Deep Reinforcement Learning for Real-Time Optimization in NB-IoT Networks

Preprint · December 2018

CITATIONS
0

READS
519

4 authors:

Nan Jiang
Toshiba Research Europe Limited
25 PUBLICATIONS **257** CITATIONS
SEE PROFILE

Yansha Deng
King's College London
178 PUBLICATIONS **1,907** CITATIONS
SEE PROFILE

Arumugam Nallanathan
Queen Mary, University of London
608 PUBLICATIONS **12,600** CITATIONS
SEE PROFILE

Jonathon Chambers
Leicester University & Harbin Engineering University, China
644 PUBLICATIONS **10,359** CITATIONS
SEE PROFILE

Deep Reinforcement Learning for Real-Time Optimization in NB-IoT Networks

Nan Jiang, *Student Member, IEEE*, Yansha Deng, *Member, IEEE*, Arumugam Nallanathan, *Fellow, IEEE*, and Jonathon A. Chambers, *Fellow, IEEE*,

Abstract

NarrowBand-Internet of Things (NB-IoT) is an emerging cellular-based radio access technology, which offers a range of flexible configurations for different coverage enhancement (CE) groups to provide reliable uplink connections for massive IoT devices with diverse data traffic. To optimize the number of served IoT devices, the uplink resource configurations need to be adjusted in real-time according to the dynamic traffic, this brings the challenge of how to select the configurations at the Evolved Node B (eNB) in the multiple CE groups scenario with high-dimension and interdependency. To compare with the load estimation based conventional uplink resource configuration approach (LE-URC), we first develop real-time optimization approaches for configuration selection based on the tabular Q-learning (tabular-Q), the linear approximation based Q-learning (LA-Q), and the deep neural network based Q-learning (DNN-Q) in the single CE group single scenario. Our results show that the tabular-Q, LA-Q, and DNN-Q based approaches considerably outperform the LE-URC approach in terms of the number of served IoT devices in the single CE group scenario, which indicates that LA-Q and DNN-Q can be good alternatives for tabular-Q to achieve almost the same performance with much less training time. We further advance LA-Q and DNN-Q via actions simplification approach (denoted as AS-DNN-Q and AS-LA-Q) and via centralized multi-agent cooperation approach (denoted as CMC-DNN-Q and CMC-LA-Q) for the real-time multi-parameters optimization in multiple CE groups scenario, thereby solve the problem that Q-learning agents do not converge in high-dimensional configurations. The superiority of the advanced Q-learning based approaches over the conventional LE-URC approach significantly improves with the increase of configuration dimensions, and the CMC-DNN-Q approach outperforms the other approaches in both the number of served IoT devices and the training speed.

the

I. INTRODUCTION

To effectively support the emerging massive Internet of Things (mIoT) ecosystem, the 3rd generation partnership project (3GPP) partners have standardized a new radio access technology, namely NarrowBand-IoT (NB-IoT) [1]. NB-IoT is expected to provide reliable wireless access for IoT devices with various

N. Jiang, and A. Nallanathan are with the School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, UK (e-mail: {nan.jiang, a.nallanathan}@qmul.ac.uk).

Y. Deng is with the Department of Informatics, King's College London, London WC2R 2LS, UK (e-mail: yansha.deng@kcl.ac.uk).

J. A. Chambers is with the Department of Engineering, University of Leicester, Leicester LE1 7RH, UK (e-mail: jonathon.chambers@le.ac.uk).

types of data traffic, and to meet the requirement of extended coverage. As most mIoT applications favor delay-tolerant data traffic with small size, such as data from alarms, and meters, monitors, the key target of NB-IoT design is to deal with the sporadic uplink transmissions of massive IoT devices [2].

NB-IoT is built from legacy Long-Term evolution (LTE) design, but only deploys in a narrow bandwidth (180 KHz) for coverage enhancement (CE) [3]. Different from the legacy LTE, NB-IoT only defines two uplink physical channel resources to perform all the uplink transmission, including the Random Access (RACH) resource (i.e., using NarrowBand Physical Random Access CHannel, a.k.a. NPRACH) for RACH preamble transmission, and the data resource (i.e., using NarrowBand Physical Uplink Shared CHannel, a.k.a. NPUSCH) for control information and data transmission. To support various traffic with different coverage requirements, NB-IoT supports up to three CE groups sharing the uplink resource in the same band. The evolved Node B (eNB) flexibly balances RACH resource and data resource for each CE group in order to serve more IoT devices. This inevitably brings a real-time optimization problem that is, how to choose suitable configurations to achieve the maximum number of served IoT devices at the eNB for different traffic scenarios.

In NB-IoT networks, existing research either focused on the resource allocation of the RACH procedure [4, 5], or that of the data transmission [6, 7]. In the studies of RACH procedure in NB-IoT, the access success probability of whole network was statistically optimized in [4] using exhaustive search, and the authors in [5] studied the fixed-size data resource scheduling for various resource requirements. Except for the studies on the RACH procedure, [6] presented a uplink data transmission time slot and power allocation scheme to optimize the overall channel gain, and [7] proposed a link adaptation scheme, which dynamically selects modulation and coding scheme level, and the repetition value according to the acknowledgment/negative acknowledgment feedback to reduce the uplink data transmission block error ratio.

