Deep Reinforcement Learning-Based Access Class Barring for Energy-Efficient mMTC Random Access in LTE Networks

ANH-TUAN H. BUI, (Graduate Student Member, IEEE), AND ANH T. PHAM, (Senior Member, IEEE)
Computer Communications Laboratory, The University of Aizu, Aizuwakamatsu 965-8580, Japan
Corresponding author: Anh T. Pham (pham@u-aizu.ac.jp)

This work was supported by the Japanese Society for the Promotion of Science’s Grants-in-Aid Scientific Research under Grant 18K11269.

ABSTRACT

Long-Term Evolution (LTE) networks are expected to be a key enabler for the massive Machine-Type Communications (mMTC) service in the 5G context. As highly synchronized access attempts from a massive number of Machine-Type Devices (MTDs) may overload the Physical Random Access Channel (PRACH), the Access Class Barring (ACB) scheme has been officially adopted as a control in LTE specifications. The baseline ACB scheme with fixed barring factor and fixed mean barring time has been shown to prevent the PRACH overload issue at the cost of a sharp increase in access delay. In order to improve the baseline ACB’s delay performance, several studies have suggested discarding the barring time and dynamically adjusting the barring factor over time. While neglecting the barring time can indeed bring a significant delay improvement, it may also cause an increased level of energy consumption for the MTDs because an MTD may need to continuously listen in order to obtain the updated barring factor without getting to actually transmit. In this paper, we propose to dynamically tune both the barring factor and the mean barring time using a reinforcement learning approach known as the Dueling Deep Q-Network. Computer simulations show that given a certain tolerance level on access delay and energy consumption, our design can achieve a significantly higher level of energy satisfaction while maintaining a comparable level of delay satisfaction compared to schemes that only focus on tuning the barring factor.

INDEX TERMS

Access class barring, energy efficient, LTE, massive Machine-Type Communications, random access protocols, reinforcement learning.

I. INTRODUCTION

The massive Machine-Type Communications (mMTC) service featuring billions of autonomous Machine-Type Devices (MTDs) has been identified as one of the three main use cases of 5G networks [1]. Due to its ubiquity and low-cost requirement, the service is to be mainly supported by the current generation Long-Term Evolution (LTE) cellular access technology that has matured in terms of coverage and manufacturing cost. Nevertheless, the LTE technology is originally designed for human-centric communications and may not cope well with the massive device population of the mMTC service. Studies from the 3GPP and literature have indeed shown that when tens of thousands of MTDs wake up and try to access an LTE cell in a highly synchronized manner (e.g., in event-based mMTC applications), the cell’s Physical Random Access Channel (PRACH) may be severely overloaded, which results in a significant portion of the MTDs not being able to obtain access rights before exceeding the maximum number of allowed attempts.

To cope with the PRACH overload issue, the 3GPP has adopted the Access Class Barring (ACB) scheme as a countermeasure in LTE specifications. The scheme works by forcing the devices that are about to request access on the PRACH to first perform a probabilistic check whose passing probability, i.e., the “barring factor” $p_{ACB}$, is under the evolved NodeB’s (eNB) control. Devices that pass the check may proceed while the others are barred for a random duration whose average length is determined by the “mean barring time” $T_{ACB}$ (also controlled by the eNB) before they can redo the check. This mechanism reshapes the highly synchronized access pattern to prevent the PRACH from being overloaded...
by too many concurrent MTDs. In fact, it has been shown that the baseline ACB with fixed \( p_{ACB} \) and \( T_{ACB} \) can significantly increase the access success rate for the MTDs in massive bursty access scenarios at the cost of a sharp increase in access delay [2].

A. RELATED WORKS

In order to improve the baseline ACB’s delay performance while still maintaining its high success rate, various works have proposed to dynamically adjust \( p_{ACB} \) over time. In [3], \( p_{ACB} \) is increased and decreased in a heuristic manner depending on whether the average number of colliding preambles in the last three slots (also known as Random Access Opportunities, or RAOs) exceeds and drops below certain thresholds, respectively. [4]–[8] on the other hand, introduce innovative methods to estimate the number of backlogged MTDs from the numbers of colliding, singleton, and idle preambles, and the used \( p_{ACB} \) in previous RAO(s). Based on such estimates, \( p_{ACB} \) for the next RAO can be set correspondingly so that the average number of transmitting MTDs is kept at an optimum. [9] employs a Kalman filter to refine the estimate further after the arrival period is over, and there is no new MTDs. It is worth noting that except for [8] which assumes a fixed \( T_{ACB} \), most of the works assume that MTDs who fail the barring check in an RAO will retry in the very next RAO (which effectively means discarding the \( T_{ACB} \)) to simplify the backlog estimation process. The effect of different combinations of \( p_{ACB} \) and \( T_{ACB} \) in the baseline ACB scheme has been investigated in [10].

