Pacheco-Paramo, DF.; Tello-Oquendo, L.; Pla, V.; Martínez Bauset, J. (2019). Deep Reinforcement Learning Mechanism for Dynamic Access Control in Wireless Networks Handling mMTC. Ad Hoc Networks. 94:1-14. https://doi.org/10.1016/j.adhoc.2019.101939

Deep Reinforcement Learning Mechanism for Dynamic Access Control in Wireless Networks Handling mMTC ☆

Diego Pacheco-Paramo[a,*], Luis Tello-Oquendo[b], Vicent Pla[c], Jorge Martinez-Bauset[c]

[a] *Escuela de Ciencias Exactas e Ingeniería, Universidad Sergio Arboleda, Bogotá 111221, Colombia*
[b] *College of Engineering, Universidad Nacional de Chimborazo, Riobamba 060108, Ecuador*
[c] *Instituto ITACA. Universitat Politècnica de València, Valencia 46022, Spain*

Abstract

One important issue that needs to be addressed in order to provide effective massive deployments of IoT devices is access control. In 5G cellular networks, the Access Class Barring (ACB) method aims at increasing the total successful access probability by delaying randomly access requests. This mechanism can be controlled through the barring rate, which can be easily adapted in networks where Human-to-Human (H2H) communications are prevalent. However, in scenarios with massive deployments such as those found in IoT applications, it is not evident how this parameter should be set, and how it should adapt to dynamic traffic conditions. We propose a double deep reinforcement learning mechanism to adapt the barring rate of ACB under dynamic conditions. The algorithm is trained with simultaneous H2H and Machine-to-Machine (M2M) traffic, but we perform a separate performance evaluation for each type of traffic. The results show that our proposed mechanism is able to reach a successful access rate of 100 % for both H2H and M2M UEs and reduce the mean number of preamble transmissions while slightly affecting the mean access delay, even for scenarios with very high load. Also, its performance remains stable under

☆ The research of D. Pacheco-Paramo was supported by Universidad Sergio Arboleda, P.I. Tecnologías para la inclusión social y la competitividad económica. O.E.6.

* Corresponding author
Email address: diego.pacheco@usa.edu.co (Diego Pacheco-Paramo)

the variation of different parameters.

Keywords: access control, deep reinforcement learning, massive machine type communications, 5G, cellular networks.

1. Introduction

The Internet of Things (IoT) is an ubiquitous network of interconnected objects that harvest information from diverse environments, interact with the physical world, and use existing Internet infrastructure to provide services for information transfer, analytics, and applications. IoT devices come with vastly different computing capabilities, energy constraints, and radio access technologies. Therefore, IoT architectures need to efficiently handle a massive number of devices with reliability, and real-time capabilities. One of the leading facilitators of the IoT environment is machine-type communication (MTC) [1, 2]. Unlike H2H communications, MTC introduces new traffic patterns: typically, they produce small reports, either periodical or triggered by specific events. Distinct peculiarities of MTC traffic require specialized and inter-operable communication technologies [3, 4]. Cellular networks are the natural choice to satisfy these requirements and handle a significant part of this emerging traffic [5]. Features such as an already existing infrastructure, extensive area coverage, and high-performance capabilities have allowed cellular-based MTC growing at a break-neck pace enabling an entirely new class of ubiquitous applications and services, such as smart metering, e-healthcare, intelligent transportation, environmental monitoring, among others [6].

The unique characteristics of massive MTC (mMTC) pose significant challenges from the networking point of view, even for state-of-the-art technologies, such as those found in 5G and beyond [4]. Most applications and scenarios based on mMTC usually involve a vast number of devices. There, a fundamental issue is the efficient management of network resources in overload situations caused by signaling congestion in the random access channel. Indeed, while the number of connected objects is increasing, current cellular networks may not be

prepared to support this new trend, because they are originally designed for a different traffic type.

The 3GPP suggests access class barring (ACB) [7, 8] as one mechanism to address the congestion in cellular networks. ACB enables barring UEs probabilistically, according to a barring rate and a barring time; the base station broadcasts these parameters. Given the complexity to find the optimal configuration for those parameters in scenarios with variable traffic, we developed a real-time configuration selection scheme based on a reinforcement learning (RL) approach to dynamically tune the barring rate. The proposed scheme can rapidly react to traffic changes using local information available at the base station. Our experiments are based on realistic traffic behavior by making use of traces from cellular network operators to enhance the access control of simultaneous H2H and M2M traffic in cellular networks.

We closely follow the 3GPP directives so that our proposed solution conforms the specifications of the random access procedure in current network deployments, and therefore can be successfully integrated into the system. The main contributions of this paper are summarized as follows

- We contrive a deep reinforcement learning algorithm for dynamic access control by tuning the ACB barring rate to reduce the congestion on the Random Access Channel (RACH). We rely on the generalization ability of deep neural networks to enhance the adaptation of the barring rate to new traffic conditions.

- We evaluate the performance of our solution in terms of network key performance indicators under a wide range of MTC access request intensities following the traffic model suggested by the 3GPP for this kind of studies, while considering realistic H2H traffic by using real traces from cellular network operators. We determine the optimal parameters for the deep reinforcement learning solution.

- We compare the deep reinforcement learning algorithm with a Q-Learning based solution and the well-known D-ACB dynamic solution under differ-

ent traffic conditions [9]. We show how our new proposal is able to reduce the energy consumption of M2M devices while allowing that all UEs access the system successfully.

The remainder of this paper is organized as follows. In Section 2, we conduct a literature review of different studies that evaluate the performance of ACB and the proposed algorithms for its dynamic operation. Then, in Section 3 we describe the LTE-A random access procedure detailing the resources available for the UEs in the initial access to the network, the four-way message handshake, and the backoff procedure. In Section 3.1, we present the ACB mechanism suggested by the 3GPP, and detail how it works and how UEs get the barring parameters. In Section 4, we introduce two different ACB schemes based on reinforcement learning to tackle the dynamic tuning of the barring rate according to the ever-changing network traffic. Next, in Section 5, we first describe scenarios where we evaluate the proposed reinforcement learning based ACB schemes; then, we test and analyze their sensibility to different parameters and network conditions, showing the most relevant results in terms of key performance indicators (KPIs). Also, we compare the reinforcement learning based solutions with a well-known dynamic solution named D-ACB [9]. Finally, in Section 6 we draw the conclusions.

2. Related Work

The use of data-analysis tools to enhance the performance of cellular networks is a trend that has received a lot of attention in recent years. This approach seems intuitive considering the large amounts of data that Telcos need to collect in order to provide services. These databases are composed of call detail records (CDRs) that provide different types of information about the usage of the network, and they have been used to provide different applications. For example, in [10] the authors propose a location management mechanism that aims to reduce the signaling associated to location updates by predicting the future locations of UEs. In [11], an energy-reduction mechanism is proposed

based on the usage patterns of base stations in India. This solution exploits temporal changes in the occupation of the base stations, by adapting the power consumption of antennas during the day. In other works such as [12, 13], the relation between network performance and QoE is analyzed through the analysis of data obtained from Telcos.

Regarding the optimization of access control mechanisms in LTE/LTE-A, it is a well studied problem that has gained interest due to the imminent arrival of mMTC, in particular those that support IoT applications. In LTE-A, the most studied access control mechanisms is ACB, and different solutions have been proposed based on it [9, 14, 15, 16, 17, 18]. Some of these schemes can achieve near-optimal performance (i.e., maximize the resources utilization and guarantee the access of the vast majority of the UEs to the base station). However, such a high performance has been obtained by assuming an idealized ACB operation and by setting exact barring parameters, for which complex processes and questionable assumptions are often needed. Consequently, these ACB schemes cannot be implemented in current cellular systems.

In [9], a mechanism to set ACB in real-time without any *a priori* knowledge of the contending UEs in the network is proposed, and its performance resembles the optimal. However, in this solution the delay is reduced by assuming that UEs retry on the following random access opportunity (RAO) after a collision, which is not the procedure specified in the standards. In [15] an access scheme with priorities is proposed and evaluated against ACB and Extended Access Barring (EAB). However, it is assumed that the access control parameters, such as P_{ACB}, are updated in every access opportunity, without any constraints. On the contrary, our solution considers these restrictions. In [19], a mechanism that dynamically adapts ACB based on an estimation of the number of contending UEs is proposed. Furthermore, the authors improve the estimation using a Kalman Filter. However, in this work only M2M UEs are considered, which does not allow an understanding of how the mechanism affects H2H UEs.

It is also possible to find in the literature reinforcement learning solutions that aim to optimize the access control for wireless networks, and in particu-

lar for cellular networks such as LTE-A and NB-IoT. In our work in [20], we proposed a Q-Learning solution that was able to adapt to dynamic traffic conditions and it was compared against static solutions. In this work we present a scheme that improves all the relevant KPIs, with respect to our previous proposal. In [21], the authors propose a Q-Learning mechanism that aims to assign preambles to H2H or M2M UEs according to the traffic intensity. However, their scheme requires that the system knows how much traffic per UE type is offered in order assign access priorities, in a similar fashion as Extended Access Barring. In [22], a Q-Learning approach is proposed, where each M2M UE has to learn when to transmit. This mechanism does not make use of ACB and it is completely decentralized. Note that decentralized access schemes do not adhere to current LTE-A recommendations. Likewise, in [23] the authors propose a decentralized mechanism to optimize the access control. In this case, the UEs are able to learn the characteristics of different coexisting medium access control mechanisms, and adapt their transmissions using cognitive radio. Although this scenario is very promising, it heavily relies on the processing capacities of the terminals, which is not feasible in scenarios with low-power, low-processing devices such as those frequently found in IoT applications. In [24] a deep reinforcement learning mechanism for controlling ACB is proposed and evaluated with both M2M and H2H traffic. The results show that their mechanism is able to effectively improve the probability of successful access and the delay. However, they assume that the base station has a complete knowledge of the number of UEs contending for resources in the network. The same assumption is done in [25], where a reinforcement learning approach is proposed to control ACB in real-time. This assumption is not realistic. However, in our work we consider that the base station only knows the number of UEs whose preambles were successfully decoded.

Learning Automata [26] is another approach that has been used to enhance the access of massive M2M communications in wireless networks. In [27], the authors propose a mechanism that allows efficient multi-hop communications in wireless sensor networks. In this solution, the objective is to reduce data losses

associated to congestion, which also reduces energy consumption. In [28], the authors propose a learning automata solution to adapt the ACB mechanism in LTE-A networks. This solution relies exclusively on the information that is available at the base station, and is able to adapt the barring probability so the number of successfully decoded preambles on each access opportunity is maximized. Therefore, it aims at maximizing the throughput of access requests on the system, reducing the total delay. However, this behavior can also increase the number of collisions, and as a consequence, the total number of retries and the energy consumption. This problem is more intense as the number of users grow, for example in scenarios with mMTC. In [29], a learning automata based solution is proposed to enhance the access control in LTE-A with mMTC. In this case, the learning automata is used to estimate the number of users that should contend for resources on the system after applying the ACB mechanism in order to maximize the system throughput. Then, the system adapts the barring probability according to the information that is available at the base station, adhering in this way to the LTE-A specifications. Although this solution is also able to enhance the rate of accepted preambles, it also increases the mean number of retries which is undesirable for energy-constrained devices. Also, this will have a negative impact on scenarios with a maximum number of retries, as it can diminish the successful access probability. In our approach, we aim at minimizing the number of collisions, so the mean number of retries is minimized while increasing the successful access probability.