Unfortunately, dynamically configuring the RACH as well as data transmission resource in real-time is an unsolved untreated problem in NB-IoT, and it faces the following two main challenges: 1) the optimization problem is with high-dimensional complexity due to the massive interdependency of the controllable configurations among multiple CE groups over time; 2) the wireless communication procedure consists of massive hidden information, such as the nature of stochastic traffic and unobservable channels (i.e., random collision, and effects of physical radio including path-loss and fading). The real-time resource configuration of RACH procedure and/or data transmission have been studied in [8, 9], but its configuration dimension is small. In [8], the authors proposed a dynamic access class barring (ACB) scheme to optimize the number of served IoT devices for the Poisson traffic. The ACB factor was adjusted in real-time based on the Poisson load estimation approach, which cannot be used for other traffic models. In [9], the authors

designed a dynamic ACB scheme based on the model-free load estimation approach using the observed number of the collided preambles. However, the schemes in [8, 9] can only optimize the number of served IoT devices in the current time slot without consideration of future performance. Furthermore, these schemes are only capable of small configuration dimension, and cannot address the uplink resource configuration in the multi-parameters multiple CE groups scenario.

To tackle this, the reinforcement learning (RL) is proposed as a promising solution, where the RL agent (i.e., implemented at the eNB) automatically updates the uplink resource configuration by interacting with the environment (i.e., the communication procedures in NB-IoT). In the domain of wireless communication, distributed RL based on tabular Q-learning has been proposed in [10–13]. In [10–12], the authors studied distributed Q-learning in slotted-Aloha networks, where each device learns how to avoid collisions by finding a proper time slot to transmit packets. In [13], the authors implemented Q-learning agents (Q-agents) at the relay nodes for cooperatively selecting the its transmit powers and transmission probabilities to optimize the total number of useful received packets per consumed energy. Centralized RL has also been studied in [14–16], where the RL agent was implemented at the base station site. In [14], a learning-based scheme was proposed for radio resource management in multimedia wide-band code-division multiple access systems to improve spectrum utilization. In [15, 16], the authors studied the ACB scheme in cellular networks, and implemented a tabular Q-agent at an eNB aiming at selecting the optimal ACB factor to optimize the access success probability of RACH procedure.

Unfortunately, the Q-learning approaches proposed in [10–12, 14–16] set the reward function as constants corresponding to different regimes of observation, following the RL for Games in computer science community [17, 18]. However, this constant reward value does not have flexibility, and may need to be changed for the same environment with different parameter settings. Furthermore, the cetralized tabular-Q learning framework in [15, 16] cannot be used to solve the multi-parameters multiple groups optimization problem in uplink resource configuration of NB-IoT networks, due to that they are incapable in dealing with the high-dimensional variable selection. Besides, whether their proposed RL-based resource configuration approaches [15, 16] outperform the conventional resource configuration approaches [8, 9] is still unknown. In this paper, we develop six RL approaches implemented at the eNB to optimize the number of served IoT devices via the real-time uplink resource configuration in NB-IoT networks, which are compared with the conventional optimal uplink resource allocation approach to showcase the efficiency of our proposed approaches. The contributions can be summarized as follows:

- We first model the uplink communication procedure in NB-IoT by taking into account the CE group selection, the RACH procedure, and the uplink resource scheduling for data transmission. To dynam-

ically optimize the number of served IoT devices in real-time, we propose a novel Q-learning based uplink resource configuration framework.

- To compare with the conventional optimal resource allocation approach based on real-time load estimation (LE-URC), we develop the uplink resource configuration approaches, namely, tabular Q-learning (tabular-Q), linear approximation based Q-learning (LA-Q), and deep neural networks based Q-learning (DNN-Q) in the single parameter single CE group scenario to show the capability and efficiency of different function approximators. We focus on optimizing the number of preambles in real-time to achieve the maximum number of served IoT devices.
- To support diverse IoT devices with different coverage requirements, we then study the multi-parameters multiple CE groups scenario, where the number of RACH periods, the repetition value, and the number of preambles in each RACH period are optimized for each CE group to achieve the maximum number of served IoT devices at the eNB, resulting in the high-dimensional configuration challenge. To solve it, we develop two actions simplified approaches based on LA-Q and DNN-Q, namely, AS-LA-Q and AS-DNN-Q, which guarantee convergence capability by sacrificing certain accuracy in the resource configuration. To guarantee configuration selection accuracy, we further develop centralized multi-agent cooperation based on DNN-Q (CMC-DNN-Q) to divide high-dimensional configurations into multiple parallel sub-tasks.
- In the single parameter single CE group scenario, our results show that the number of served IoT devices of the NB-IoT network with tabular-Q, LA-Q, and DNN-Q approaches largely outperform that of the conventional LE-URC approach, which indicates that LA-Q and DNN-Q can be good alternatives for tabular-Q to achieve almost the same number of served IoT devices using much less training time. In the multi-parameters multiple CE groups scenario, the superiority of Q-learning based approaches over the conventional LE-URC approach significantly improves, and CMC-DNN-Q outperforms all other approaches in terms of the number of served IoT devices and the training speed.

The rest of the paper is organized as follows. Section II illustrates the network model and the communication procedure. Section III illustrates the problem formulation and preliminary. Section IV provides the conventional and the Q-leaning based uplink resource configuration approaches in the single parameter single CE group scenario. Section V presents the advanced Q-learning based approaches for uplink resource configuration in the multi-parameters multiple CE groups scenario. Section VI presents the numerical results. Finally, Section VII concludes the paper.

II. SYSTEM MODEL

As illustrated in Fig. 1(a), we consider a NB-IoT network composed of an eNB located at the center, and a set $\mathcal{D}$ of IoT devices, which are randomly located in an area of the plane $\mathbb{R}^2$, and remain spatially static once deployed. The eNB is unaware of the status of these IoT devices, hence no uplink channel resource is scheduled to them. In each IoT device, uplink data is generated according to a random inter-arrival process, and any backlogged IoT device tries to establish synchronization with the eNB in order to request uplink channel resource for data transmission.

To capture the effects of the physical radio, we consider the standard power-law path-loss model that the path-loss attenuation is $u^{-\eta}$, with the propagation distance u and the path-loss exponent η. The system is operated in a Rayleigh flat-fading environment, where the channel power gains h are exponentially distributed (i.i.d.) random variables with unit mean.