Recently, reinforcement learning, more specifically the Q-learning approach to the dynamic ACB tuning problem, has started to gain traction owing to its ability to produce reasonable solutions using only computer simulations of the system, i.e., without the need to formulate complicated theoretical models. In [11], tabular Q-learning is used to learn a \( p_{ACB} \) control that tries to balance access delay and PRACH collision rate, while \( T_{ACB} \) was not specified. On the other hand, [12] assumes a fixed \( T_{ACB} = 4 \) seconds and uses tabular Q-learning to learn a \( p_{ACB} \) control that greatly reduces contentions on PRACH while accepting the delay cost. Although the tabular method is guaranteed to converge to the optimal policy given that the system satisfies the Markovian property and some mild assumptions [13], it becomes infeasible when the number of possible state-action pairs \((s, a)\) is too big such that the estimate of the Q-function \( Q^*(s, a) \) (also known as the action-value function) cannot be represented exactly in table forms. There has thus been a transition to using deep neural networks (DNNs) to approximate the estimate \( Q(s, a) \) of \( Q^*(s, a) \) by a parameterized version \( Q(s, a; \theta) \). Being an approximation, DNN-based Q-learning is not guaranteed to arrive at optimal policies but instead allows us to obtain sub-optimal solutions of much more complex systems. For example, [14] considers two different systems where the MTDs are single-class and multi-class, and where the reward functions are defined as the number of successful MTDs and the sum of ratios of the number of successful MTDs that meet their class’ delay requirement to the number of successful MTDs of the same class, respectively. A Feed-Forward Network (FFN) is then trained to learn two corresponding sub-optimal \( p_{ACB} \) control policies conforming to the two reward definitions. Additionally, the authors assume exponentially growing \( T_{ACB} \) and backoff time at each individual MTD. [15], on the other hand, discards \( T_{ACB} \) and uses a Gated Recurrent Units (GRU) to learn a \( p_{ACB} \) control that maximizes the overall number of successful MTDs. In [16], an FFN is employed in order to learn a dynamic PRACH resource allocation scheme in NB-IoT systems rather than to control the barring parameters.

B. CONTRIBUTIONS

It is seen that many dynamic ACB solutions only focus on tuning the \( p_{ACB} \) and either discard or assume a fixed or exponential \( T_{ACB} \). While discarding \( T_{ACB} \) and relying on \( p_{ACB} \) alone is beneficial from a delay standpoint, it may lead to very high energy consumption at the MTDs, especially in bursty access scenarios where the devices may need to continuously monitor the downlink to obtain the updated \( p_{ACB} \) without getting to actually transmit due to a persistently low \( p_{ACB} \). This can be partly mitigated by using a per-device \( T_{ACB} \) that grows exponentially with the number of times that the device has failed the barring check as in [14]. Nevertheless, the energy expense may still be noticeable as an MTD may need a considerable number of failed barring checks or, equivalently, \( p_{ACB} \) updates, before its \( T_{ACB} \) grows big enough to escape the low-\( p_{ACB} \) period. Employing a fixed high \( T_{ACB} \), on the other hand, is energy-efficient but obviously results in a much higher overall access delay even if \( p_{ACB} \) is dynamically adjusted.

In order to balance the access delay and energy consumption at the MTDs, both \( p_{ACB} \) and \( T_{ACB} \) should be adaptive. However, backlog estimation and proper barring parameters selection in each RAO become challenging due to the involvement of the variable \( T_{ACB} \). Reinforcement learning is a promising approach to avoid dealing with such mathematical complications, yet there is no (Q-learning) dynamic ACB solution that jointly controls the barring pair to our best knowledge. Therefore, in this paper, we propose a DNN-based Q-learning dynamic ACB scheme that controls both the \( p_{ACB} \) and \( T_{ACB} \) to flexibly realize the tradeoff between the access delay and energy consumption.

II. SYSTEM MODEL AND PROPOSED SOLUTION

A. SYSTEM MODEL

We consider a single LTE cell environment in which \( N \) MTDs arrive following the bursty Beta(3,4) distribution, i.e., traffic model 2 in [17]. All \( N \) MTDs have the same requirements on the tolerable access delay and energy consumption, respectively denoted by \( D_{\text{thresh}} \) and \( E_{\text{thresh}} \). When an MTD arrives at the system, it first monitors the downlink to obtain the System Information Block (SIB2) containing the ACB parameter pair \( (p_{ACB}, T_{ACB}) \) and performs the ACB check by comparing a
Thus, adopted this strategy in our work. This is in line with


every time they are about to transmit a preamble and have


can be achieved if all devices are forced to perform the check


unsuccessful. However, we notice that a better performance


can be achieved if all devices are forced to perform the check


even if their preamble attempts are


required to redo the check even if their preamble attempts are


redone the SIB2) and redo the check.


Even if the MTD passes the check and is allowed to


to send a preamble, it is not guaranteed to succeed as there


may be other MTDs using the same preamble in the same


RAO due to the randomness in preamble selection. In such


case, a collision happens and renders the preamble undecodable [17]. The eNB will thus not send back acknowledgments for the collided preamble(s) during the Random Access Response (RAR) window, and MTDs who have selected those preambles must update the barring parameters (by re-obtaining the SIB2) and re-perform the ACB check after backing off for a period of length \([0, T_{BO}]\). On the contrary, MTDs whose preambles are successfully received will be acknowledged and granted dedicated resources for transmissions of the next signaling message (known as Msg3) containing their IDs and the reasons for requesting access. For each correctly received Msg3, the eNB echoes back the decoded ID via a message known as Msg4. The random access procedure is considered finished upon the correct reception of the Msg4 at the relevant device’s side. Note that if an MTD experiences a certain number of consecutive preamble transmission failures (denoted by \(N_{TX}^{\text{ACB}}\)), it will terminate the random access procedure and is considered temporarily “blocked” by the network. On the other hand, there is no upper limit to the number of failed ACB checks at a device. The ACB scheme is summarized in Fig. 1.

For simplicity, we assume that a preamble transmission only fails due to collisions. Furthermore, in the LTE specifications, the devices that have passed the ACB check will not be required to redo the check even if their preamble attempts are unsuccessful. However, we notice that a better performance can be achieved if all devices are forced to perform the check every time they are about to transmit a preamble and have thus adopted this strategy in our work. This is in line with most existing works on dynamic ACB.