3. Random Access Procedure and Access Control Scheme

In LTE-A, when a UE wants to access the cellular network, it performs a random access procedure. The random access channel (RACH) is used to signal the connection request; it is allowed in predefined time/frequency resources, hereafter random access opportunities (RAOs) [30, 31]. The base station has a number of preambles available for initial access to the network. These preambles are generated by Zadoff-Chu sequences due to their good correlation prop-

erties [31, 32] and are transmitted by the UEs when attempting to access the network.

There are two different forms of the random access procedure in LTE-A: contention-free and contention-based. The former is used for critical situations such as handover, downlink data arrival or positioning, in which the access procedures can be initiated by the base station. The latter, which is the concern of this paper, is the standard mode for network access; it is employed by UEs to change the radio resource control state from idle to connected, to recover from a radio link failure, to perform uplink synchronization or to send scheduling requests [30].

A four-message handshake is performed in the contention-based random access. In *Msg1*, a UE transmits a randomly chosen preamble from the preamble pool during one of the available RAOs. A preamble will be detected at the base station if it has not been chosen by more than one UE in the same RAO. Otherwise, a collision occurs. Then, the base station sends a random access response message, *Msg2*, which includes one uplink grant for each detected preamble. *Msg2* is used to assign time-frequency resources to the UEs for the transmission of *Msg3*. UEs wait for a predefined time window to receive the uplink grant. If no uplink grant is received by the end of this window and the maximum number of access attempts has not been reached, the UEs wait for a random time and then perform a new access attempt. That is, they select a new preamble and transmit it at the next RAO. The UEs that receive an uplink grant send their connection request message, *Msg3*, using the resources specified by the base station. Finally, the base station responds to each *Msg3* transmission with a contention resolution message, *Msg4*. The interested reader is referred to [33, 34, 30, 7, 35] for further details.

In the following, we present access class barring [7, 8] as one mechanism suggested by the 3GPP to address the congestion in cellular networks.

3.1. Access Class Barring

Access Class Barring (ACB) is a congestion control scheme designed to limit the maximum number of UEs that simultaneously access the network. The main purpose of ACB is to randomly delay the number of access requests and in this way help to cope with massive temporary surges. Note that massive-synchronized accesses demands to the PRACH might jeopardize the accomplishment of QoS objectives. If ACB is not implemented, all ACs are allowed to access the PRACH. When ACB is implemented, the base station broadcasts (through SIB2) a barring rate P_{ACB} and a mean barring time T_{ACB} that are applied to all ACs 0-9. Note that ACB is applied only to the UEs that have not yet initiated its random access procedure as explained in Section 3. UEs subject to the ACB scheme must perform a barring check as described in Algorithm 1 [7, 8] before initiating the random access procedure.

Algorithm 1: ACB Scheme

repeat
  Set the mean barring time $T_{\text{ACB}}(n)$ and the barring rate $P_{\text{ACB}}(n)$ broadcast by the base station in the nth SIB2;
  Generate $\mathcal{U}[0,1)$ = a random number with uniform distribution between 0 and 1;
  if $\mathcal{U}[0,1) \leq P_{ACB}(n)$ **then**
    initiate the random access procedure;
  else
    Generate a new $\mathcal{U}[0,1)$;
    Set the barring time as

$$T_{\text{barring}} = [0.7 + 0.6\,\mathcal{U}[0,1)]\; T_{\text{ACB}}(n); \tag{1}$$

    wait for T_{barring};
  end
until *the random access procedure is initiated*;

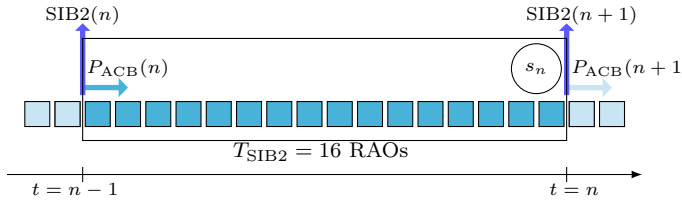

Figure 1: State definition and SIB2.

4. Implementation of Reinforcement Learning Mechanisms

In this section, we describe the implementation of two different ACB mechanisms. The first one is based on classical Q-Learning [36], an off-policy algorithm that uses estimates in order to calculate the actions that optimize the long term rewards of the system. The second one is based on Double Deep Q-Learning [37], which uses a deep neural network in order to generate a function that assigns Q values to states.

The classical Q-Learning mechanism implemented in this paper was originally presented in [20], although some of its characteristics were not evaluated due to lack of space. The Double Deep Q-Learning mechanism implemented is a new contribution of this paper, and it shares with the former implementation the state space and the reward function. This allows a fair comparison of both solutions. In the following, we explain how the Markov Decision Process is defined, with its corresponding actions, rewards, and state space. Then, we will explain how the learning mechanisms were implemented.

4.1. System Model

One of the most important characteristics of many centralized access control problems is that the access controller does not know how many users are contending for its resources, nor their access times. In LTE-A, the base station is able to identify the number of successfully received preambles N_{psu} in a specific RAO. However, due to collisions, interference, decoding problems or other impairments, the value of N_{psu} normally differs from the total number of preambles sent, especially in [heavily] congested scenarios. Consider-

ing that the base station uses ACB as its access control mechanism, the selection of P_{ACB} will be the action used by the learning scheme to optimize the performance of the network. Since there are 16 possible values of P_{ACB}, $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1\}$, we define the set of actions as $\mathcal{A} = \{1, 2, .., 16\}$.

In LTE-A, the base station informs the UEs about changes on P_{ACB} through SIB2 messages, sent with a periodicity given by T_{SIB2}. Following the specifications [7], in this work we define $T_{\text{SIB2}} = 80\,\text{ms}$. Therefore, T_{SIB2} represents the duration of a decision epoch of our system, where each epoch is composed of one or more RAOs. The number of RAOs that compose a decision epoch is defined according to *prach-ConfigIndex*[31] [7]. In this work, a RAO is generated every 5 ms, and therefore a single decision on P_{ACB} affects 16 RAOs as it can be seen in Fig. 1. Hence, a state has to summarize the situation of the network across all RAOs in an epoch. Thus, we use three variables that properly summarize the traffic on a decision epoch: the mean number of successfully received preambles per RAO ($\overline{N_{psu}}$), the coefficient of variation of the successfully received preambles per RAO ($CV_{N_{psu}}$), and the difference between the mean number of successfully received preambles in the current period and the previous one (ΔN_{psu}). Due to the effect that P_{ACB} has on traffic, we use this value and the three previously mentioned to define the state s. Hence, a state s is defined as $s = \left(\overline{N_{psu}}, CV_{N_{psu}}, \Delta N_{psu}, P_{\text{ACB}}\right)$. In Fig. 1, it can be seen that at time n, the computation of the state value is performed considering the previous 16 RAOs. The number of successfully decoded preambles on each of these RAOs depends, among other factors, on $P_{\text{ACB}}(n)$, which is the action taken on the previous state s_{n-1}.

Bear in mind that the maximum number of UEs that can successfully access the medium in a single RAO is 54, as 54 is the number of available preambles at the base station for contention-based random access [38, 30]. However, due to connectivity issues and the possible selection of the same preamble by different UEs, the effective number of decoded preambles by the base station is much lower. In Fig. 2, the probability mass function (PMF) of successfully

received preambles as a function of the number of sent preambles is shown. It can be seen that when the number of sent preambles is below 10, the base station will successfully receive all sent preambles with high probability. However, as the number of sent preambles increases, the number of successfully received preambles at the base station will considerably decrease with high probability. This reduction of successfully received preambles at the base station is due to preamble collisions. Note that this reduction is more acute when channel impairments are taken into account. As observed in Fig. 2, the probability of receiving more than 29 successful preambles in a single RAO is very low for any number of contending UEs. Therefore, in our system, states where $\overline{N_{psu}} > 29$ are aggregated into the state $\overline{N_{psu}} = 29$. For $CV_{N_{psu}}$, the values were discretized as, $CV_{N_{psu}} \in \{0, 0.2, 0.4, 0.6, 0.8\}$. For example, if $0 \leq CV_{N_{psu}} \leq 0.19$ when measured over the 16 RAOs, the value that will appear on the state is $CV_{N_{psu}} = 0$. In the case of ΔN_{psu}, the results are also modified so it take only 3 possible values. That is, when the observed traffic grows, $\Delta N_{psu} = 1$, when it decreases, $\Delta N_{psu} = -1$, and when it remains constant, $\Delta N_{psu} = 0$.

Finally, the rewards associated to the action taken on a given state s_t, and its resulting state s_{t+1}, are defined by $\mathcal{R}$. In our design, the main objective is to avoid congestion, that is, to control the number of contending UEs through the proper adjustment of P_{ACB}. This allows to reduce the number of transmissions for each UE, an important aspect considering that the maximum number of preamble transmissions according to the standards is 10 [7]. This is a critical constraint for IoT devices that is not considered in other works [9],[29]. Therefore, we have defined a set of general guidelines in order to assign rewards to a group of states:

- A hard limit that allows to minimize the uncertainty about the number of contending UEs due to collisions in the system is 10, according to Fig. 2. Therefore, P_{ACB} should adapt so $\overline{N_{psu}}$ does not surpass this value.
- The specific values of P_{ACB} should be proportional to $\overline{N_{psu}}$ with the ob-

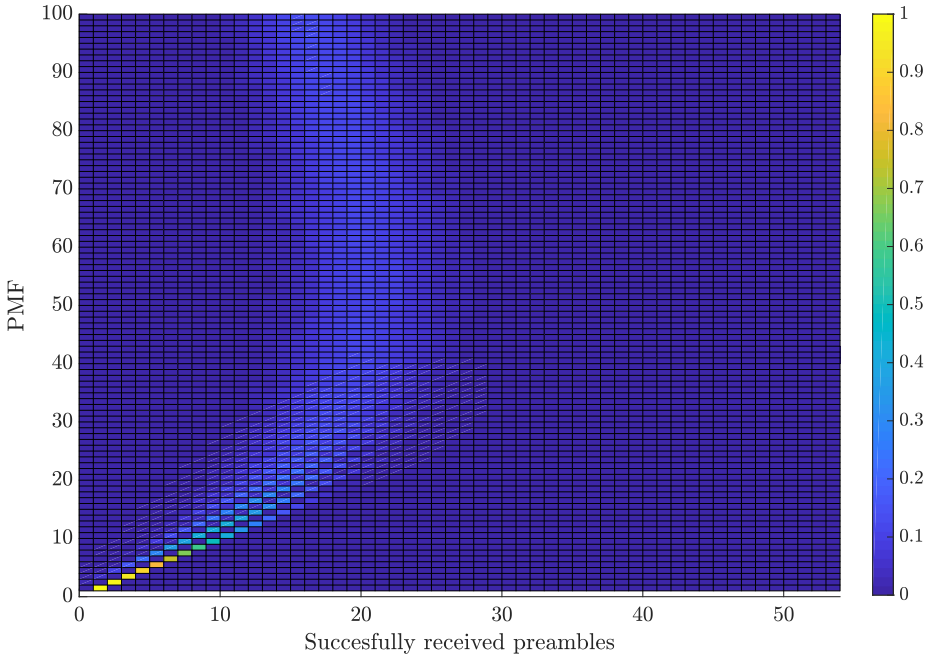

Figure 2: PMF of successfully received preambles as a function of the number of sent preambles.

jective of staying beyond the limit mentioned earlier (i.e., 10).