A. NB-IoT Access Network

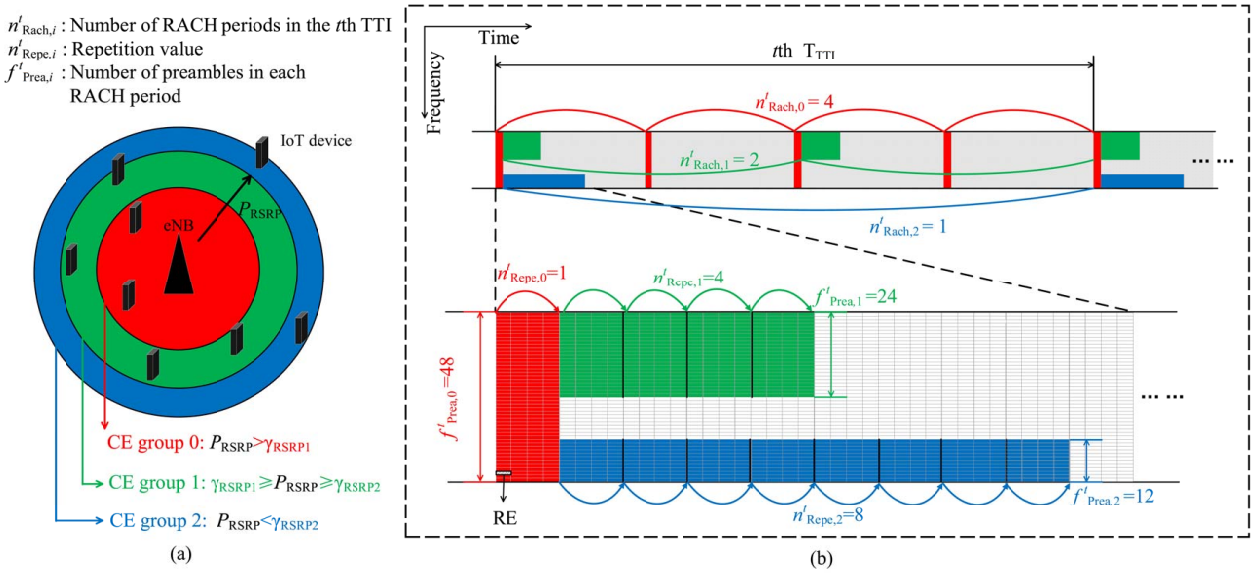

Fig. 1: (a) Illustration of system model; (b) Uplink channel frame structure.

To provide reliable connections with extended coverage, the NB-IoT standard defined up to three CE groups as shown in Fig. 1(a). When an IoT device requests an uplink transmission service within the tth Transmission Time Interval (TTI), it first needs to confirm its allocated CE group, which is determined according to the following rule:

$$\begin{cases} \text{CE group 0}, & \text{if } P_{\text{RSRP}} > \gamma_{\text{RSRP1}}, \\ \text{CE group 1}, & \text{if } \gamma_{\text{RSRP1}} \geq P_{\text{RSRP}} \geq \gamma_{\text{RSRP2}}, \\ \text{CE group 2}, & \text{if } P_{\text{RSRP}} < \gamma_{\text{RSRP2}}, \end{cases} \tag{1}$$

where γ_{RSRP1} and γ_{RSRP2} are the Reference Signal Received Power (RSRP) thresholds that will be detailed in Section II.B.2), and P_{RSRP} is the received power of the broadcast signal.

After that, each IoT device excutes RACH procedure to establish Radio Recouce Control (RRC) connection with the eNB using the configured RACH resource for its associated CE group. We consider the contention-based RACH rather than the contention-free RACH, due to its relevance for IoT traffic [19]. The contention-based RACH procedure consists of four steps, where an IoT device transmits a randomly selected preamble to initial RACH procedure in step 1, and exchanges control information with the eNB in the next three steps[1] [20]. The RACH procedure may fail due to the following two reasons: 1) the eNB cannot decode a received preamble due to low Signal-to-Noise Ratio (SNR) in Step 1, namely, *SNR outage*; 2) a *collision* occurs when two or more IoT devices select the same preamble in Step 1. To mitigate the SNR outage, preambles of its associated CE group are repeated in each RACH period according to its repetition value $n^t_{\text{Repe},i}$ [21]; to mitigate the collision, one RACH period is repeated $n^t_{\text{Rach},i}$ times in each TTI, which consists of $f^t_{\text{Prea},i}$ number of preambles in the frequency domain[2] as shown in Fig. 1 (b) [21].

B. NB-IoT Modeling Details

Fig. 2 presents the communication procedure from the view of an IoT device, which consists of the four stages that are explained in the following four subsections to introduce the system model.