For our energy model, we assume that once arrived at the cell, an MTD is always in either one of the three states, namely transmitting (Tx), receiving (Rx), or idle, with respective power consumption levels denoted by \(P_{\text{Tx}}, P_{\text{Rx}},\) and \(P_{\text{idle}}\). The Tx state is applicable in the subframe where the MTD transmits a preamble. On the other hand, the Rx state is assumed when the MTD is either capturing the RAR window in an attempt to look for an acknowledgment from the eNB, or monitoring the downlink channel to obtain a new SIB2 for the purpose of performing an ACB check. Note that the device may stop capturing the RAR window in the very same subframe where a relevant acknowledgment is found, but otherwise has to capture the whole window without finding any acknowledgments. On the other hand, the MTD is assumed to spend one subframe in Rx state each time it wants to obtain the updated SIB2. When the device is neither transmitting nor receiving, it falls into the idle state.

**B. NEURAL NETWORK-BASED Q-LEARNING**

To achieve both a high delay and energy performance, the barring parameters should be changed over the RAOs to adapt to the unknown access traffic variation. However, the selection of barring parameters itself also affects the observable access pattern. As such, the evolution of dynamic ACB systems (including the proposal) may be approximated by Markov Decision Processes (MDPs) which are succinctly described via four sets, namely a set \(S\) of states of the system, a set \(A\) of possible actions (or decisions) to be taken, a set \(P_{\mu}(s, a)\) modeling the conditional probability that the system will transit to state \(s'\) given that it is currently at state \(s\) and an action \(a\) is to be taken, and a set \(R_{\mu}(s, a)\) representing the intermediate rewards obtained when the system transits from state \(s\) to \(s'\) due to the action \(a\). The ultimate goal in a dynamic system is to find a mapping \(\mu^*: S \rightarrow A\), i.e., a policy dictating which action to take given the current state of the system, that maximizes the expected total rewards, i.e.,

\[
\mu^* = \arg \max_{\mu} \mathbb{E} \left\{ \sum_{t=0}^{\infty} \gamma^t R_{\mu(s_t)}(s_{t+1}) \right\},
\]

where \(\gamma \in (0, 1)\) is the discounted factor needed to make the sum converges.

![FIGURE 1. Access Class Barring (ACB) scheme.](image-url)
It is noted that given the optimal policy \( \mu^* \), the value which the expected sum \( \mathbb{E} \left\{ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right\} \) converges to is a function of the initial state \( s_0 \). This function, denoted by \( V^* \), is known as the value function encoding the maximum achievable total rewards given each starting state. The value function must satisfy the Bellman equations

\[
V^*(s) = \max_a \mathbb{E} \left\{ R_a(s, s') + \gamma V^*(s') \right\},
\]

for all states \( s, s' \in S \). Furthermore, the action produced by the optimal policy, i.e., \( a^* = \mu^*(s) \) must also be the action that achieves the maximum of the right hand side in the Bellman equation (2) [18]. Thus, if we instead have the \( V^*(s) \), then the optimal policy can be readily recovered as

\[
\mu^*(s) = \arg \max_a \mathbb{E} \left\{ R_a(s, s') + \gamma V^*(s') \right\},
\]

for all \( s \). The central problem of finding \( \mu^* \) in a dynamic system can thus be cast into the problem of solving the system of Bellman equations to get the value function \( V^* \). If we define \( \mathbb{E} \left\{ R_a(s, s') + \gamma V^*(s') \right\} \) by \( Q^*(s, a) \), then (2) can be rewritten as

\[
Q^*(s, a) = \mathbb{E} \left\{ R_a(s, s') + \gamma \max_{a'} Q^*(s', a') \right\}.
\]

Intuitively, \( Q^*(s, a) \) is the maximum achievable expected total rewards given that the system is currently at the state \( s \) and action \( a \) is going to be taken. Following (3), the optimal policy \( \mu^* \) can be extracted from \( Q^*(s, a) \) simply as

\[
\mu^*(s) = \arg \max_a Q^*(s, a),
\]

Thus, the problem of finding \( \mu^* \) can also be alternatively cast into the problem of finding \( Q^*(s, a) \) for all pair of \( (s, a) \).

The problems of finding \( V^*(s) \) and \( Q^*(s, a) \) are theoretically equivalent and can both be solved using the Value Iteration (VI) algorithm [18] given the transition probabilities \( P_a(s, s') \) and the associated rewards \( R_a(s, s') \). However, when this information is not available, and the expectations in (2) and (4) are to be estimated by sampling from actual interactions with the system or computer simulations of the system, the approach of finding \( Q^*(s, a) \) can be more convenient. In fact, Watkins et al. [13] showed that by maintaining a table storing the estimates of \( Q^*(s, a) \), henceforth denoted by \( Q(s, a) \), for every possible pair of \( (s, a) \) and only update the estimate of the actual encountered \( (s_t, a_t) \) at time step \( t \) according to

\[
Q_{t+1}(s_t, a_t) = (1 - \xi)Q_t(s_t, a_t) + \xi \left[ r_t + \gamma \max_{a'} Q_t(s'_t, a') \right],
\]

where \( s'_t \) is the actual next state that the system transits into after \( a_t \) has been performed, \( r_t \) is the corresponding observed immediate reward, and \( \xi \) is the learning step at time \( t \), the \( Q_t(s, a) \) will converge to \( Q^*(s, a) \) as \( t \to \infty \) given mild assumptions on the \( \xi \). This method to find \( Q^* \) without requiring knowledge on the \( P_a(s, s') \) and \( R_a(s, s') \) is also known as tabular Q-learning. Note that the discount factor \( \gamma \) plays an important role in determining the convergence rate of (6). The closer \( \gamma \) is to 1, the longer it takes for (6) to converge but at the same time the optimal policy \( \mu^* \) obtained via (5) can achieve a higher maximum expected total reward.