- If $\overline{N_{psu}}$ is less than 10, a reward or a penalty will be assigned as a function of the traffic changes characterized by $CV_{N_{psu}}$ and ΔN_{psa}. For example, a high $CV_{N_{psu}}$ value can be acceptable if $\overline{N_{psu}}$ is low, but not if it is close to 10. Also, a $\overline{N_{psu}}$ close to 10 can be acceptable if $\Delta N_{psa} = -1$, but not if $\Delta N_{psa} = 1$.

- The reward/penalty values are set in the range between -100 and 100 with step sizes of 20 to facilitate its evaluation. The magnitude of the reward/penalty is associated to how the action helps to avoid/reach the hard limit mentioned earlier.

The reward function defined following these guidelines is shown in [39].

4.2. Classical Q-Learning Mechanism

The implementation of Q-Learning [36] was already shown in [20]. In this case, the update of the reward value associated to the state s given the action a, $Q(s, a)$ is done according to:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[\mathcal{R} + \gamma \max_{a' \in \mathcal{A}} [Q(s',a')] - Q(s,a) \right], \tag{2}$$

where α is the constant learning rate, s' is the following state, a' is the action taken in state s', and $\mathcal{R} + \gamma \max_{a' \in \mathcal{A}} [Q(s', a')]$ is the target. The parameter $\mathcal{R}$ is the reward associated to taking the action a in state s and its values are shown in [39]. The parameter γ is the discount factor which controls the impact of future rewards. Two important results are obtained due to the factor $\max_{a' \in \mathcal{A}}[Q(s', a')]$ in Q-learning. First, that the future rewards are based on an estimate that might suffer a positive bias, leading to sub-optimal performance [40]. Second, that the future rewards are independent of the current policy, hence its off-policy definition. This allows a real-time implementation. The policy used by the system follows an ϵ-greedy approach. The parameter ϵ is the exploration probability, and it affects which action is followed on each state. With probability ϵ the algorithm will choose a random action, that is, it will explore the state space. With probability $1 - \epsilon$ the algorithm will choose the action with the highest $Q(s, a)$ value, that is, it will exploit the maximum reward. Although it has been shown that Q-learning converges [36], its performance might suffer as the state space grows, due in part to the overestimation mentioned earlier [39], but also because of its limited ability to generalize for previously unvisited states. We solved this problem in part in [20] by aggregating states as it was seen in the previous section. However, this does not guarantee that the system performs correctly under previously unseen traffic conditions.

The implementation of Q-learning is shown in Algorithm 2.

Algorithm 2: QL-Based ACB Scheme

Controller: Q-learning($\mathcal{S}, \mathcal{A}, \alpha, \mathcal{R}, \gamma, \epsilon$)

Input : $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, α is the learning rate, $\mathcal{R}$ is the reward, γ is the discount factor, ϵ is the exploration probability

Local : real array $\mathbf{Q}[s, a]$, state s, action a

```
repeat
    if RAO(i) mod T_SIB2 = 0 then
        select action a' from A using the ε−greedy approach;
        observe reward R(s, a', s') and state s';
        update Q(s, a) by (2);
    end
    s = s'
until i = max RAO;
```

4.3. Double Deep Q-Learning Mechanism

Deep Q-Learning [41] can be classified as an approximation method that has been successfully used to solve different problems where a deep neural network is used to represent complex relations of the input data. The neural network is used to approximate Q values using three parameters: s,a and θ, where s is the state, a is the action, and θ represents the weights of the neural network. Just like in the classical Q-learning method, we want to find the Q values that maximize the expected total reward. However, in this case the expected value of the rewards depend not only on s and a, but also on θ. Therefore, every time the system interacts with the environment, the target will be

$$Y = \mathcal{R} + \gamma \max_{a' \in \mathcal{A}} [Q(s', a', \theta)]. \tag{3}$$

According to the Bellman equation, by reaching the target, the system is able to maximize the expected reward. Therefore, after each interaction with the environment, θ should be adapted in such a way that the loss function L is

minimized, where

$$L = \frac{1}{2}\big(\mathcal{R} + \gamma \max_{a' \in \mathcal{A}} \left[Q(s', a', \theta^{-})\right] - Q(s, a, \theta)\big)^2. \tag{4}$$

Note that in (4), the target depends on θ^{-}, that is, the weights of the reference neural network. On the other hand θ represents the weights of the neural network that is currently interacting with the environment. This method has an important advantage over classical Q-Learning, because its neural network allows generalization, which means that the system does not need to interact with every state-action combination to adapt to the environment, and should perform better to previously unseen traffic conditions. However, there are no guarantees on its convergence. In [41], two techniques are proposed to deal with this issue. First, they propose the use of a *target network*. This network is represented in (4) by θ^{-}, but in this case, it is updated every τ iterations. Second, they use *experience replay* [42]. This method allows to break the dependency of consecutive interactions with the environment by recording them on a buffer, and then training the neural network by sampling randomly these experiences. The size of the buffer is defined by ER.

In this work, we have implemented a variation called Double Deep Q-Learning [37]. This algorithm aims to mitigate the overestimation of the Q-values that results from the max operator in (3), that is used both for evaluating and selecting actions. In Double Deep Q-Learning, a second neural network is trained in such a way that the system uses one of them to evaluate the policy, and the other one to choose the action. This is represented in [37] by the Double Deep Q-Learning target

$$Y^{DDQL} = \mathcal{R} + \gamma\, Q(s', \max_{a' \in \mathcal{A}} \left[Q(s', a, \theta)\right], \theta^{'}), \tag{5}$$

where θ refers to the weights of the neural network currently being used online, and $\theta^{'}$ refers to the neural network that will remain unchanged during τ iterations. The implementation of Double Deep Q-Learning is shown in Algorithm 3. We name the two neural networks Q_A and Q_B for clarification purposes. The parameter N_L defines the number of hidden layers of the multi-layer perceptron

implemented. The neural network has four inputs (one for each variable of the state) and 16 outputs (one for each action).

Algorithm 3: Double Deep QL-Based ACB Scheme

Controller: Double Deep Q-learning($\mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma, \epsilon, \tau, ER, N_L$)

Input : $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{R}$ is the reward, γ is the discount factor, ϵ is the exploration probability, ER is the size of the experience replay, τ is the size of the target update and N_L is the number of layers of the neural network

Local : Neural Network $\mathbf{Q}_A[s, a, \theta]$, state s, action a, weights θ

Local : Neural Network $\mathbf{Q}_B[s, a, \theta']$, state s, action a, weights θ'

```
repeat
    repeat
        if RAO(i) mod T_SIB2 = 0 then
            select action a' from A using the ε−greedy approach and Q_A[s, a, θ];
            observe reward R(s, a', s') and state s';
            WRITE a,s,r,s' in buffer
        end
        s = s'
    until j = ER;
    Randomize buffer order
    k=0;
    repeat
        m=kτ+1;
        repeat
            target= r+γ Q_B[s', max_{a∈A}{Q_A[s', a, θ]}, θ'];
            WRITE target in buffer;
        until m = (k+1)τ;
        Q_B[s, a, θ']=Q_A[s, a, θ];
        Read buffer and train Q_A[s, a, θ] to reach target;
    until k = ER/τ;
until i = max RAO;
```

5. Performance Evaluation

In this section, we first describe the conditions of the scenarios where we train and evaluate the proposed ACB adaptation mechanisms; then, we test

and analyze their sensitivity to different parameters and network conditions. Finally, we compare the two proposed mechanisms with a well-known dynamic solution.

Our training and experiments are done on a single cell where both M2M and H2H UEs coexist. The H2H traffic is obtained from Call Detail Records (CDRs) of the Italian Telco Telecom Italia, that provided this information as part of its "big data challenge" in 2014 [43], for the cities of Milan and Trento. This data is aggregated every 10 minutes, and its intensity is provided according to a numerical value whose units are not specified. Therefore, it is necessary to pre-process this data so it can be used in our system. In [44], it is stated that an base station can support 55 EPS Radio Access Bearer setups per second when there is high load. Hence, we use this value as a reference to normalize the data obtained from [43]. Since we want to base our scenario on real data as much as possible, we do not perform any interpolations, although traffic is only obtained every 10 minutes. Therefore, we assume that the traffic value obtained after the processing will remain constant during periods of 10 minutes. This means that in the period with the highest traffic, the arrival rate of H2H UEs will be 55 per second during 10 minutes. For M2M traffic, we use the model proposed in [38], where they represent a heavy load scenario as 30 000 UEs following a Beta(3,4) distribution over 10 seconds. This is a bursty traffic limit scenario that might arise when many M2M devices are synchronized, for example when an alarm is set due to the same event, or a power outage has occurred [38]. Bear in mind that training will be defined by periods of 10 minutes. Every 10 minutes, H2H traffic will change as it was explained earlier. Also, every 10 minutes a new M2M burst of traffic will be injected into the system. However, the characteristics of each burst will be identical. Therefore, when the system is trained with one day of data, 144 cycles of 10 minutes will occur. In order to explore the state space, when the systems are trained the first simulation run starts with an exploration probability of $\epsilon = 0.9$, and then this value decreases linearly until the last simulation run, in which $\epsilon = 0.01$ is used.

For the evaluation of the different mechanisms, we focus on the mean access

delay $E[D]$, the successful access probability P_{sa}, and the mean number of preamble transmissions $E[K]$. These are the typical KPIs evaluated in this type of studies, which will allow us to compare the performance of the proposed adaptive scheme to the performance of other schemes proposed in the literature. Just like for the training phase, a single cell environment is assumed to evaluate the network performance. The system conveys traffic from both H2H and M2M UEs. The access requests of H2H UEs are distributed uniformly over time with a mean arrival rate of $\lambda_H = 55$ arrivals/s. That is, the evaluation is done for the highest H2H traffic possible. Unless otherwise stated, the M2M traffic will follow a $Beta(3, 4)$ distribution over a period of 10 s with 30 000 UEs according to the traffic model 2 specified by the 3GPP in [38]. We consider the typical PRACH configuration for these kind of studies, *prach-ConfigIndex 6*, in conformance to the LTE-A specification [30, 38], with the parameter values listed in Table 1. Note that a preamble detection probability ($P_d = 1 - \frac{1}{e^k}$ for the kth preamble transmission [38]) is assumed to take into account the effects of radio channels, for example path-loss, fading, inter-cell interference, among others. Unless otherwise stated, the results shown are the mean values of 200 simulation runs using the above-mentioned parameters. Each simulation run uses a different random seed and ends when all the M2M UEs have completed their random access procedure. For the evaluation of the different schemes, the exploration probability is set to $\epsilon = 0$.