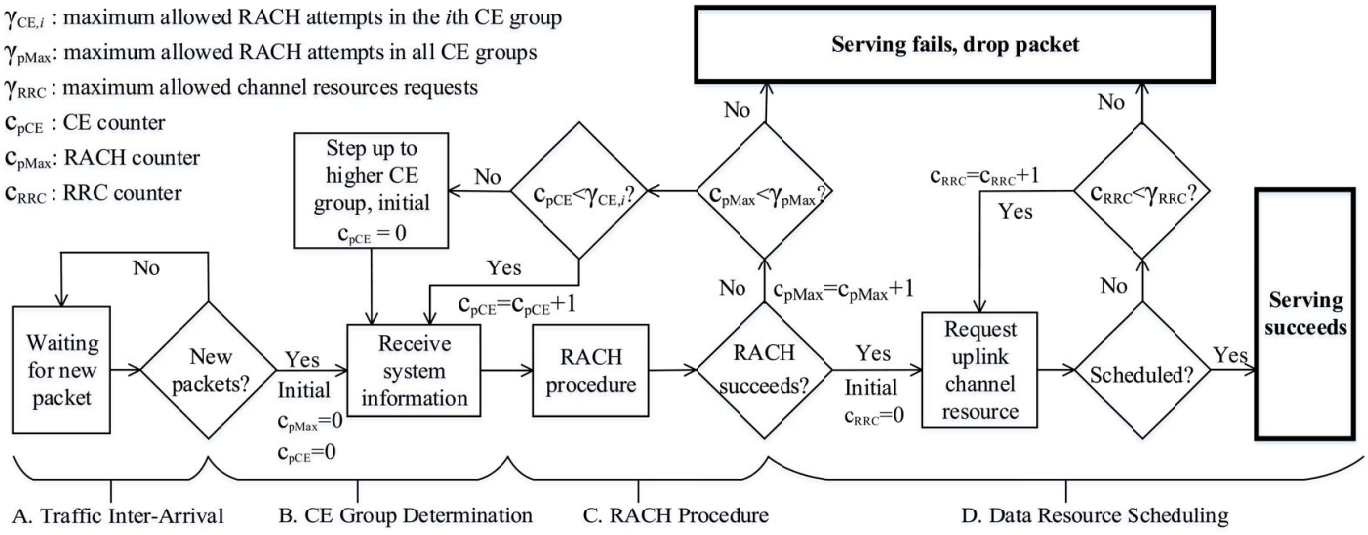

Fig. 2: Communication Procedure from the view of an IoT device.

[1]Same as [8, 9], we assume that the step 2, 3, and 4 of RACH procedure are always successful whenever the step 1 is successful.

[2]Different from legacy LTE [20], each specific preamble in NB-IoT is in one-to-one correspondence with each sub-carrier [21].

1) Traffic Inter-Arrival: We consider two types of IoT devices with different traffic models. The first type is periodical traffic due to periodic uplink reporting tasks, such as metering or environmental monitoring, and is the most common traffic model in NB-IoT networks [22]. The second type is bursty traffic that captures the complementary scenario in which a massive number of IoT devices tries to establish RRC connection with the eNB [23]. Due to the nature of slotted-Aloha, an IoT device can only transmit a preamble at the beginning of a RACH period, which means that IoT devices executing RACH in a RACH period comes from those who received a inter-arrival within the interval between with the last RACH period. For the periodical traffic, the first packet is generated using uniform distribution over T_{periodic} (ms), and then repeated every T_{periodic} ms. The packet inter-arrival rate measured in each RACH period at each IoT device is hence expressed by

$$\mu_{\text{period}} = \frac{T_{\text{TTI}}}{n^{t}_{\text{Rach},i}} \times \frac{1}{T_{\text{periodic}}}, \tag{2}$$

where $n^{t}_{\text{Rach},i}$ is the number of RACH periods in the tth TTI, $\frac{T_{\text{TTI}}}{n^{t}_{\text{Rach},i}}$ is the duration between neighboring RACH periods. The bursty traffic is generated within a short period of time T_{bursty} starting from a random time τ_0. The traffic instantaneous rate in packets in a period is described by a function $p(\tau)$ so that the packets arrival rate in the rth RACH period of the tth TTI is given by

$$\mu^{t,r}_{\text{bursty}} = \int_{\tau_{r-1}}^{\tau_r} p(\tau) d\tau, \tag{3}$$

where τ_r is the starting time of the rth RACH period in the tth TTI, $\tau_r - \tau_{r-1} = \frac{T_{\text{TTI}}}{n^{t}_{\text{Rach},i}}$, and the distribution $p(\tau)$ follows the Beta distribution given as [23, Section 6.1.1]

$$p(\tau) = \frac{\tau^{\hat{\alpha}-1}(T_{\text{bursty}} - \tau)^{\hat{\beta}-1}}{{T_{\text{bursty}}}^{\hat{\alpha}+\hat{\beta}-2}\text{Beta}(\hat{\alpha}, \hat{\beta})}, \tag{4}$$

and $\text{Beta}(\hat{\alpha}, \hat{\beta})$ is the Beta function [24].

2) CE Group Determination: Once an IoT device receives an inter-arrival packet, its associated CE group will be determined by comparing the RSRP thresholds $\{\gamma_{\text{RSRP1}}, \gamma_{\text{RSRP2}}\}$ to the received power of broadcast signal P_{RSRP} as given in Eq. (1) [25, 26]. The received power of broadcast signal P_{RSRP} is expressed as

$$P_{\text{RSRP}} = P_{\text{NPBCH}} u^{-\eta}, \tag{5}$$

where P_{NPBCH} is transmit power of eNB, u is the propagation distance, and η is the path-loss exponent. Note that P_{RSRP} is obtained by averaging the small-scale Rayleigh fading of the received power [27].

3) RACH Procedure: In RACH preamble transmission, a preamble is repeated with the Pseudo random hopping based on the current repetition time and the Narrowband Physical Cell ID, and in one repetition, a preamble consists of four symbol groups, which are transmitted with fixed size frequency hopping [1, 28, 29]. The event of preamble being successfully decoded S_{pd} occurs if *at least one preamble repetition*

succeeds, which only occurs if *all its four symbol groups being successfully decoded* [29]. This event is expressed as

$$S_{\text{pd}} \triangleq \bigcup_{j=1}^{n^t_{\text{Repe},i}} \Big(\bigcap_{k=1}^{4} \big\{ \text{SNR}^t_{\text{sg},j,k} \geq \gamma_{th} \big\} \Big), \tag{6}$$

where k is the index of symbol group in the jth repetition, $n^t_{\text{Repe},i}$ is the repetition value of the ith CE group in the tth TTI, $\big\{\text{SNR}^t_{\text{sg},j,k} \geq \gamma_{th}\big\}$ means that the preamble symbol group is successfully decoded when its received SNR $\text{SNR}^t_{\text{sg},j,k}$ above a threshold γ_{th}, and $\text{SNR}^t_{\text{sg},j,k}$ is expressed as

$$\text{SNR}^t_{\text{sg},j,k} = P_{\text{RACH},i} u^{-\alpha} h / \sigma^2. \tag{7}$$