The main issue with tabular Q-learning is that for complex systems, the number of possible \( (s, a) \) pairs can be too large such that the estimate \( Q(s, a) \) cannot be represented exactly in table forms. A natural solution is thus to instead represent the estimate of \( Q^*(s, a) \) by a parameterized \( Q(s, a; \theta) \) using the powerful non-linear function approximators known as DNNs, then, at time \( t \), try to find (learn) the best set \( \theta_t \) of parameters that minimize the squared difference between the two sides of the Bellman equation (4), i.e., try to minimize the loss function

\[
L(\theta_t) = \left[ \mathbb{E} \left\{ Q(s_t, a_t; \theta_t) - \left( r_t + \gamma \max_{a'} Q(s'_t, a'; \theta_t) \right) \right\} \right]^2.
\]

However, early attempts in using DNNs for Q-learning were not fruitful due to the fact that \( Q(s, a; \theta) \) cannot be updated in isolation for each pair of \( (s, a) \) and that consecutive transition samples from the system are highly correlated. The first breakthrough in DNN-based Q-learning is in [19] where Mnih et al. propose to use a memory of size \( T_{\text{mem}} \) known as the “experience replay” that records the transition history, i.e., the tuples \( (s_i, a_i, s'_i, r_i) \) for \( t - T_{\text{mem}} < i \leq t \), and use randomly drawn batches of samples from the memory to mitigate the correlation. Furthermore, a target network is also used instead of the online network in the “target” portion of (7). The parameters \( \theta_t^{\text{target}} \) of this network is periodically overwritten by \( \theta_t \) from the online one with a period of \( T_{\text{copy}} \) time steps but remains fixed otherwise. This helps to stabilize the training process as the target is now fixed w.r.t. the online parameters \( \theta_t \), and the problem of minimizing (7) only changes once every \( T_{\text{copy}} \) step. That is, their objective at time \( t \) is to find a set \( \theta_t \) of parameters that minimizes

\[
L_{\text{DQN}}(\theta_t) = \left[ \mathbb{E} \left\{ Q(s_t, a_t; \theta_t) - \left( r_t + \gamma \max_{a'} Q(s'_t, a'; \theta_t^{\text{target}}) \right) \right\} \right]^2
\]

whose value is estimated from a randomly drawn batch of experiences \( (s, a, s', r) \) from the memory. The experience replay and target network significantly improve performance of their Deep Q Network (DQN).

The work in [20] further prove that \( Q(s, a) \) can be viewed as the sum of two separate terms, namely the value term \( V(s) \) representing the “value” of being at state \( s \) and the advantage term \( A(s, a) \) representing the “advantage” of choosing action \( a \) at state \( s \), and propose the “Dueling DQN” architecture in which the two terms are parameterized as \( V(s, \theta^V) \) and \( A(s, a; \theta^A) \) and combined into the estimate \( Q(s, a; \theta^V, \theta^A) \) as in (9). They then show that such decomposition can greatly improve the system performance compared to the case where \( Q(s, a) \) is parameterized directly,
especially for systems with many “bad” states where a bad state is defined as a state at which the reward is more or less similar regardless of the action to be performed. An example of a dueling DQN with three hidden layers are shown in Fig. 2. It is worth noting that even with the improvements, DNN-based Q-learning solutions remain an approximation, and the corresponding obtained policies are most likely sub-optimal.

\[
Q(s, a; \theta, \theta^V, \theta^A) = V(s; \theta, \theta^V) + \left( A(s, a; \theta, \theta^A) - \frac{1}{|A|} \sum_{a' \in A} A(s, a' ; \theta, \theta^A) \right)
\]

\[\text{(9)}\]

C. PROPOSED DYNAMIC ACB SCHEME

Unlike previous solutions which only control the \(p_{ACB}\), we propose to have the eNB controlling both the barring factor \(p_{ACB}\) and the mean barring time \(T_{ACB}\). The Duelling DQN architecture [20] is then employed to parameterize \(Q(s, a)\). In our system, an action \(a\) is a nonnegative integer, more specifically the index of one among \(|A|\) barring pairs. On the other hand, we assume that the eNB always knows the delay and energy consumption of the successful MTDs in the \(i\)-th RAO before the upcoming \((i + 1)\)-th RAO and stores into its memory the information tuple \(o_i = (N_i^s, N_i^c, D_i^{avg}, E_i^{avg}, a_i)\) where \(N_i^s\) and \(N_i^c\) are the observed number of successful and colliding preambles, while \(D_i^{avg}\) and \(E_i^{avg}\) are the average access delay and average energy consumption of the successful MTDs in the \(i\)-th RAO, respectively. Also, \(a_i\) is the index of the barring pair that were enforced at the beginning of the \(i\)-th RAO. A state \(s\) is then defined as a window of length \(T_{win}\) consecutive observations, i.e., \(s_i = [o_i - T_{win} + 1, o_i - T_{win} + 2, \cdots, o_i]\). The definition of a state as a window of observations improves the performance by enabling the eNB to estimate the trend of access traffic [16].