5.1. Evaluation of QL-ACB Mechanism

In Fig. 3 and Table 2, we can see the performance of the classical QL-ACB algorithm when it is trained with 1 day of data with the learning rate α taking the values 0.05, 0.15, and 0.25. As it can be observed, using higher values of α can increase the delay, which in the case of α =0.25 is about 0.5 s higher than for α =0.05. Furthermore, as α grows beyond 0.25, the performance of the system keeps on declining, and therefore the intermediate value of 0.15 seems adequate. However, for both α =0.25 and α =0.05, all the H2H UEs access successfully the system, which is not the case when α =0.15. A similar thing could be said

Table 1: Default System Configuration for Evaluation Purposes

Parameter	**Setting**
PRACH Configuration Index	*prach-ConfigIndex* = 6
Periodicity of RAOs	5 ms
Subframe length	1 ms
Available preambles for contention-based random access	$R = 54$
Maximum number of preamble transmissions	*preambleTransMax* = 10
RAR window size	$W_{\mathrm{RAR}} = 5$ subframes
Maximum number of uplink grants per subframe	$N_{\mathrm{RAR}} = 3$
Maximum number of uplink grants per RAR window	$N_{UL} = W_{\mathrm{RAR}} \times N_{\mathrm{RAR}} = 15$
Preamble detection probability for the kth preamble transmission	$P_d = 1 - \frac{1}{e^k}$ [38]
Backoff Indicator	$BI = 20$ ms
Re-transmission probability for *Msg3* and *Msg4*	0.1
Maximum number of *Msg3* and *Msg4* transmissions	5
Preamble processing delay	2 subframes
Uplink grant processing delay	5 subframes
Connection request processing delay	4 subframes
Round-trip time (RTT) of *Msg3*	8 subframes
RTT of *Msg4*	5 subframes
Discount factor	$\gamma = 0.7$
Learning rate	$\alpha = 0.15$
Num. hidden layers (feedforward neural network)	$N_L = 10$
Update period (events) of the second neural network	$\tau = 100$
Buffer size for experience replay	$ER = 500$

for M2M UEs, since for α =0.25 and α =0.05, 99.98 % of UEs can access the system, more than for the original value of α. In terms of mean number of preamble transmissions there are no important differences to be noticed, and the values are consistent with the behaviour explained earlier.

Similar results are obtained when we change the value of γ from 0.3 to 0.9,

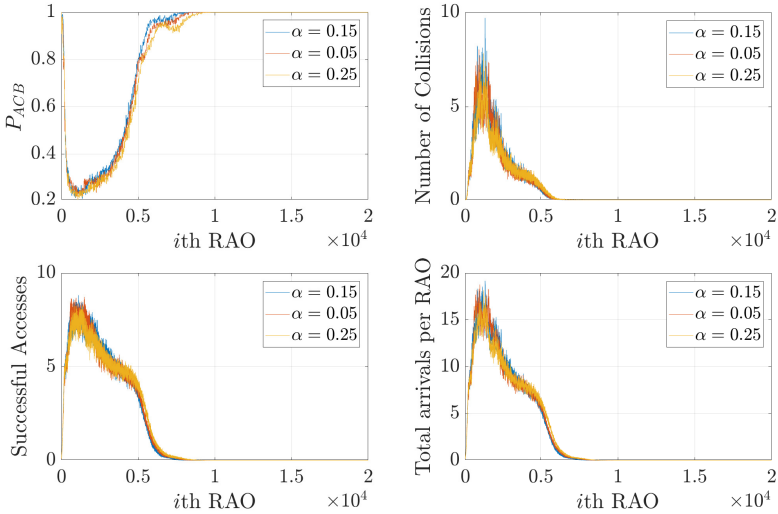

Figure 3: Congestion and Successful Accesses for QL-ACB as α varies.

Table 2: Mean Access Delay and Mean Number of Pream. Tx for QL-ACB as α varies.

	Mean Access Delay (s)			**Mean N. Pream. Tx**			**Success. Access Prob. (%)**		
	α=0.05	α=0.15	α=0.25	α=0.05	α=0.15	α=0.25	α=0.05	α=0.15	α=0.25
M2M	8.37	8.24	8.87	1.78	1.80	1.76	99.98	99.97	99.98
H2H	3.65	3.61	3.77	1.58	1.60	1.58	100	99.99	100
All UE	8.17	8.05	8.65	1.77	1.79	1.75	99.99	99.97	99.99

as it can be seen in Fig. 4 and Table 3. For very low values of γ, the delay of the system grows due to the tendency of the design to reduce the congestion by limiting the amount of UEs that can access the system. In fact, when $\gamma = 0.7$, the mean access delay for all UE is almost 1 s less than when we use $\gamma = 0.3$. This value of γ is also desirable because it allows all H2H UEs to access the system. However if we increase γ beyond 0.7, the congestion of the system grows and some access attempts are discarded after the maximum number of preamble transmissions is reached. This might be due to the fact that increasing γ adds weight to the estimation of Q-values, which intensifies the tendency of the Q-Learning algorithm to overestimate them due to the difficulty on predicting

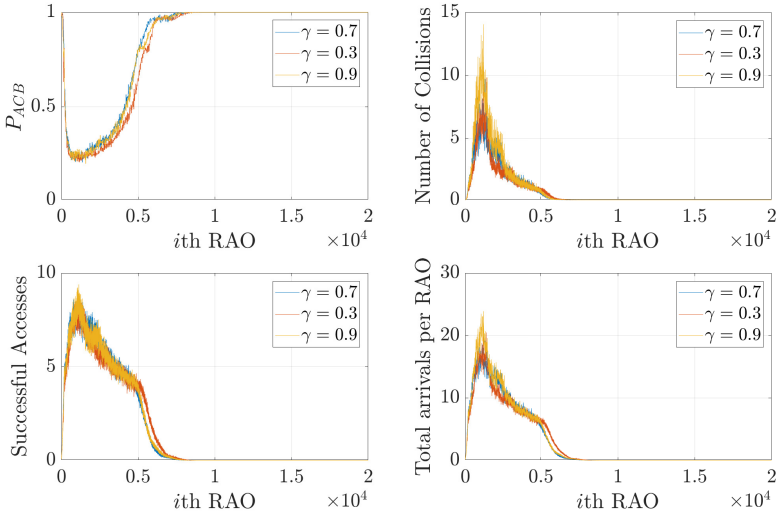

Figure 4: Congestion and Success Probability for QL-ACB as γ varies.

future rewards in the transient regime.

Table 3: Mean Access Delay and Mean Number of Pream. Tx for QL-ACB as γ varies.

	Mean Access Delay (s)			**Mean N. Pream. Tx**			**Success. Access Prob. (%)**		
	γ=0.30	γ=0.70	γ=0.90	γ=0.30	γ=0.70	γ=0.90	γ=0.30	γ=0.70	γ=0.90
M2M	8.81	8.05	8.15	1.77	1.80	1.91	99.97	99.97	99.95
H2H	4.04	3.61	3.70	1.58	1.60	1.62	100	99.99	99.99
All UE	8.81	8.05	8.15	1.76	1.79	1.90	99.97	99.97	99.95

In Fig. 5 and Table 4 the performance of the QL-ACB mechanism is shown after being trained with data from 1, 2, 3, and 4 days and with 30 000 M2M UEs. It can be seen that after training with 4 days of data, the system is able to reduce the mean access delay in more than 0.6 s for M2M UEs, and 0.3 s for H2H UEs. Also, there is a slight reduction of the mean number of preamble transmissions. However, what is more important is that in the case of training with four days of data, the success probability reaches 100 % for H2H UEs, and 99.99 % for M2M UE. It is also interesting to notice that when the algorithm is

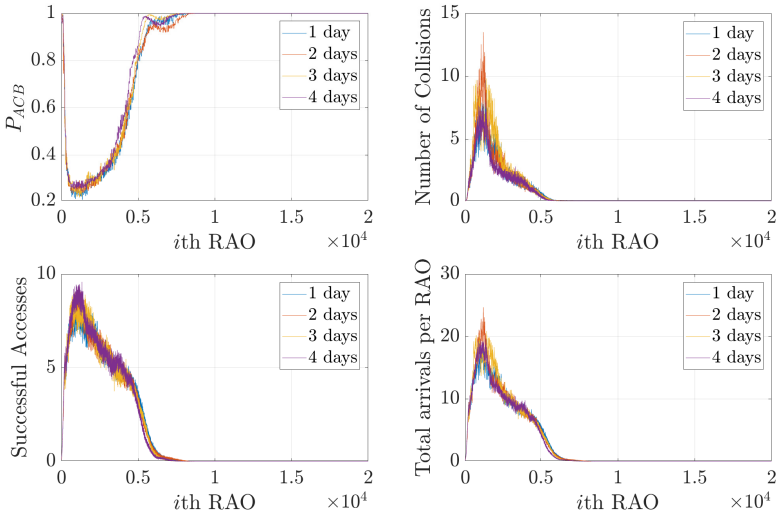

Figure 5: Congestion and Success Probability for QL-ACB as training evolves.

trained with data from 2 or 3 days, the success probability is reduced for M2M UEs. This can be explained because in these cases the number of collisions grows. It is also worth noticing that the training time was 803 s for the first case, 1 692 s for the second case, 2 429 s for the third case, and 3 146 s for the last case, that is, the training time was less than an hour in the worst case. This means that it takes around 800 s to train the system with one day of data.

Table 4: Mean Access Delay and Mean Number of Pream. Tx for QL-ACB as training evolves and 30 000 M2M UEs.

	Mean Access Delay (s)				**Mean N. Pream. Tx**				**Success. Access Prob. (%)**			
	1 day	2 days	3 days	4 days	1 day	2 days	3 days	4 days	1 day	2 days	3 days	4 days
M2M	8.24	7.99	7.85	7.57	1.80	1.86	1.89	1.76	99.97	99.87	99.89	99.99
H2H	3.61	3.41	3.56	3.35	1.60	1.60	1.64	1.59	99.99	99.98	99.98	100
All UE	8.05	7.80	7.68	7.41	1.79	1.85	1.88	1.76	99.97	99.88	99.89	99.99

In Table 5 and Fig. 6, we evaluate the performance of the QL-ACB algorithm when it is trained during 1, 2, 3, and 4 days, and there are 10 000 M2M UEs. We consider that H2H traffic is the same as in the previous experiments. It can

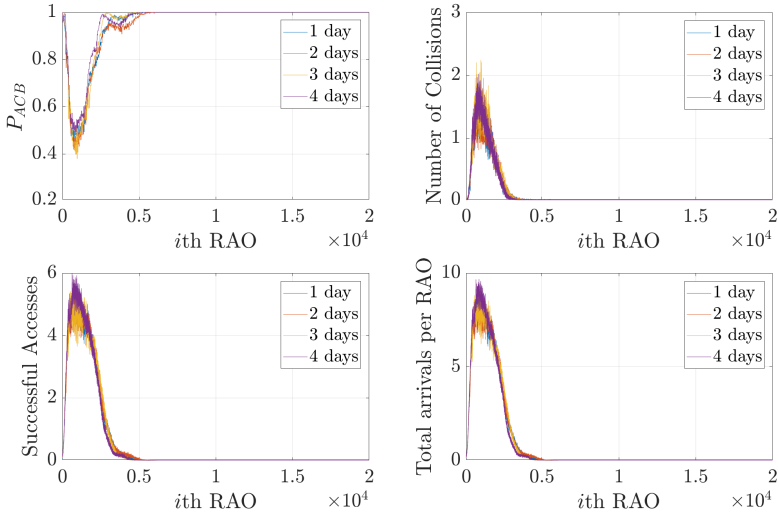

Figure 6: Congestion and Success Probability for QL-ACB as training evolves with 10000 M2M UEs.

be seen that the system adapts correctly although the traffic is much lower, and the system is able to accept all UEs. As expected, the mean access delay and the mean number of preamble transmissions are reduced, which is consistent with the low traffic offered to the system. It is also interesting to notice that in this case the minimum value of P_{ACB} is 0.4, that is higher than the one achieved for the scenario with 30 000 M2M UEs. This shows the effectiveness of the proposed adaptive scheme.