In (7), u is the Euclidean distance between the IoT device and the eNB, α is the path-loss attenuation factor, h is the Rayleigh fading channel power gain from the IoT device to the eNB, σ^2 is the noise power, and $P_{\text{RACH},i}$ is the preamble transmit power in the ith CE group defined as

$$P_{\text{RACH},i} = \begin{cases} \min\ \{\rho u^{\eta},\ P_{\text{RACHmax}}\}, & i = 0, \\ P_{\text{RACHmax}}, & i = 1 \text{ or } 2. \end{cases} \tag{8}$$

where i is the index of CE groups, IoT devices in the CE group 0 ($i = 0$) transmit preamble using the full path-loss inversion power control [27], which maintains the received signal power at the eNB from IoT devices with different distance equalling to the same threshold ρ, u is the distance between the IoT device and the eNB, and P_{RACHmax} is the maximal transmit power of an IoT device. The IoT devices in the CE group 1 and group 2 always transmit preamble using the maximum transmit power [27].

As shown in the RACH procedure of Fig. 2, if a RACH procedure fails, the IoT device performs the next RACH attempts by transmitting a reselected preamble. In the 3GPP standard, RACH attempts are locally counted in a CE group using counter c_{pCE}, and, if the RACH attempts exceed the maximum allowed RACH attempts $\gamma_{\text{pCE},i}$, the IoT device reattempts an RACH procedure in a higher CE group [30]. The IoT device is allowed to attempt the RACH procedure no more than γ_{pMax} times, otherwise, the packet will be dropped.

4) Data Resource Scheduling: After the RACH procedure succeeds, the RRC connection is successfully established, and the eNB schedules resource from the data channel resource R^t_{DATA} to the associated IoT device for control information and data transmission. As shown in Fig 1(b), the total number of resource elements[3] (REs) reserved for uplink transmission within one TTI T_{TTI} (ms) is $R_{\text{Uplink}} = 48 \times (T_{\text{TTI}}/2)$, and the data resource R_{DATA} occupies all other uplink resource available after the configuration of RACH

$$R^t_{\text{DATA}} = R_{\text{Uplink}} - R^t_{\text{RACH}}, \tag{9}$$

[3]The uplink channel consists of 48 sub-carriers within 180 kHz bandwidth, where the basic resource allocation unit is the RE, which is composed of one time slot of 2 ms and one sub-carrier of 3.75 kH. Note that the NB-IoT also supports 12 sub-carriers with 15 kHz tone spacing for NPUSCH, but NPRACH only supports 3.75 kHz tone spacing [1].

where R^t_{RACH} is the uplink resource configured for RACH in the tth TTI expressed as

$$R^t_{\text{RACH}} = B_{\text{RACH}} \sum_{i=0}^{2} n_{\text{Rach},i} n_{\text{Repe},i} f_{\text{Prea},i}. \tag{10}$$

In (10), B_{RACH} is the required REs for one repetition in each RACH period, $n^t_{\text{Rach},i}$ is the number of RACH periods in each TTI, $n^t_{\text{Repe},i}$ is the repetition value, and $f^t_{\text{Prea},i}$ is the number of preambles in each RACH period.

The eNB successfully serves all the RRC connected IoT devices if the uplink resource R^t_{DATA} is sufficient, and this condition is expressed as

$$R^t_{\text{DATA}} \geq \sum_{i=0}^{2} r^t_{\text{DATA},i} \big(V^t_{\text{sp},i} + V^{t-1}_{\text{unsc},i} \big), \tag{11}$$

where $V^t_{\text{sp},i}$ is the number of IoT devices that RACH succeeds with the ith CE group in the tth TTI, $V^{t-1}_{\text{unsc},i}$ is the number of IoT devices that were unscheduled with the ith CE group in the $(t-1)$th TTI, and $R^t_{\text{Serve},i}$ is the required REs for serving one IoT device with the ith CE group expressed as

$$r^t_{\text{DATA},i} = B_{\text{DATA}} \times n^t_{\text{Repe},i}, \tag{12}$$

In (12), B_{DATA} is the number of required REs for serving single packet of one IoT device with one repetition[4], $n^t_{\text{Repe},i}$ is the repetition value for the ith CE group in the tth TTI. Note that $n^t_{\text{Repe},i}$ is the same for data transmission as well as the preamble transmission [1].

To allocate data resource, we consider a basic random scheduling strategy, where each request has the same priority being randomly handled. If the data resource is insufficient to serve all the requests in one TTI, the RRC connections between the unscheduled IoT devices and the eNB will be preserved, so that they do not need to initiate a RACH procedure again [20, 30]. Note that once the RRC connection is built, the IoT device will initiate an RRC counter c_{RRC} as shown in Fig. 2, and it can request data resource at most γ_{RRC} times. If RRC counter c_{RRC} exceeds γ_{RRC}, the IoT device will drop the packets, and declare that the network is unavailable [25].