On the other hand, the reward function design is another significant difference in our work compared to previous DNN-based Q-learning ACB solutions. More specifically, we explicitly take the energy consumption of the \(N_i^s\) successful MTDs into account and define the immediate reward of the \(i\)-th RAO as a weighted sum between the number of successful MTDs whose delay is below \(D_{thresh}\) and that number of successful MTDs whose energy consumption is below \(E_{thresh}\), normalized by the number of preambles \(R\). The reward in the \(i\)-th RAO, denoted by \(r_i\), is thus

\[
r_i = \frac{\alpha N_i^{s,E \leq E_{thresh}} + (1 - \alpha) N_i^{s,D \leq D_{thresh}}}{R}
\]

\[\text{(10)}\]

where \(\alpha \in [0, 1]\) is the weight indicating the relative importance of energy performance at the MTDs. Similar to \(o_i\), \(r_i\) is also stored in the eNB’s replay memory once observed.

Note that the information on the access delay and energy consumption of the successful MTDs in an RAO can be obtained by requiring these MTDs to report their delays (in the unit of subframes) and the number of subframes they spent in the Tx and Rx states via the subsequent Msg3. If the network is to be trained online via interaction with an actual system, this information may not be available to the eNB before the next RAO as per our assumption because the successful MTDs do not send their Msg3 immediately after an RAO. However, given an expected arrival distribution, the network can always be trained offline via interaction with computer simulation of the system where such information is available instantly after each simulated RAO. After the offline training, the best-performed set of weights can then be extracted and used as the starting point for online training. This paper only considers the offline training phase as in line with most Q-learning-based ACB solutions. The problems of delayed information and mismatched arrival distribution in online training will be left for future studies.

We use an FFN with 3 fully connected hidden layers, each contains 128 nodes, as the online network in the Duelling DQN architecture. We note that the \(Q^*(s, a)\) are not functions.
of time and therefore not necessarily be approximated by DNN types that are designed to capture temporal correlation of the input, such as GRU networks. The FFN was chosen because it achieves good end results despite not having any temporal characteristics (as per section III-B) and is easy to train. Besides, the temporal correlation between the RAOs has been partly encoded in the input states themselves since a state is defined as a window of observations across $T_{\text{win}}$ consecutive RAOs. This online FFN network is then trained offline using the Adam optimizer [21] in a series of simulated “episodes”. An episode starts at the same time as the arrival period of $T_{\text{arr}}$ seconds and finishes when all $N$ MTDs have either succeeded in accessing the cell or blocked after undergoing $N_{\text{trans}}$ preamble transmission failures, at which point the states of all MTDs are reset, and a new episode starts. The learning rate of Adam is initially $\lambda_{\text{init}}$, and is halved every time the reward accumulated over the RAOs of an episode (henceforth referred to as episode reward) has not increased for 50 consecutive episodes until it reaches a minimum value of $\lambda_{\text{min}}$. Also, during training, the network parameters $(\theta, \theta^V, \theta^A)$ are backup once every 10000 RAOs. Note that for online training, the Q-agent may not be able to perform gradient descent updates of the online network every RAO since the computation time required may exceed the RAO periodicity. However, since we are considering offline training, the online network can always be updated on a per simulated RAO basis.

In the $i$-th RAO of the offline training phase, the Q-agent observes the system’s state $s_i$ and makes a decision following the well-known $\epsilon$-greedy strategy where it chooses the most promising action (index of a barring pair) according to the current estimates, i.e., $a_i = \arg \max_{a_i} Q(s_i, a_i; \theta)$ with probability $1 - \epsilon$ and a random action in the set $A$ with probability $\epsilon$. The $\epsilon$ parameter is set to 1 at first but is gradually annealed to a minimum value of 0.1 within the first 1 million simulated RAOs during training. This strategy aims to provide a tradeoff between exploration and exploitation during the limited number of RAOs of the training phase. The training process is summarized in Algorithm 1.

**Algorithm 1 Offline Training Procedure for the Proposed ACB Scheme**

```latex
\begin{algorithm}
\caption{Offline Training Procedure for the Proposed ACB Scheme}
\begin{algorithmic}[1]
\For{$i = 1$ to $T_{\text{train}}$} \\
1: Observe the state $s_i$ at the beginning of $i$-th RAO; \\
2: With probability $\epsilon$, select a random $a_i \in A$; otherwise select the best $a_i$ according to the current estimate $a_i = \arg \max_{a_i} Q(s_i, a_i; \theta, \theta^V, \theta^A)$; \\
3: Broadcast the barring pair indexed by $a_i$ via the SIB2; \\
4: Observe $D_i, E_i, N_i^I, E_i$, and the immediate reward $r_i$; \\
5: Create the observation tuple $o_i = (N_i^I, N_i^E, D_i, E_i, a_i)$ and store $o_i, a_i, r_i$ into the memory; \\
6: Sample a random batch of experiences from memory, where each experience is of the form $(s_j, a_j, r_j, s_{j+1})$, and each state $s_j$ is a window of consecutive observations $s_j = [o_j_{-T_{\text{win}}+1}, \ldots, o_j]$; \\
7: \For{each experience $j$ in the batch} \\
8: \State $y_j^\text{pred} = Q(s_j, a_j; \theta, \theta^V, \theta^A)$; \\
9: \State $y_j^\text{target} = r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta^\text{target}, \theta^V, \theta^A, \theta^A_\text{target})$; \\
10: \State Calculate the sample loss $L_j(\theta, \theta^V, \theta^A) = (y_j^\text{pred} - y_j^\text{target})^2$; \\
11: \EndFor
12: Based on $s_j, a_j$, and $L(\theta, \theta^V, \theta^A)$ of the experiences, perform a stochastic gradient descent update on $(\theta, \theta^V, \theta^A)$ following Adam’s update rules; \\
13: \If{$i \mod T_{\text{copy}} = 0$} \\
14: \State Overwrite target network parameters $(\theta^\text{target}, \theta^V, \theta^A, \theta^A_\text{target})$ with current online network parameters $(\theta, \theta^V, \theta^A)$; \\
15: \EndIf
\EndFor
\end{algorithmic}
\end{algorithm}
```