Table 5: Mean Access Delay and Mean Number of Pream. Tx for QL-ACB as training evolves and 10000 M2M UEs.

	Mean Access Delay (s)				**Mean N. Pream. Tx**				**Success. Access Prob. (%)**			
	1 day	2 days	3 days	4 days	1 day	2 days	3 days	4 days	1 day	2 days	3 days	4 days
M2M	2.63	2.72	2.68	2.24	1.58	1.58	1.60	1.59	100	100	100	100
H2H	1.09	1.08	1.11	0.91	1.50	1.50	1.51	1.50	100	100	100	100
All UE	2.52	2.59	2.58	2.15	1.58	1.57	1.59	1.58	100	100	100	100

In Table 6 and Fig. 7, the performance of the system after being trained

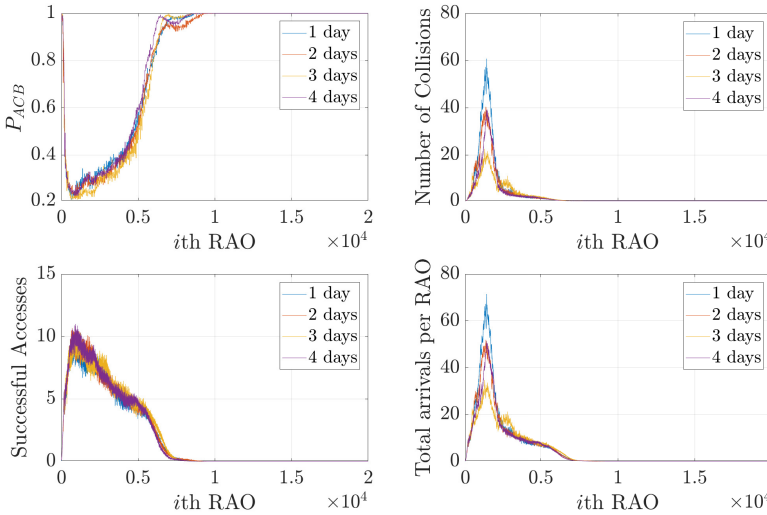

Figure 7: Congestion and Success Probability for QL-ACB as training evolves with 40000 M2M UEs.

with data from 1, 2, 3, and 4 days, when the system receives 40 000 M2M UEs is shown. This scenario has 33 % more traffic than the scenario of maximum congestion proposed in [38], and as expected the performance of the system suffers. In fact, the best value of the successful access probability for M2M UEs is below 98 %, and for H2H UEs the 100 % could not be achieved, as it occurred in the scenario with 30000 M2M UEs. The system that is trained with 3 days of data is able to reduce the congestion considerably by reducing P_{ACB} for a longer time, but this is not enough to accept more UEs in the system.

Table 6: Mean Access Delay and Mean Number of Pream. Tx for QL-ACB as training evolves and 40000 M2M UEs.

	Mean Access Delay (s)				**Mean N. Pream. Tx**				**Success. Access Prob. (%)**			
	1 day	2 days	3 days	4 days	1 day	2 days	3 days	4 days	1 day	2 days	3 days	4 days
M2M	8.99	9.09	9.68	8.78	2.12	2.09	1.95	1.93	92.03	95.17	97.66	95.15
H2H	4.02	3.96	4.50	3.94	1.67	1.67	1.68	1.64	97.36	98.57	99.15	98.16
All UE	8.81	8.91	9.50	8.62	2.11	2.07	1.94	1.92	92.16	95.26	97.69	95.22

5.2. Evaluation of Double Deep QL-ACB Mechanism

In the case of the Double Deep QL-ACB (DDQL) mechanism, we used the Levenberg-Marquardt implementation used in the MATLAB Optimization Toolbox with its default parameters to train the feedforward neural network. In Fig. 8 and Table 7, we evaluate the impact of γ on the Double Deep QL algorithm when $N_L = 10$ and it is trained with 1 day of data. In the top-left graph of Fig. 8, it can be seen that the policy when $\gamma = 0.9$ changes drastically from the other two between the 3 600th and 10 000th RAOs. This policy takes a longer time to increase P_{ACB} back to one after the traffic peak has finished, and it also imposes the highest restriction on UEs when the traffic is peaking, by reducing P_{ACB} more than the other two systems. As a result of this, that policy increases the mean access delay for M2M, which are the majority of UEs. However, although the policies might seem very different, their performance do not substantially differ. The mean number of preamble transmissions does not suffer too much, and in fact, all the three policies allow 100 % of the UE to access successfully the network. This is an important achievement considering that this is the main objective of ACB.

Table 7: Mean Access Delay and Mean Number of Pream. Tx for Double Deep QL-ACB N_L =10 as γ varies.

	Mean Access Delay (s)			**Mean N. Pream. Tx**			**Success. Access Prob. (%)**		
	γ=0.30	γ=0.70	γ=0.90	γ=0.30	γ=0.70	γ=0.90	γ=0.30	γ=0.70	γ=0.90
M2M	6.28	5.96	6.62	1.72	1.77	1.77	100	100	100
H2H	2.60	2.98	2.66	1.57	1.63	1.55	100	100	100
All UE	6.13	5.87	6.42	1.71	1.77	1.75	100	100	100

In Fig. 9 and Table 8, the performance of the Double Deep QL-ACB mechanism is shown when $N_L = 5$ and γ varies from 0.3 to 0.9. It is evident that when γ =0.9, the system does not perform correctly, showing that increasing the weight on the estimates of future values of Q to update its value according to Eq. (2) is detrimental to the performance of the system. This is a result of

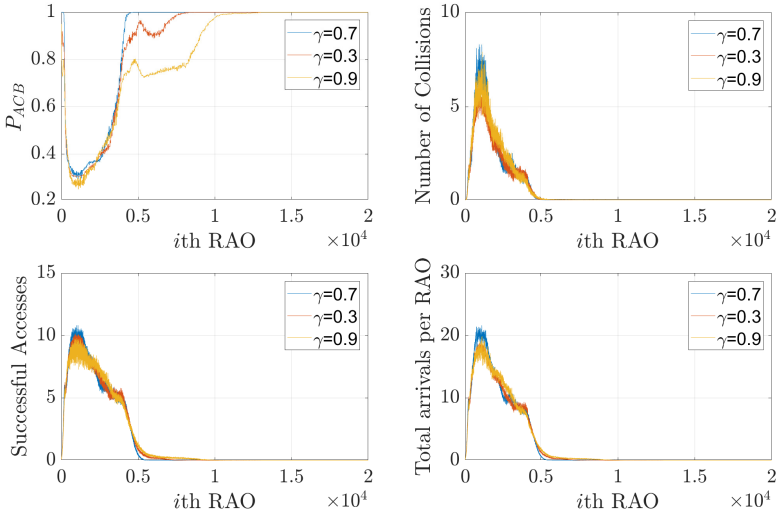

Figure 8: Congestion and Success Probability for Double Deep QL-ACB N_L =10 as γ varies.

a wrong estimation of Q, which in this cases is done by a feedforward neural network with only 5 hidden layers. In fact, it is only in this configuration that the system does not reach a 100 % of successful access for both types of UEs. On the other hand, when $\gamma = 0.3$, the mean access delay is reduced for both types of UEs while maintaining a 100 % of successful access probability. Therefore, one might consider that a lower value might be convenient when there is not too much data available for training. Also, it should be noted that this behaviour was not seen when $N_L = 10$, and therefore having more hidden layers provides stability.

Table 8: Mean Access Delay and Mean Number of Pream. Tx for Double Deep QL-ACB N_L =5 layers as γ varies.

	Mean Access Delay (s)			**Mean N. Pream. Tx**			**Success. Access Prob. (%)**		
	γ=0.30	γ=0.70	γ=0.90	γ=0.30	γ=0.70	γ=0.90	γ=0.30	γ=0.70	γ=0.90
M2M	5.77	6.24	6.37	1.76	1.73	1.89	100.00	100.00	99.52
H2H	2.40	3.13	2.59	1.56	1.61	1.74	100.00	100.00	99.81
All UE	5.63	6.14	6.25	1.75	1.73	1.88	100.00	100.00	99.52

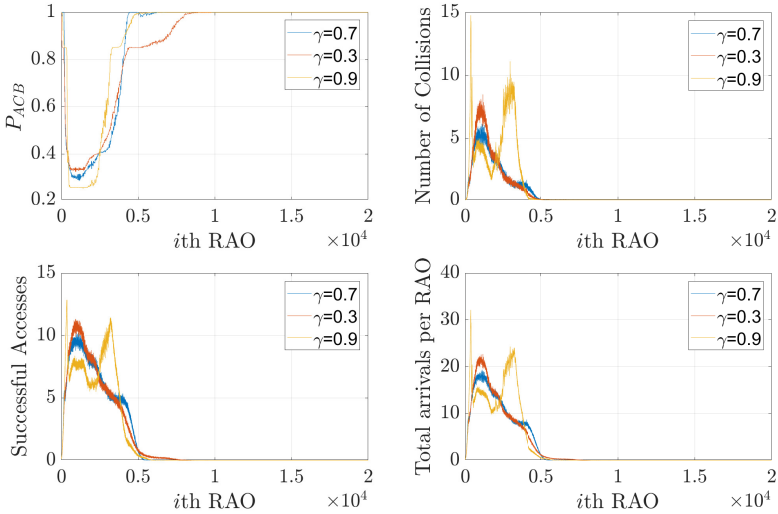

Figure 9: Congestion and Success Probability for Double Deep QL-ACB N_L =5 as γ varies.