III. Problem Formulation and Conventional Solutions

A. Problem Formulation

The objective of this work is to study how the eNB dynamically optimize $n^t_{\text{RA},i}$, $n^t_{\text{RPV},i}$, and f^t_i of the ith group in each TTI, in order to serve more IoT devices at the eNB over the long term. This brings a sequence that IoT devices execute communication procedure according to the system configuration within

[4]The basic scheduling unit of NPUSCH is resource unit (RU), which has two formats. NPUSCH format 1 (NPUSCH-1) is with 16 REs for data transmission, and NPUSCH format 2 (NPUSCH-2) is with 4 REs for carrying control information [3, 21].

the $(t-1)$th TTI, which generates an actual state S^{t+1}_{actual} informing the number of back-logged IoT devices n^t in the tth TTI; and then, the eNB obtains a belief state S^t from environment by observing the transmission reception from the 1th to $(t-1)$th TTI O^t; and finally, the eNB selects an action A^t based on the belief state S^t at the beginning of the tth TTI. Since the eNB is not aware of the communication experience of each IoT device and the actual state S^{t+1}_{actual}, this introduces a Partial Observation Markovian Decision Process (POMDP) problem, where the eNB can only observe the number of successfully served IoT devices V^t_{succ}, the number of unscheduled IoT devices V^t_{unsc}, the number of the collided preambles V^t_{cp}, the number of succeeded preambles V^t_{sp}, and the number of idle preambles V^t_{ip} in the tth TTI.

Denoting a policy π mapping from the state S^t to probabilities of selecting each configuration, we aim to solve the following real-time optimization problem:

$$(\text{P1}): \max_{\pi(n^t_{\text{Rach},0},\, n^t_{\text{Repe},0},\, f^t_{\text{Prea},0},\, n^t_{\text{Rach},1},\, n^t_{\text{Repe},1},\, f^t_{\text{Prea},1},\, n^t_{\text{Rach},2},\, n^t_{\text{Repe},2},\, f^t_{\text{Prea},2} \,\big|\, S^t)} \quad \sum_{k=t}^{\infty} \sum_{i=0}^{2} \gamma^k \mathbb{E}_\pi[V^k_{\text{succ},i}], \tag{13}$$

where k is the index of TTIs after the tth TTI, $\gamma \in [0, 1)$ is the discount rate determining how the importance of the utility changes with time, and $V^t_{\text{succ},i}$ is the number of served IoT devices of the ith CE group in the tth TTI. $n^t_{\text{Rach},i}$, $n^t_{\text{Repe},i}$, and $f^t_{\text{Prea},i}$ are the optimized configurations, namely the number of RACH periods, the repetition value, the number of preambles in each RACH period for the ith CE group in the tth TTI, respectively.

B. Preliminary

The optimized number of served IoT devices over the long term given in Eq. (13) is really complicated, which cannot be easily solved via the conventional uplink resource approach. Therefore, most previous works simplified the objective to dynamically optimize the single parameter to achieve the maximum number of served IoT devices in the single group without consideration of future performance [8, 9], which is expressed as

$$(\text{P2}): \max_{\pi(x \,|\, S^t)} \quad \mathbb{E}_\pi[V^t_{\text{succ},0}], \tag{14}$$

where x is the optimized single parameter.

To maximize number of served IoT devices in the tth TTI, the configuration x is expected to be dynamically adjusted according to the actual number of IoT devices that will execute RACH attempts D^t_{RACH}, which refers to the current load of the network. Note that in practice, this load information is unable to be detected at the eNB. Thus, it is necessary to estimate the load based on the previous transmission reception from the 1th to $(t-1)$th TTI O^t before the uplink resource configuration in the tth TTI.

In [9], the authors designed a dynamic ACB scheme to optimize the problem given in Eq. (13) via adjusting the ACB factor. This scheme is based on a model-free load estimation approach, where the estimated number of RACH attempting IoT devices in the tth TTI $\hat{D}^t_{\text{RACH}}$ is expressed as:

$$\hat{D}^t_{\text{RACH}} = \max\left\{0, \hat{D}^{t-1}_{\text{RACH}} + \max\left\{-f^{t-1}_{\text{Prea},0}, \hat{D}^t_{\text{RACH}} - \hat{D}^{t-1}_{\text{RACH}}\right\}\right\} \tag{15}$$

where $f^{t-1}_{\text{Prea},0}$ is the number of allocated preambles in the $(t-1)$th TTI, and $\hat{D}^{t-1}_{\text{RACH}}$ is the estimated number of devices performing RACH attempts in the $(t-1)$th TTI given as

$$\hat{D}^{t-1}_{\text{RACH}} = f^{t-1}_{\text{Prea},0} / \Big[\min\Big\{1, p^{t-1}_{\text{ACB}}\big(1 + \frac{(V^{t-1}_{\text{coll}} - u_{M,p^*})e}{2f^{t-1}_{\text{Prea},0}}\big)^{-1}\Big\}\Big]. \tag{16}$$

In Eq. (16), p^{t-1}_{ACB}, $f^{t-1}_{\text{Prea},0}$, and V^{t-1}_{coll} are the ACB factor, the number of preambles and the observed number of collided preambles in the $(t-1)$th TTI, and u_{M,p^*} is an estimated factor given in [9, Eq. (32)].

In Eq. (15), $\hat{D}^t_{\text{RACH}} - \hat{D}^{t-1}_{\text{RACH}} \approx \hat{D}^{t-1}_{\text{RACH}} - \hat{D}^{t-2}_{\text{RACH}}$ is the difference between the estimated numbers of RACH requesting IoT devices in the $(t-1)$th and the tth TTIs, which is obtained by assuming that the number of successful RACH IoT devices does not change significantly in these two TTIs [9].

This approach can accurately estimate the load in bursty traffic scenarios when the exact load is bigger than the number of preambles (i.e., $D^t_{\text{RACH}} > f^t_{\text{Prea},0}$), due to that it is specifically designed for a dynamic ACB scheme (i.e., the ACB scheme is only triggered when $D^t_{\text{RACH}} > N^t_{\text{p}}$). However, it cannot estimate the load when $D^t_{\text{RACH}} < f^t_{\text{Prea},0}$, which is required in our problem.