**TABLE 1. Training hyperparameters.**

| Hyperparameters | Values |
|-----------------|--------|
| Discount factor | $\gamma = 0.9$ |
| Replay memory size | $T_{\text{mem}} = 50000$ |
| Observation window size | $T_{\text{win}} = 20$ |
| Experience batch size | 32 |
| Initial and minimum exploration rate | $\epsilon_{\text{init}} = 1, \epsilon_{\text{min}} = 0.1$ |
| Adam learning rate $\lambda$ | $\lambda_{\text{init}} = 10^{-4}, \lambda_{\text{min}} = 2.5 \times 10^{-4}$ |
| Target network update frequency | $T_{\text{copy}} = 25000$ |
| Weight of energy | $\alpha = 0, 0.1, 0.3, 0.5, 0.7$ |

III. SIMULATION RESULTS AND DISCUSSION

A. SIMULATION SETUP

In this section, computer simulations are performed to compare the effectiveness of the proposed solution to the dynamic ACB solution in [14], where exponentially growing backoff and barring times are used. The hyperparameters of the training procedure of our scheme are detailed in Table 1. The environment settings used for both training and verification, on the other hand, are detailed in Table 2. The delay tolerance $D_{\text{thresh}}$ is set to 1000 subframes, i.e., 1 second. The limit $E_{\text{thresh}}$ on the energy consumption at an MTD, meanwhile, is chosen to be equal to the consumption level when the MTD spends 2 subframe in Tx state, $2 \times W_{\text{RAR}} + 2$ subframes in Rx state, and $D_{\text{thresh}} = (4 + 2 \times W_{\text{RAR}})$ subframes in idle state, where $W_{\text{RAR}}$ is the length of the RAR window in the unit of subframes. This corresponds to the case where the MTDs succeed in its second preamble attempt and receives the acknowledgment in the last subframe of the RAR window, all at an exact delay of $D_{\text{thresh}}$. Note that given the power levels in Table 2, a Tx or Rx subframe can be traded for 2000 idle subframes, which implies a way to tradeoff between access delay and energy consumption. It is the Q-agent’s task to learn how to control the barring pair to perform the tradeoff efficiently given a certain $\alpha$ in (10).
B. RESULTS AND DISCUSSIONS

After the offline training phase, we pick the episodes whose reward is among the top 1% of all episodes and extract the \((\theta, \theta^V, \theta^A)\) sets that were backup during or close to those episodes to evaluate and report the best results among them in this section. For a fair comparison, we compare the reported results with those obtained in the case of perfect control, where the eNB always knows the exact number of backlogged MTDs and can thus select a corresponding \(p_{ACB}\) such that the average number of MTDs per RAO is equal to the number of available preambles \(R\), in the single priority network setting of section III-A of [14]. This reference scheme also assumes exponentially growing \(T_{BO}\) and \(T_{ACB}\) at the MTDs. More specifically, in the unit of milliseconds, \(T_{BO}\) is a random number in \([0, 10 \times 2^P - 1]\) and \(T_{ACB} = (0.7 + 0.6 \times \text{rand}) \times 2^N_{barring}\), where \(P_t\) and \(N_{barring}\) are the number of times an MTD has transmitted a preamble and the number of failed ACB checks at the MTD, respectively.

The ratios of the numbers of successful MTDs that also meet the delay and energy thresholds to the total number \(N\) of MTDs, respectively denoted by \(r_{del}\) and \(r_{ene}\), are shown in Table 3 for the two schemes. It is seen that compared to the reference scheme, ours with different \(\alpha\) can achieve comparable \(r_{del}\) but at a significantly lower energy cost as suggested by the much higher \(r_{ene}\). For instance, the proposed scheme at \(\alpha = 0.1\) can achieve a similar level of \(r_{del}\) while outperforming the reference in terms of \(r_{ene}\) by a hefty margin of 16%. To further prove that such energy gain comes from the fact that MTDs are not forced to update \(p_{ACB}\) too frequently and not simply from reduced contention levels, we modify the reference scheme so that the average number of MTDs per RAO is kept at \(0.7 \times R\) instead of \(R\). The corresponding results are shown in the last column of Table 3. It is clear that while this modification helps to boost the reference scheme’s energy efficiency, the \(r_{ene}\) metric is still far lower than that of the proposed scheme. This has proven that \(p_{ACB}\) updates are indeed a major source of energy consumption at the MTDs, and that dynamic control of both \(p_{ACB}\) and \(T_{ACB}\) is crucial for simultaneously improving the network’s delay and energy performance in bursty access scenarios.

In terms of average delay, the reference scheme (second-to-last column) poses better performance. Nevertheless, the data should be relevant as long as the MTDs can obtain access rights within the delay threshold \(D_{\text{thresh}}\), and the metric \(r_{del}\) is thus more relevant than the average access delay. On the contrary, the average energy consumption of successful MTDs is as crucial as \(r_{ene}\) since it dictates the amount of unrecoverable energy spent by the battery-limited devices. Here, as foreshadowed by a much higher \(r_{ene}\), successful MTDs in the proposed scheme consumes anywhere from 25% to 42% less energy on average than those in the reference scheme depending on \(\alpha\), which is a remarkable improvement. When the blocking ratio, defined as the ratio between the number of MTDs that cannot successfully access the network before exceeding \(N_{PTmax}\) to the total number of MTDs \(N\), is concerned, the two schemes perform comparably with less than 0.5% of the population getting blocked.