In Fig. 10 and Table 9, the performance of the Double Deep QL-ACB mechanism is shown when $N_L = 10$, and the algorithm is trained with data from 1, 2, 3, and 4 days. It can be seen that the system aims to reduce congestion, which is reflected in a reduction on the mean number of preamble transmissions and an increase in the mean access delay, which is clearly a trade-off. However, every configuration maintains a 100 % successful probability for both types of UEs. In fact, in this scenario the number of collisions never surpasses 8, a very low value when compared with the Classical Q-Learning solution. This means that our design reflected on the reward function is consistent with the behaviour of the system. This performance is superior to that achieved through classical Q-learning which implies that the generalization achieved by Double Deep Q-learning is appropriate to this problem. The training time was 2 459 s for 1 day of trining data, 4 953 s for 2 days, 7 042 s for the 3 days, and 8 632 s for 4 days. These times are about 3 times higher than those obtained with the classical Q-learning algorithm, and therefore another trade-off has to be considered.

In Fig. 11 and Table 10, the performance of the Double Deep QL-ACB mech-

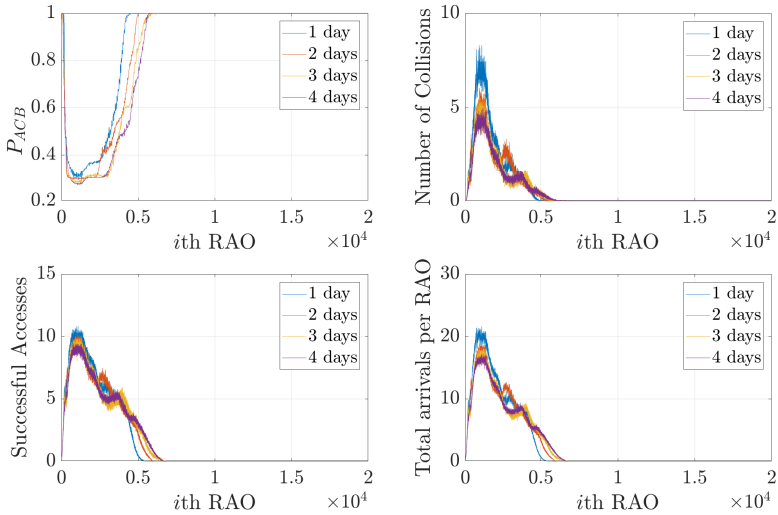

Figure 10: Congestion and Success Probability for Double Deep QL-ACB N_L =10 as training evolves.

Table 9: Mean Access Delay and Mean Number of Pream. Tx for Double Deep QL-ACB N_L =10 as training evolves.

	Mean Access Delay (s)				**Mean N. Pream. Tx**				**Success. Access Prob. (%)**			
	1 day	2 days	3 days	4 days	1 day	2 days	3 days	4 days	1 day	2 days	3 days	4 days
M2M	5.96	6.79	7.38	7.62	1.77	1.71	1.68	1.67	100.00	100.00	100.00	100.00
H2H	2.98	3.39	3.58	3.87	1.63	1.59	1.57	1.56	100.00	100.00	100.00	100.00
All UE	5.87	6.68	7.25	7.48	1.77	1.70	1.68	1.66	100.00	100.00	100.00	100.00

anism is shown when $N_L = 5$, and the algorithm is trained with data from 1, 2, 3, and 4 days. It can be seen that as the training increases, the algorithm is able to reduce the congestion successfully, bounding the mean number of collisions below 5 when the algorithm is trained with data from 4 days. However, while the congestion is reduced, and with it the mean number of preamble transmissions, the mean access delay is also increased, which can be seen from the shift that the P_{ACB} curve has to the right as training increases. This is a result of the design solution that we have made, which rewards important congestion reduc-

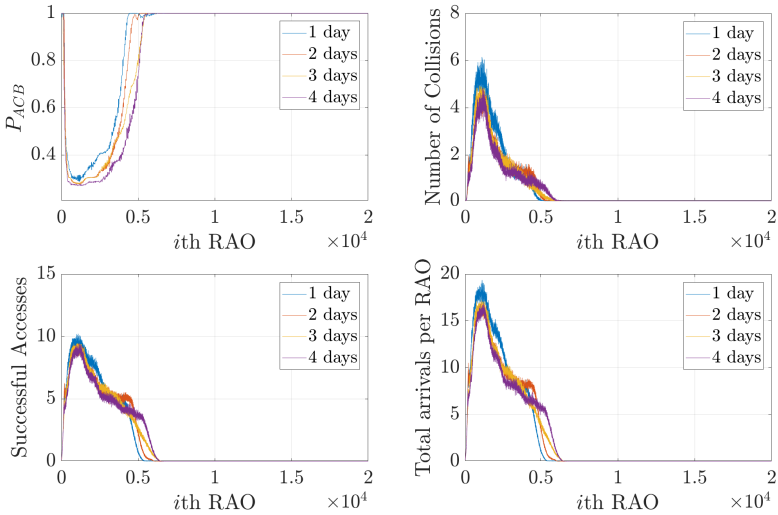

Figure 11: Congestion and Success Probability for Double Deep QL-ACB N_L =5 as training evolves.

tions. One might argue that this is not the best solution due to the increase on delay, but we have to keep in mind that the ACB mechanism aims to maximize the acceptance rate, not to minimize the delay. Also, this allows to reduce the mean number of preamble transmissions which is important in scenarios where a large number of low-powered devices are deployed. Moreover, this algorithm is able to accept all M2M and H2H UEs, which makes it very efficient in all four cases. The training time was 2 420 s for 1 day, 4 704 s for 2 days, 6 782 s for 3 days, and 8 526 s for 4 days. In all four cases, we see a reduction in relation to the system where $N_L = 10$, which makes this system more efficient. However, we should remember that this system is not as insensitive to γ variations as the system with $N_L = 10$ is.

In Fig. 12 and Table 11, the performance of the Double Deep QL-ACB mechanism with $N_L = 10$ is illustrated when it is trained with data from 1, 2, 3, and 4 days of data, and the number of M2M UEs is 10 000. As it can be seen, all UEs can successfully access the system in every scenario, although the mean access delay grows when more data is used for training. This happens because

Table 10: Mean Access Delay and Mean Number of Pream. Tx for Double Deep QL-ACB N_L =5 as training evolves.

	Mean Access Delay (s)				**Mean N. Pream. Tx**				**Success. Access Prob. (%)**			
	1 day	2 days	3 days	4 days	1 day	2 days	3 days	4 days	1 day	2 days	3 days	4 days
M2M	6.24	7.17	7.34	8.10	1.73	1.68	1.68	1.66	100.00	100.00	100.00	100.00
H2H	3.13	3.50	3.65	4.29	1.61	1.59	1.57	1.57	100.00	100.00	100.00	100.00
All UE	6.14	7.05	7.21	7.96	1.73	1.68	1.67	1.66	100.00	100.00	100.00	100.00

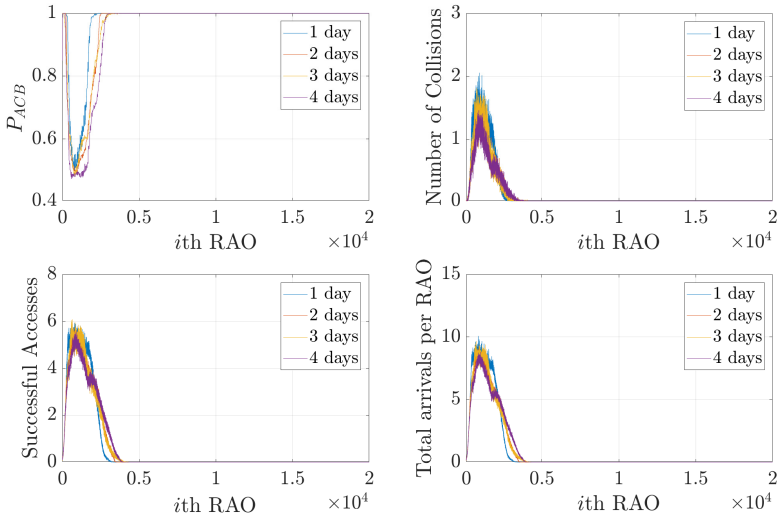

Figure 12: Congestion and Success Probability for Double Deep QL-ACB N_L =10, M2M UEs=10000, as training evolves.

the algorithm aims to reduce the collisions, and in fact after being trained with 4 days of data, the number of collisions is reduced below 1.5. Also, the mean number of preamble transmissions is reduced. It can be seen that the algorithm adapts to the traffic, as P_{ACB} tends to be higher than 0.5.

In Fig. 13 and Table 12, the performance of the Double Deep QL-ACB mechanism with $N_L = 10$ and 40000 M2M UEs is shown, when the algorithm is trained with 1, 2, 3, and 4 days of data. The most important characteristic that can be seen is that after training with 3 and 4 days of data, all UEs can

Table 11: Mean Access Delay and Mean Number of Pream. Tx for Double Deep QL-ACB N_L =10, M2M UEs=10 000, as training evolves.

	Mean Access Delay (s)				**Mean N. Pream. Tx**				**Success. Access Prob. (%)**			
	1 day	2 days	3 days	4 days	1 day	2 days	3 days	4 days	1 day	2 days	3 days	4 days
M2M	1.82	2.43	2.36	2.90	1.60	1.56	1.57	1.55	100.00	100.00	100.00	100.00
H2H	0.82	1.16	1.06	1.42	1.54	1.52	1.50	1.50	100.00	100.00	100.00	100.00
All UE	1.77	2.36	2.28	2.81	1.60	1.56	1.57	1.55	100.00	100.00	100.00	100.00

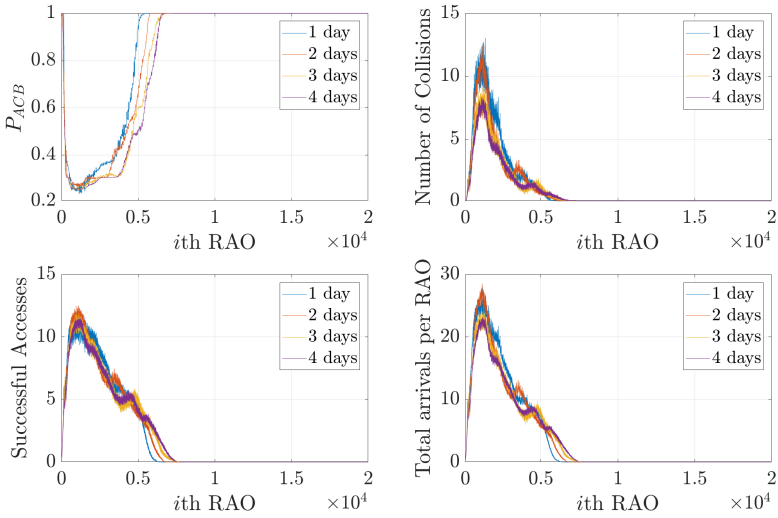

Figure 13: Congestion and Success Probability for Double Deep QL-ACB N_L =10, M2M UEs=40 000 as training evolves.

access the system. This shows that the Double Deep QL-ACB mechanism is able to adapt to higher loads, unlike the previous mechanism. This is very positive if we consider that there is about a 30 % increase in the load of the system, showing that the algorithm outperforms other solutions even in critical scenarios. However, this performance is achieved at the expense of increasing the mean access delay, that is, there is a trade-off. In fact, the mean access delay grows over 8 s for M2M UEs and over 4 s for H2H UEs.