C. Resource Configuration in Single Parameter Single CE Group Scenario

In this subsection, we modify the load estimation approach given in [9] via estimating based on the last number of the collided preambles V^{t-1}_{coll} and the previous numbers of idle preambles $V^{t-1}_{\text{idle}}, V^{t-2}_{\text{idle}}, \cdots$. And then, we propose an uplink resource configuration approach based on this revised load estimation, namely, LE-URC.

1) Load Estimation: By definition, $\boldsymbol{\mathcal{F}}_{\text{Prea}}$ is the set of valid number of preambles that the eNB can choose, where each IoT device selects a RACH preamble from $f^t_{\text{Prea},0}$ available preambles with an equal probability given by $1/f^t_{\text{Prea},0}$. For a given preamble j transmitted to the eNB, let M_j denotes the number of IoT devices that selects the preamble j. The probability that no IoT device selects preamble j is

$$\mathbb{P}\{M_j = 0 \big| D^{t-1}_{\text{RACH},0} = n\} = \big(1 - \frac{1}{f^{t-1}_{\text{Prea},0}}\big)^n. \tag{17}$$

The expected number of preambles experiencing idles $\mathbb{E}\{\mathcal{V}^{t-1}_{\text{idle},0} \big| D^{t-1}_{\text{RACH},0} = n\}$ in the $(t-1)$th TTI is given by

$$\mathbb{E}\{\mathcal{V}^{t-1}_{\text{idle},0} \big| D^{t-1}_{\text{RACH},0} = n\} = \Big(\sum_{j>1}^{f^{t-1}_{\text{Prea},0}} \mathbb{P}\{M_j = 0 \big| D^{t-1}_{\text{RACH}} = n\}\Big) = f^{t-1}_{\text{Prea},0}\big(1 - \frac{1}{f^{t-1}_{\text{Prea},0}}\big)^n. \tag{18}$$

Due to that the actual number of preambles experiencing idles $V_{\text{idle},0}^{t-1}$ can be observed at the eNB, the number of RACH attempting IoT devices in the $(t-1)$th TTI ζ^{t-1} can be estimated as

$$\zeta^{t-1} = f^{-1}(\mathbb{E}\{V_{\text{idle},0}^{t-1}\big|D_{\text{RACH},0}^{t-1}) = \log_{(\frac{f_{\text{Prea},0}^{t-1}-1}{f_{\text{Prea},0}^{t-1}})}(\frac{V_{\text{idle},0}^{t-1}}{f_{\text{Prea},0}^{t-1}}), \tag{19}$$

To obtain the estimated number of RACH attempting IoT devices in the tth TTI $\tilde{D}_{\text{RACH},0}^{t}$, we also need to know the difference between the estimated numbers of RACH attempting IoT devices in the $(t-1)$th and the tth TTIs, denoted by δ^t, where $\delta^t = \tilde{D}_{\text{RACH},0}^{t} - \tilde{D}_{\text{RACH},0}^{t-1}$ for $t = 1, 2, \cdots$, and $\tilde{D}_{\text{RACH},0}^{0} = 0$. However, $\tilde{D}_{\text{RACH},0}^{t}$ cannot be obtained before the tth TTI. To solve this, we can assume $\delta^t \approx \delta^{t-1}$ according to [9]. This is due to that the time between two consecutive TTIs is small, and the available preambles are gradually updated leading to that the number of successful RACH IoT devices does not change significantly in these two TTIs. Therefore, the number of RACH attempting IoT devices in the tth time slot is estimated as

$$\tilde{D}_{\text{RACH},0}^{t} = \max\{2V_{\text{coll},0}^{t-1}, \zeta^{t-1} + \delta^{t-1}\}, \tag{20}$$

where $2V_{\text{coll}}^{t-1}$ represents that there are at least $2V_{\text{coll},0}^{t-1}$ number of IoT devices colliding in the last TTI.

2) Uplink Resource Configuration Based on Load Estimation: In the following, we propose a conventional resource configuration approach based on load estimation (LE-URC) by taking into account the resource condition given in Eq. (11). The number of RACH periods $n_{\text{Rach},0}$ and the repetition value $n_{\text{Repe},0}$ is fixed, and only the number of preambles in each RACH period $f_{\text{Prea},0}$ is dynamically configured in each TTI. Using the estimated number of RACH attempting IoT devices in the tth TTI $\tilde{D}_{\text{RACH},0}^{t}$, the probability that only one IoT device selects preamble j (i.e., no collision occurs) is expressed as

$$\mathbb{P}\{M_j = 1\big|\tilde{D}_{\text{RACH},0}^{t} = n\} = \binom{n}{1}\frac{1}{f_{\text{Prea},0}^{t}}\big(1 - \frac{1}{f_{\text{Prea},0}^{t}}\big)^{n-1}. \tag{21}$$

The expected number of RACH attempting IoT devices in the tth TTI is derived as

$$\mathbb{E}\{\mathcal{V}_{\text{RACH},0}^{t}\big|\tilde{D}_{\text{RACH},0}^{t} = n\} = \sum_{j>1}^{f_{\text{Prea},0}^{t}} \mathbb{P}\{M_j = 1\big|\tilde{D}_{\text{RACH},0}^{t} = n\} = n\big(1 - \frac{1}{f_{\text{Prea},0}^{t}}\big)^{n-1}, \tag{22}$$

Based on (22), the expected number of IoT devices requesting uplink resource in the tth TTI is derived as

$$\mathbb{E}\{\mathcal{V}_{\text{reqs}}^{t}\big|\tilde{D}_{\text{RACH},0}^{t} = n\} = \mathbb{E}\{\mathcal{V}_{\text{RACH},0}^{t}\big|\tilde{D}_{\text{RACH},0}^{t} = n\} + V_{\text{unsc},0}^{t} = n\big(1 - \frac{1}{f_{\text{Prea},0}^{t}}\big)^{n-1} + V_{\text{unsc},0}^{t-1}, \tag{23}$$

where V_{unsc}^{t-1} is the number of unscheduled IoT devices in the last TTI. Note that V_{unsc}^{t-1} can be observed.