Looking at the proposed scheme alone, it is obvious that as the energy prioritization parameter \(\alpha\) is increased, the achieved \(r_{ene}\) and \(r_{del}\) tend to increase and decrease, respectively. This is expected and implies that by adjusting \(\alpha\),
the proposed scheme can efficiently realize the tradeoff between access delay and energy consumption. To support this claim, we report the smoothed curves of $r_{\text{del}}$ and $r_{\text{ene}}$ observed during the training episodes for different values of $\alpha$ in Fig. 3. As seen, there is a definite trend showing that the Q-agent is driven towards solutions that yield lower $r_{\text{del}}$ (and higher $r_{\text{ene}}$) for higher values of $\alpha$ and vice versa. Note that the curves’ occasional fluctuation is a typical behavior in DNN-based Q-learning as there is no guarantee that the agent will converge at the optimal solutions.

It is also interesting to take a detailed look at the solutions discovered by the agent for different $\alpha$ settings. We have thus plotted in Fig. 4 the $p_{\text{ACB}}$ and $T_{\text{ACB}}$ control that the Q-agent comes up with using the same sets of $(\theta, \theta^V, \theta^A)$ that were used to obtain the results reported in Table 3. The vertical axis on the right represents the values of $T_{\text{ACB}}$ normalized by 4000ms and $p_{\text{ACB}}$. Here, the immediately visible feature is that during the peak period (from around RAO 500 to RAO 1000), the $p_{\text{ACB}}$ is kept relatively high at 0.7 or above for all cases, which is in stark contrast with
the reference scheme where $p_{ACB}$ may reach as low as 0.25. More importantly, the high barring factors are coupled with a very high $T_{ACB}$ which mostly varies between 3400ms to 4000ms, except for the case of $\alpha = 0$, i.e., when energy consumption does not matter. This implies that most of the MTDs will pass the barring check and are allowed to partake in the busy RAOs for a smaller success chance but a high possibility to meet the access delay threshold $D_{thresh}$. The smaller fraction of the population who fails the check, on the other hand, is immediately deferred until much later when the new traffic has settled down. While those devices will certainly miss the delay deadline, they also save a notable amount of energy thanks to not having to update the barring parameters frequently and a higher chance of a successful preamble transmission. The results in Table 3 indeed prove that the combination of a high $p_{ACB}$ and a high $T_{ACB}$ during the peak period is an overall more efficient strategy compared to greatly lowering the $p_{ACB}$ to keep the average number of MTDs close to $R$.

Additionally, when $\alpha$ is increased, i.e., energy performance is more emphasized, the Q-agent chooses to 1) slightly lower both $p_{ACB}$ and $T_{ACB}$ during the peak period, and 2) keep $T_{ACB}$ at those lower values for longer even after the peak has elapsed. As a result, the number of contending MTDs is consistently maintained at lower levels over a longer period. This reduces the contention level without requiring the MTDs to update the barring pair too frequently to achieve a higher $p_{\text{crit}}$. At the same time, the PRACH is still kept busy enough to avoid severe under-utilization.

Finally, we note that these best $(\theta, \theta^V, \theta^A)$ sets are found relatively late, as shown by the last row in Table 3. However, other good sets that yield performance levels (in terms of average episode reward) within 96-98% of the reported ones are also found frequently during training. Furthermore, the Q-agent actually starts to make reasonable barring parameters selections quite early, as shown in Fig. 5. It is evident that depending on $\alpha$, usable sets achieving performance levels within 90% those of the reported sets in Table 3 are found as early as after 0.75 million simulated RAOs.

### IV. CONCLUSION AND FUTURE DIRECTIONS

In this paper, we proposed a DNN-based dynamic ACB solution that explicitly takes energy consumption into consideration and controls both the barring factor $p_{ACB}$ and the mean barring time $T_{ACB}$ in order to improve the energy performance of the MTDs in massive bursty mMTC access scenarios. Computer simulations show that compared to conventional schemes that only control the barring factor, the proposed scheme can achieve a similar delay performance at a significantly lower energy cost. Furthermore, by adjusting a parameter representing the importance of energy performance in the reward function, our scheme can efficiently
realize the tradeoff between access delay and energy consumption. In future works, we aim to investigate the impact of the delayed reward, the computational latency of the gradient descent update procedure, and the mismatched arrival distribution on the performance of the proposed scheme when the best weights extracted during the offline training phase are used as the initial weights for online training. The designs that can overcome such issues to further improve the performance will also need to be considered.

REFERENCES

[1] IMT Vision—Framework and Overall Objectives of the Future Development of IMT for 2020 and Beyond, ITU, Geneva, Switzerland, Sep. 2015.

[2] I. Leyva-Mayorga, L. Tello-Oquendo, V. Pla, J. Martinez-Bauset, and V. Casares-Giner, “On the accurate performance evaluation of the LTE-A random access procedure and the access class barring scheme,” IEEETrans. Wireless Commun., vol. 16, no. 12, pp. 7785–7799, Dec. 2017.

[3] S. Duan, V. Shah-Mansouri, and W. S. Wong, “Dynamic access class barring for M2M communications in LTE networks,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2013, pp. 4747–4752.