Table 12: Mean Access Delay and Mean Number of Pream. Tx for Double Deep QL-ACB N_L =10, M2M UEs=40 000 as training evolves.

	Mean Access Delay (s)				**Mean N. Pream. Tx**				**Success. Access Prob. (%)**			
	1 day	2 days	3 days	4 days	1 day	2 days	3 days	4 days	1 day	2 days	3 days	4 days
M2M	7.58	7.85	8.57	8.84	1.91	1.84	1.77	1.75	99.99	99.97	100.00	100.00
H2H	3.80	4.05	4.25	4.54	1.68	1.63	1.60	1.59	100.00	99.99	100.00	100.00
All UE	7.48	7.75	8.43	8.71	1.90	1.84	1.76	1.74	99.99	99.97	100.00	100.00

5.3. Comparison of dynamic ACB mechanisms

In this subsection, we compare the two reinforcement learning dynamic solutions, with a well-known dynamic solution proposed in [9]. This solution, named D-ACB, dynamically adapts P_{ACB} to maximize the number of preambles that access the system. However, this solution cannot been implemented directly because it ignores the constraint on the maximum number of preamble transmissions that a UE can perform before considering that its connection has failed. Therefore, we have implemented a modified version, called D-ACB modified, which extends the maximum number of preamble transmissions to 150, in order to avoid that UEs are blocked due to their number of preamble transmissions. Also, for both D-ACB solutions, we increased the backoff indicator to 960 ms to allow a fair comparison. In the case of the QL-ACB solution, we use the results after training with 1 day of data with α=0.15 and γ=0.7. For the Double Deep QL-ACB solution we use the results after training with 1 day of data the neural network with N_L=10 and when γ=0.7.

We have evaluated our two reinforcement learning solutions, the D-ACB solution without modification, and the D-ACB solution with the previously mentioned modification when 55 H2H UEs arrive per second, and when there are 10000, 30000 and 40000 M2M UEs that arrive following a Beta(3,4) distribution during 10 s. In Fig. 14, it can be seen that the successful access probability remains unchanged as the number of M2M UEs grows for the D-ACB modified and Double Deep QL-ACB solutions. On the other hand, the QL-ACB solution is not able to accept all UEs, and when there are 40000 M2M UEs, P_s is re-

duced to 92 %. This is the result of the inability of the classical QL mechanism to generalize, which harms the performance of the system when it is presented to new traffic patterns. In the case of the D-ACB solution, P_s is 79 % when there are 30000 M2M UEs and 46 % when there are 40000 M2M UEs. This happens because this solution tries to maximize the number of preambles that are accepted in the system per second, causing more collisions. Since the maximum number of preamble transmissions is 10, a high percentage of UEs are blocked when they reach this value. The modified version of D-ACB does not suffer this problem since the maximum number of preamble transmissions has been increased to 150. This behaviour can be easily seen in Fig. 15, where the mean number of preamble transmissions is higher for the D-ACB modified scheme. Although this solution is able to accept all the UEs even for very high traffic loads, it does that by increasing the mean number of preamble transmissions, which is detrimental to the performance of energy-constrained devices, such as those that use M2M communications. In fact, when there are 40000 M2M UEs, the maximum number of preamble transmissions is around 100. Although not as much as in the previous case, the D-ACB solution also increases the mean number of preamble transmissions. In the case of 30000 M2M UEs, the mean number of preamble transmissions for the D-ACB solution is around 4.6, while it reaches 1.79 and 1.77 for the QL-ACB and Double Deep QL-ACB solutions, respectively. Hence, in this scenario the UEs will use around three times more energy when they use the D-ACB mechanism instead of the reinforcement learning solutions. This is more critical if we consider that the D-ACB solution can only accept around 80 % of the UEs as it was mentioned earlier. However, since the D-ACB solutions aim to increase the throughput, they are able to reduce the mean delay considerably. As illustrated in Fig. 16, the D-ACB solution has the lowest mean delay in all the scenarios, though this is done by reducing P_s. The modified D-ACB solution also reduces the mean delay, although it is considerably increased when there are 40000 M2M UEs. This happens because as more UEs retry, more congestion occurs, and therefore the delay also grows. Hence, for the modified D-ACB the mean delay grows considerably when there

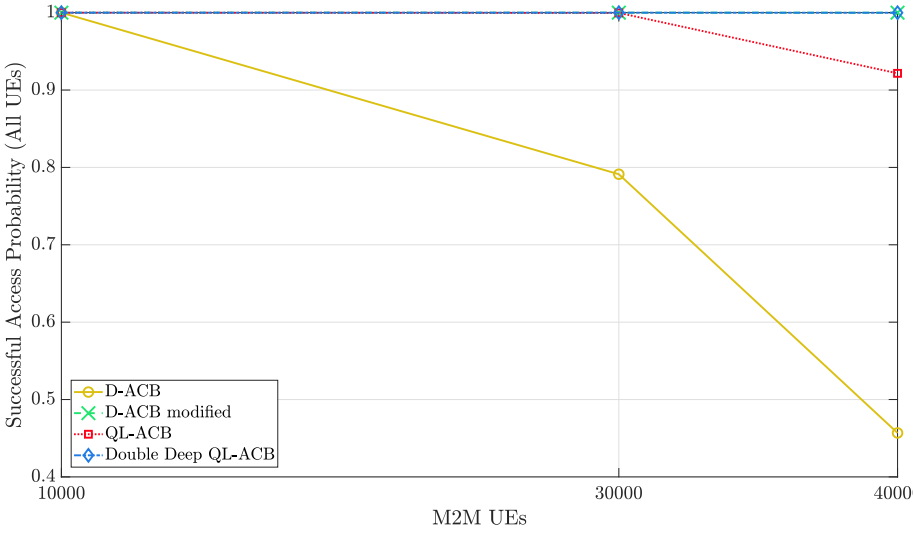

Figure 14: Successful Probability Access for All UEs.

are 40000 M2M UEs. In the case of the reinforcement learning solutions, the delay is usually higher. However, it should be noted that the Double Deep QL-ACB solution maintains a lower mean delay than that of the QL-ACB solution, and even if it is higher than the D-ACB solutions, it does it while accepting all the UEs in the system with a lower number of transmissions. Therefore, by deciding to avoid congestion through a lower number of transmissions, our solutions are able to increase the successful access probability by increasing the delay.

One aspect that has to be considered when implementing reinforcement learning based solutions, is the computational cost associated to learning. In Fig. 17 the cost in time associated to training the two algorithms with data from 1 to 4 days is depicted. Clearly, although the performance of the Double Deep QL-ACB solution is better than the QL-ACB solution in all the KPIs observed previously, it has higher computational cost. In fact, when we use 1 day of data to train the systems, it takes almost three times longer to train the Double Deep QL-ACB solution than the QL-ACB solution. These times are measured on a 2.5 GHz two core processor. The computational cost is marginal for the

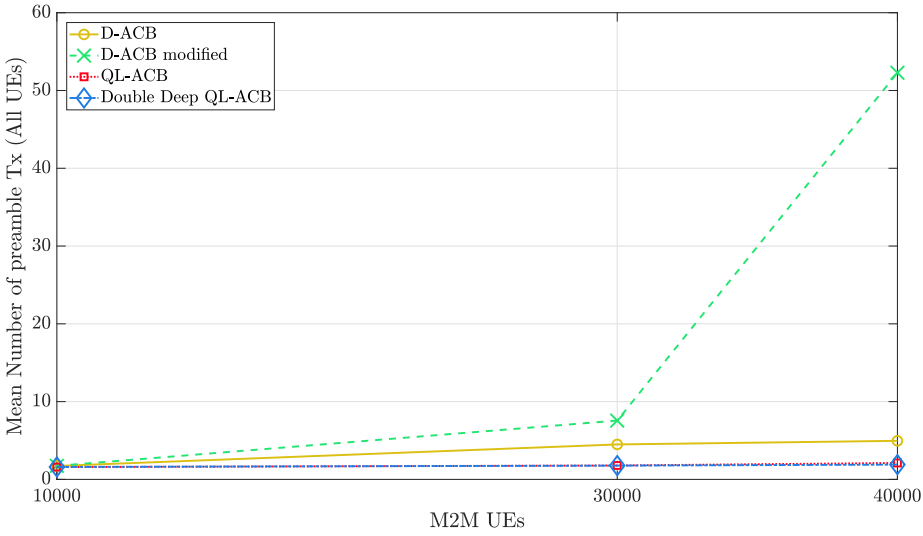

Figure 15: Mean number of preamble transmissions for All UEs.

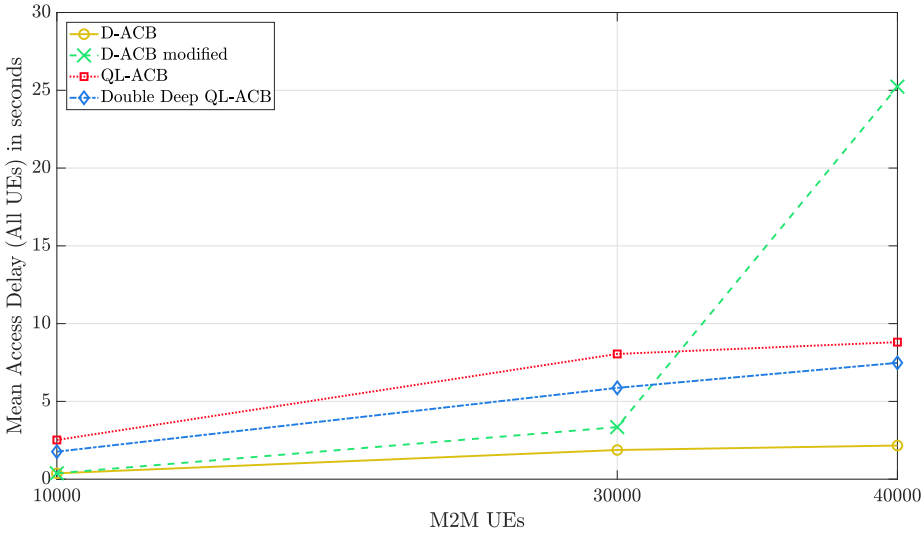

Figure 16: Mean Access Delay for All UEs.

D-ACB mechanism, being just an online algorithm that does not need to learn the system behaviour.

