visiting an already known state/action pair, what is the maximum Q-value that one can get from that state in the end of the episode. This avoids having to set the values of ϕ, α and γ present in Bellman equation. In addition, this procedure speeds-up the learning process.

In line 25, the ϵ value is decreased and, consequently, greedy actions will be chosen with a higher probability in the next episodes. The main idea is to explore more in the initial episodes and exploit more in the final ones. Thus, we adapt the traditional ϵ-greedy decision policy by appropriately reducing ϵ value during the learning process. The main advantage of this is to avoid loss of performance of Alg. 1 due to the constant exploration probability. Lastly, in line 26 the current episode ends and another starts, but with an agent interacting with a less uncertain environment.

V. Simulation Results

In this section, we evaluate our proposed solution and compare it with the optimal solution and with the solutions of [2] and [3]. We firstly present the main simulations parameters and, after that, the results and their discussion.

A. Simulation Parameters

We consider $N = 6$, $J = 4$, $L = 2$ and we admit that UEs from service 2 demand a throughput of 150 kbps higher than the UEs from the service 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 $\xi_{j\in\mathcal{J}_1} = (150, 220, \ldots, 850)$.

Important parameter related to the adopted channel model are presented in Table I. Due to the limit of space here, please refer to [2] for further details regarding the adopted simulation models.

To perform qualitative comparisons with our proposed algorithm (PROP), we simulate the optimal algorithm of problem (1) (OPT) as well as Reallocation-based Assignment for Improved Spectral Efficiency and Satisfaction (RAISES) [2] and Rate Maximization under Experience Constraints (RMEC) [3] algorithms.

Regarding the performance metrics, we consider the system throughput and the outage rate, i.e., percentage of cases in which (1c) was not satisfied for at least one of the services.

The results were obtained by running $8,000$ feasible instances of problem (1) 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.

TABLE I
Simulation Parameters [2]

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	$35.3 + 37.6 \cdot \log_{10}(d)$[dB]
Noise spectral density	$3.16 \cdot 10^{-20}$W/Hz

B. Discussion

Fig. 1 shows the system throughput as well as individual throughputs of the UE 01 and UE 02 versus the number of episodes for the algorithms OPT and PROP. Looking at the performance of the PROP solution, we can observe that it tends to the OPT solution as the number of episodes increases. This result is expected since the more episodes, the more precise the Q-table becomes and, consequently, the more favorable it is for the agent to achieve the optimal solution of problem (1).

In Fig. 2 and Fig. 3 we plot the outage rate and the system throughput in the considered scenario versus the QoS level for the algorithms OPT, RAISES, RMEC and PROP, respectively. For the PROP algorithm we vary the number of episodes and we consider $1,000$ and $3,000$ of them in the plots of these figures. In this way, we firstly observe the better performance of our proposed solution both in terms of outage rate and system throughput in relation to the other sub-optimal solutions to solve problem (1). Besides, we highlight Fig. 2 that shows the outage curve that is considerably better for the

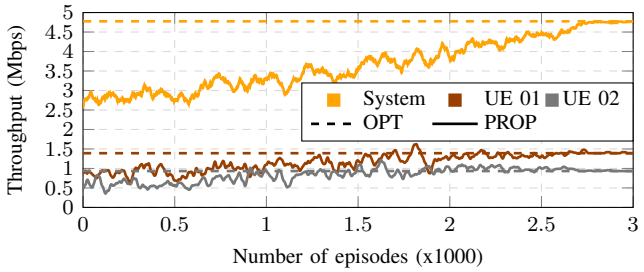

Fig. 1

SYSTEM THROUGHPUT AND INDIVIDUAL THROUGHPUTS OF UE 01 AND UE 02 VERSUS NUMBER OF EPISODES IN A PARTICULAR INSTANCE FOR OPT AND PROP ALGORITHMS.

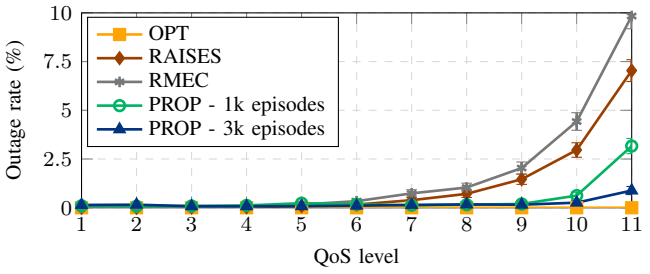

Fig. 2

OUTAGE RATE VERSUS QOS LEVEL FOR OPT, RAISES, RMEC AND PROP ALGORITHMS IN THE CONSIDERED SCENARIO.