[4] H. Wu, C. Zhu, R. J. La, X. Liu, and Y. Zhang, “FASA: Accelerated S-ALOHA using access history for event-driven M2M communications,” IEEE/ACM Trans. Netw., vol. 21, no. 6, pp. 1904–1917, Dec. 2013.

[5] H. Jin, W. T. Toor, B. C. Jung, and J.-B. Seo, “Recursive pseudo-Bayesian access class barring for M2M communications in LTE systems,” IEEE Trans. Veh. Technol., vol. 66, no. 9, pp. 8595–8599, Sep. 2017.

[6] S. Duan, V. Shah-Mansouri, Z. Wang, and W. S. Wong, “D-ACB: Adaptive congestion control algorithm for bursty M2M traffic in LTE networks,” IEEE Trans. Veh. Technol., vol. 65, no. 12, pp. 9847–9861, Dec. 2016.

[7] L. Tello-Oquendo, V. Pla, I. Leyva-Mayorga, J. Martinez-Bauset, V. Casares-Giner, and L. Guijarro, “Efficient random access channel evaluation and load estimation in LTE-A with massive MTC,” IEEE Trans. Veh. Technol., vol. 68, no. 2, pp. 1998–2002, Feb. 2019.

[8] L. Tello-Oquendo, J.-R. Vidal, V. Pla, and L. Guijarro, “Dynamic access class barring parameter tuning in LTE-A networks with massive M2M traffic,” in Proc. 17th Annu. Medit. Ad Hoc Netw. Workshop (Med-Hoc-Net), Jun. 2018, pp. 1–8.

[9] M. Tavana, V. Shah-Mansouri, and V. W. S. Wong, “Congestion control for bursty M2M traffic in LTE networks,” in Proc. IEEE Int. Conf. Commun. (ICC), Jun. 2015, pp. 5815–5820.

[10] L. Tello-Oquendo, I. Leyva-Mayorga, V. Pla, J. Martinez-Bauset, J.-R. Vidal, V. Casares-Giner, and L. Guijarro, “Performance analysis and optimal access class barring parameter configuration in LTE-A networks with massive M2M traffic,” IEEE Trans. Veh. Technol., vol. 67, no. 4, pp. 3505–3520, Apr. 2018.

[11] J. Moon and Y. Lim, “A reinforcement learning approach to access management in wireless cellular networks,” Wireless Commun. Mobile Comput., vol. 2017, pp. 1–7, May 2017.

[12] L. Tello-Oquendo, D. Pacheco-Paramo, V. Pla, and J. Martinez-Bauset, “Reinforcement learning-based ACB in LTE-A networks for handling massive M2M and H2H communications,” in Proc. IEEE Int. Conf. Commun. (ICC), May 2018, pp. 1–7.

[13] C. J. Watkins and P. Dayan, “Technical note: Q-learning,” Mach. Learn., vol. 8, no. 3–4, pp. 279–292, May 1992.

[14] Z. Chen and D. B. Smith, “Heterogeneous machine-type communications in cellular networks: Random access optimization by deep reinforcement learning,” in Proc. IEEE Int. Conf. Commun. (ICC), May 2018, pp. 1–6.

[15] N. Jiang, Y. Deng, and A. Nallanathan, “Deep reinforcement learning for discrete and continuous massive access control optimization,” in Proc. IEEE Int. Conf. Commun. (ICC), Jun. 2020, pp. 1–7.

[16] N. Jiang, Y. Deng, A. Nallanathan, and J. A. Chambers, “Reinforcement learning for real-time optimization in NB-IoT networks,” IEEE J. Sel. Areas Commun., vol. 37, no. 6, pp. 1424–1440, Jun. 2019.

[17] Study on RAN Improvements for Machine-Type Communications, document TR 37.868 V11.0.0, 3GPP, Sep. 2011.

[18] D. Bertsekas, Reinforcement Learning and Optimal Control. Belmont, MA, USA: Athena Scientific, 2019.

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

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

[21] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980

[22] M. Gerasimenko, V. Petrov, O. Galinia, S. Andreew, and Y. Koucheryavy, “Energy and delay analysis of LTE-advanced RACH performance under MTC overload,” in Proc. IEEE Globecom Workshops (GLOBECOM-WS), Dec. 2012, pp. 1632–1637.

ANH-TUAN H. BUI (Graduate Student Member, IEEE) received the B.E. degree in communications engineering from the Hanoi University of Science and Technology, Vietnam, in 2016, and the M.S. degree in computer network systems from The University of Aizu, Japan, in 2018, where he is currently pursuing the Ph.D. degree. His research in Japan is funded by the Japanese Government Scholarship (Monbukagakusho). His research interests include protocol designs, system modeling, and performance evaluation for next generation (5G) wireless networks.

ANH T. PHAM (Senior Member, IEEE) received the B.E. and M.E. degrees in electronics engineering from the Hanoi University of Technology, Vietnam, in 1997 and 2000, respectively, and the Ph.D. degree in information and mathematical sciences from Saitama University, Japan, in 2005. From 1998 to 2002, he was with NTT Corporation, Vietnam. Since 2005, he has been a Faculty Member with The University of Aizu, where he is currently a Professor and the Head of the Computer Communications Laboratory, Division of Computer Engineering. He has authored/coauthored over 200 peer-reviewed articles on these topics. His research interest includes broad areas of communication theory and networking with a particular emphasis on modeling, design, and performance evaluation of wired/wireless communication systems and networks. He is a member of IEICE and OSA.