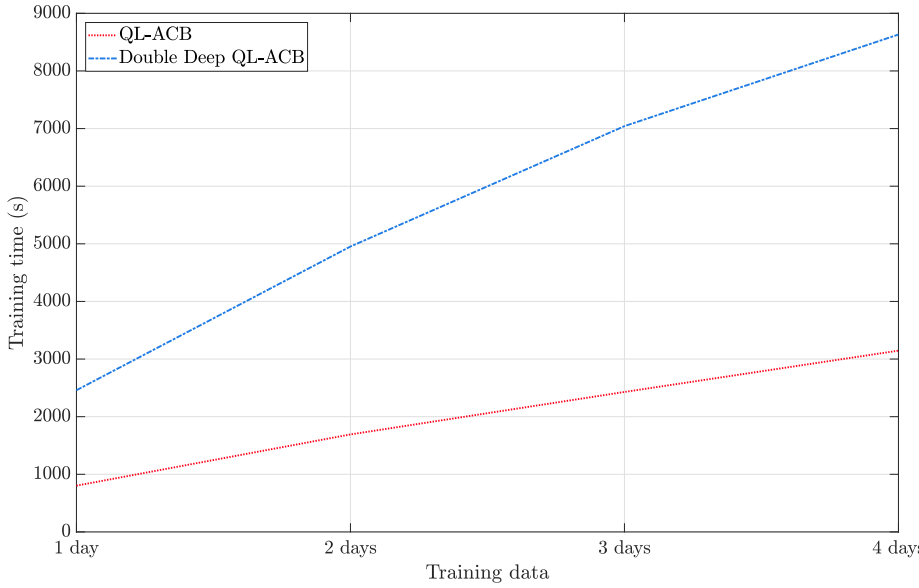

Figure 17: Training Cost QL-ACB vs Double Deep QL-ACB.

6. Conclusions

In this work, we proposed a Double Deep Q-Learning ACB scheme that aims to dynamically adapt the ACB barring rate for massive machine type communications. The mechanism is compatible with the standards, and only considers the information that is available at the base station. In order to evaluate the impact that our solution has on H2H UEs, we considered that H2H and M2M UEs coexist. Also, we used real traces obtained from CDRs of a Telco to represent H2H communications. The mechanism is designed to avoid congestion by reducing the number of transmissions required by each UE, which is beneficial for energy constrained devices. The Double Deep Q-Learning scheme is compared with our previous QL-ACB solution, and a well-known dynamic solution. Results show that the Double Deep QL-ACB scheme outperforms the other solutions by adapting to different traffic conditions, maintaining a 100 % acceptance rate while keeping the number of transmissions of UEs low. Also, the Double Deep QL-ACB solution is able to reach a lower delay than the QL-ACB solution. The performance of the Double Deep QL-ACB scheme is evaluated as

it is trained with different amounts of data, showing its capacity to work even when there are processing or data constraints.

References

[1] T. Taleb, A. Kunz, Machine type communications in 3gpp networks: potential, challenges, and solutions, IEEE Communications Magazine 50 (3) (2012) 178–184.

[2] F. Ghavimi, H.-H. Chen, M2M Communications in 3GPP LTE/LTE-A Networks: Architectures, Service Requirements, Challenges, and Applications, IEEE Commun. Surveys Tuts. 17 (2) (2015) 525–549. doi:10.1109/COMST.2014.2361626.

[3] A. Biral, M. Centenaro, A. Zanella, L. Vangelista, M. Zorzi, The challenges of M2M massive access in wireless cellular networks, Digit. Commun. Netw. 1 (1) (2015) 1–19.

[4] 3GPP, TS 22.368, Service Requirements for Machine-Type Communications (Dec 2014).

[5] Z. Dawy, W. Saad, A. Ghosh, J. G. Andrews, E. Yaacoub, Toward massive machine type cellular communications, IEEE Wireless Communications 24 (1) (2017) 120–128.

[6] I. F. Akyildiz, S. Nie, S.-C. Lin, M. Chandrasekaran, 5G roadmap: 10 key enabling technologies, Computer Networks 106 (2016) 17–48.

[7] 3GPP, TS 36.331, Radio Resource Control (RRC), Protocol specification (Sep 2017).

[8] 3GPP, TS 22.011, V15.1.0, Service Accessibility (June 2017).

[9] S. Duan, V. Shah-Mansouri, Z. Wang, V. W. Wong, D-ACB: Adaptive congestion control algorithm for bursty M2M traffic in LTE networks, IEEE Transactions on Vehicular Technology 65 (12) (2016) 9847–9861.

[10] H. Zang, J. C. Bolot, Mining call and mobility data to improve paging efficiency in cellular networks, in: Proceedings of the 13th annual ACM international conference on Mobile computing and networking, ACM, 2007, pp. 123–134.

[11] K. Kumar, A. Gupta, R. Shah, A. Karandikar, P. Chaporkar, On analyzing indian cellular traffic characteristics for energy efficient network operation, in: Communications (NCC), 2015 Twenty First National Conference on, IEEE, 2015, pp. 1–6.

[12] M. Z. Shafiq, J. Erman, L. Ji, A. X. Liu, J. Pang, J. Wang, Understanding the impact of network dynamics on mobile video user engagement, in: ACM SIGMETRICS Performance Evaluation Review, Vol. 42, ACM, 2014, pp. 367–379.

[13] A. Balachandran, V. Aggarwal, E. Halepovic, J. Pang, S. Seshan, S. Venkataraman, H. Yan, Modeling web quality-of-experience on cellular networks, in: Proceedings of the 20th annual international conference on Mobile computing and networking, ACM, 2014, pp. 213–224.

[14] M. Tavana, A. Rahmati, V. Shah-Mansouri, Congestion control with adaptive access class barring for LTE M2M overload using Kalman filters, Computer Networks 141 (2018) 222–233.

[15] T.-M. Lin, C.-H. Lee, J.-P. Cheng, W.-T. Chen, PRADA: Prioritized random access with dynamic access barring for MTC in 3GPP LTE-A networks, IEEE Trans. Veh. Technol. 63 (5) (2014) 2467–2472.

[16] T. P. de Andrade, C. A. Astudillo, L. R. Sekijima, N. L. da Fonseca, The random access procedure in long term evolution networks for the Internet of Things, IEEE Communications Magazine 55 (3) (2017) 124–131.

[17] R.-H. Hwang, C.-F. Huang, H.-W. Lin, J.-J. Wu, Uplink access control for machine-type communications in lte-a networks, Personal and Ubiquitous Computing 20 (6) (2016) 851–862. doi:10.1007/s00779-016-0961-5.

[18] C. M. Chou, C. Y. Huang, C.-Y. Chiu, Loading prediction and barring controls for machine type communication, in: 2013 IEEE International Conference on Communications (ICC), IEEE, 2013, pp. 5168–5172.

[19] M. Tavana, V. Shah-Mansouri, V. W. S. Wong, Congestion control for bursty M2M traffic in LTE networks, in: Proc. IEEE International Conference on Communications (ICC), 2015, pp. 5815–5820.

[20] L. Tello-Oquendo, D. Pacheco-Paramo, V. Pla, J. Martinez-Bauset, Reinforcement learning-based ACB in LTE-A networks for handling massive M2M and H2H communications, in: 2018 IEEE International Conference on Communications (ICC), IEEE, 2018, pp. 1–7.

[21] A. S. El-Hameed, K. M. Elsayed, A Q-learning approach for machine-type communication random access in LTE-Advanced, Telecommunication Systems (2018) 1–17.

[22] L. M. Bello, P. Mitchell, D. Grace, T. Mickus, Q-learning Based Random Access with Collision free RACH Interactions for Cellular M2M, in: Next Generation Mobile Applications, Services and Technologies, 2015 9th International Conference on, IEEE, 2015, pp. 78–83.

[23] Y. Yu, T. Wang, S. C. Liew, Deep-Reinforcement Learning Multiple Access for Heterogeneous Wireless Networks, in: 2018 IEEE International Conference on Communications (ICC), 2018, pp. 1–7.

[24] Z. Chen, D. B. Smith, Heterogeneous Machine-Type Communications in Cellular Networks: Random Access Optimization by Deep Reinforcement Learning, in: 2018 IEEE International Conference on Communications (ICC), 2018, pp. 1–6.

[25] J. Moon, Y. Lim, A reinforcement learning approach to access management in wireless cellular networks, Wireless Communications and Mobile Computing 2017 (2017) 1–17.

[26] M. S. Obaidat, G. I. Papadimitriou, A. S. Pomportsis, Guest editorial learning automata: theory, paradigms, and applications, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 32 (6) (2002) 706–709. doi:10.1109/TSMCB.2002.1049604.

[27] S. Misra, V. Tiwari, M. S. Obaidat, Lacas: learning automata-based congestion avoidance scheme for healthcare wireless sensor networks, IEEE Journal on Selected Areas in Communications 27 (4) (2009) 466–479. doi:10.1109/JSAC.2009.090510.

[28] F. Morvari, A. Ghasemi, Learning automaton based adaptive access barring for M2M communications in LTE network, in: 24th Iranian Conference on Electrical Engineering (ICEE), 2016, pp. 1466–1470. doi:10.1109/IranianCEE.2016.7585752.

[29] C. Di, B. Zhang, Q. Liang, S. Li, Y. Guo, Learning automata based access class barring scheme for massive random access in machine-to-machine communications, IEEE Internet of Things Journal (2019) 1–1doi:10.1109/JIOT.2018.2867937.

[30] 3GPP, TS 36.321, Medium Access Control (MAC) Protocol Specification (Sept 2012).

[31] 3GPP, TS 36.211, Physical Channels and Modulation (Dec 2014).

[32] D. Chu, Polyphase codes with good periodic correlation properties (corresp.), IEEE Transactions on information theory 18 (4) (1972) 531–532.

[33] L. Tello-Oquendo, I. Leyva-Mayorga, V. Pla, J. Martinez-Bauset, J.-R. Vidal, V. Casares-Giner, L. Guijarro, Performance analysis and optimal access class barring parameter configuration in LTE-A networks with massive M2M traffic, IEEE Transactions on Vehicular Technology 67 (4) (2018) 3505–3520.

[34] L. Tello-Oquendo, V. Pla, I. Leyva-Mayorga, J. Martinez-Bauset, V. Casares-Giner, L. Guijarro, Efficient random access channel evaluation

and load estimation in LTE-A with massive MTC, IEEE Transactions on Vehicular Technology 68 (2) (2019) 1998–2002. doi:10.1109/TVT.2018.2885333.

[35] 3GPP, TR 36.912, Feasibility study for Further Advancements for E-UTRA (Apr 2011).

[36] C. J. C. H. Watkins, P. Dayan, Technical Note Q-Learning, Machine Learning 8 (1992) 279–292.

[37] H. van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double Q-Learning, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2015, pp. 2094–2100.

[38] 3GPP, TR 37.868, Study on RAN Improvements for Machine Type Communications (Sept 2011).

[39] D. Pacheco-Paramo, L. Tello-Oquendo, V. Pla, J. Martinez-Bauset, Reward function for reinforcement learning-based ACB mechanism (Jan. 2019). URL https://dfpp.co/documents

[40] H. V. Hasselt, Double Q-learning, in: Advances in Neural Information Processing Systems 23, Curran Associates, Inc., 2010, pp. 2613–2621. URL http://papers.nips.cc/paper/3964-double-q-learning.pdf

[41] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.

[42] L.-J. Lin, Self-improving reactive agents based on reinforcement learning, planning and teaching, Machine learning 8 (3-4) (1992) 293–321.

[43] Telecomitalia, Telecom italia: Big data challenge (Nov. 2016). URL http://www.telecomitalia.com/tit/en/innovazione/archivio/big-data-challenge-2015.html

[44] Nokia, Mobile Broadband solutions for Mass Events, Tech. rep., Nokia (2014).