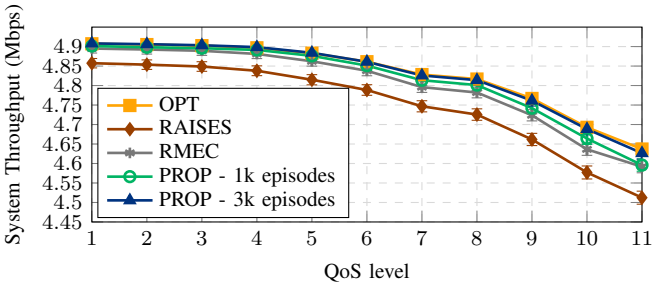

Fig. 3

SYSTEM THROUGHPUT VERSUS QOS LEVEL FOR OPT, RAISES, RMEC AND PROP ALGORITHMS IN THE CONSIDERED SCENARIO.

solution based on Q-learning. In this figure, notice that the outage rate for that solution is smaller than 1% when it is set to $3,000$ episodes, while for RAISES and RMEC solutions we have approximately 7% and 10% in outage, respectively. Even with $1,000$ episodes we have a significant gain on these solutions with only 3.1% in outage. This shows that solutions based on ML algorithms may perform better than traditional heuristics and, therefore, it may be appealing to optimization problems in the context of RRA.

However, as discussed in Section IV-A, in the case of the Q-learning technique, the convergence for the optimal solution may require high amounts of memory due to the cost of building and storing the Q-table, especially if the number of state/action pairs is large. Indeed, this does not occur with RAISES and RMEC algorithms that are of low computational cost. On the other hand, to build the Q-table its memory requirement is an array of states $\times$ actions and, for our proposed solution notice that the cardinality of the set of states is $|\mathcal{S}| = J^N$, while for the set of actions in a given state s is $|\mathcal{A}(s)| = J(N - \sum_{n\in\mathcal{N}} x_{s_n,n})$, which means that considering scenarios with larger scales makes the application of Q-learning technique troublesome. Nevertheless, an alternative in order to reduce the excessive memory used by the look-up table representation in the Q-learning technique is to use a deep neural network to approximate the Q-function. Integrating neural networks with RL techniques can be quite promising and has exhibited excellent performance in the literature [4].

VI. CONCLUSIONS

In this article, 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 [2] and [3], where efficient heuristics were proposed. To tackle this problem again we have proposed a new approach employing a reinforcement learning technique, specifically based on Q-learning. The simulations results presented in this article showed that our proposed algorithm performs near optimally and, consequently, better than the conventional heuristics proposed in [2] and [3], especially in terms of outage. However, as discussed previously, the Q-learning technique requires high amounts of memory when the space of states and actions is large and this may considerably limit its applications. Despite this, it is possible to reduce this cost by training a neural network to estimate the Q-table and, although this needs to be further analyzed, it is an interesting topic of future research.

REFERENCES

[1] 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 Commun. Mag.*, vol. 56, no. 9, pp. 138–145, Sep. 2018, ISSN: 2169-3536. DOI: 10.1109/MCOM.2018.1701031.

[2] 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 Trans. Veh. Technol.*, vol. 61, no. 3, pp. 1318–1332, Mar. 2012, ISSN: 0018-9545. DOI: 10.1109/TVT.2012.2183905.

[3] 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," *J. of Commun. and Inf. Syst.*, vol. 31, no. 1, Nov. 2016. DOI: 10.14209/jcis.2016.25.

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

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

[6] Y.-F. Huang, T.-H. Tan, Y.-L. Li, and S.-C. Huang, "Performance of resource allocation for D2D communications in Q-learning based heterogeneous networks," in *Proc. of the IEEE Internat. Conf. on Consum. Electron. - Taiwan (ICCE-TW)*, May 2018, pp. 1–5. DOI: 10.1109/ICCE-China.2018.8448975.

[7] 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 Trans. Veh. Technol.*, vol. 65, no. 11, pp. 9272–9286, Nov. 2016, ISSN: 0018-9545. DOI: 10.1109/TVT.2016.2517700.

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