However, if the data resource is not sufficient (i.e., occurs when Eq. (11) is invalid), some IoT devices may not be scheduled in the tth TTI. The upper bound of the number of scheduled IoT devices $\mathcal{V}_{\text{up},0}^{t}$ is expressed as

$$\mathcal{V}_{\text{up},0}^{t} = \frac{R_{\text{DATA}}^{t}}{r_{\text{DATA},i}^{t}} = \frac{R_{\text{Uplink}} - R_{\text{RACH}}^{t}}{r_{\text{DATA},i}^{t}}. \tag{24}$$

where R_{uplink} is the total number of REs reserved for uplink transmission in a TTI, R^t_{RACH} is the uplink resource configured for RACH in the tth TTI given in Eq. (10). $r^t_{\text{DATA},0}$ is required REs for serving one IoT device given in Eq. (12).

According to (23) and (24), the expected number of the successfully served IoT devices is given by

$$\mathcal{V}^t_{suss}(f^t_{\text{Prea},0}) = \textbf{min} \quad \{\mathbb{E}\{\mathcal{V}^t_{\text{reqs}} \big| \tilde{D}^t_{\text{RACH},0} = n\}, \mathcal{V}^t_{up}\}. \tag{25}$$

Then, we calculate the expected number of the successfully served IoT devices for each $n^t_{\text{Prea},0} \in \boldsymbol{\mathcal{N}}_{\text{Prea}}$, and the eNB selects the number of preamble $n^{t*}_{\text{Prea},0}$ in the tth TTI, which obtains the maximal expected number of the successfully served IoT devices.

The LE-URC approach based on the load estimation $\tilde{D}^t_{\text{RACH},0}$ is detailed in **Algorithm 1**. For comparison, we consider an ideal scenario that the actual number of RACH requesting IoT devices D^t_{RACH} is available at the eNB, namely, full state information based URC (FSI-URC). FSI-URC configures $f^t_{\text{Prea},0}$ still using the approach given in Eq. (20), while the load estimation approach given in Section III.C.1) is not required.

Algorithm 1: Load Estimation Based Uplink Resource Configuration (LE-URC)

input : The set of the number of preambles in each RACH period $\boldsymbol{\mathcal{N}}_{\text{Prea},0}$, Number of IoT devices D, Operation Iteration I.

for *Iteration* $\leftarrow 1$ **to** I **do**
  Initialization of $V^0_{\text{idle},0} := 12$, $V^0_{\text{coll},0} := 0$, $\tilde{D}^0_{\text{RACH},0} := 0$, $\delta^1 := 0$, and bursty traffic arrival rate $\mu^0_{bursty} = 0$;
  for $t \leftarrow 1$ **to** T **do**
    Generate μ^t_{bursty} using Eq. (3);
    The eNB observes $V^{t-1}_{\text{idle},0}$ and $V^{t-1}_{\text{coll},0}$, and calculate ζ^{t-1} using Eq. (19);
    Estimate the number of RACH requesting IoT devices $\tilde{D}^t_{\text{RACH},0}$ using Eq. (20);
    Select the number of preambles $f^{t*}_{\text{Prea},0}$ using Eq. (25) based on the estimated load $\tilde{D}^t_{\text{RACH},0}$;
    The eNB broadcasts $f^{t*}_{\text{Prea},0}$, and backlogged IoT devices attempt communication in the tth TTI;
    Update $\delta^{t+1} := \tilde{D}^t_{\text{RACH},0} - \tilde{D}^{t-1}_{\text{RACH},0}$.
  end
end

IV. Single Parameter Single CE Group Scenario

The RL approaches are well-known in addressing dynamic control problem in complex POMDPs [31]. Nevertheless, they have been rarely studied in handling the resource configuration in slotted-Aloha based wireless communication systems. Therefore, it is worthwhile to evaluate the capability of RL in the single parameter single CE group scenario first, in order to be compared with conventional resource configuration approaches. In this section, we consider the single CE group scenario with the fixed number of RACH periods $n_{\text{Rach},0}$ and the repetition value $n_{\text{Repe},0}$, and only dynamically configuring the number of preambles $f_{\text{Prea},0}$ in each RACH period. In the following, we develop Q-learning based real-time optimization approaches to deal with the uplink resource configuration.

A. Q-Learning and Tabular Value Function

In the following, we propose the real-time optimization to solve uplink resource configuration in the single CE group scenario using Q-learning, which is an efficient value-based RL technique with guaranteed convergence [31]. We first study Q-learning based on the tabular representation of the value function (tabular-Q), which is the simplest form that the exact optimal policy can always be found, but requires extremely long training time. We then study Q-learning based on the value function approximation, where the linear approximation (LA) and deep neural network (DNN) will be used to construct an approximation of the desired value function.

Considering a Q-agent deployed at the eNB to optimize the number of successfully served IoT devices in real-time, the Q-agent need to explore the environment in order to choose appropriate actions progressively leading to the optimization goal. We define $s \in \mathcal{S}$, $a \in \mathcal{A}$, and $r \in \mathcal{R}$ as any state, action, and reward from their corresponding sets, respectively. At the beginning of the tth TTI ($t \in \{0, 1, 2, \cdots\}$), the Q-agent first observes the current state S^t corresponding to a set of previous observations $(O^{t-1}, O^{t-2}, \cdots, O^0)$ in order to select an specific action $A^t \in \mathcal{A}(S^t)$. The action A^t corresponds to the number of preambles in each RACH period $f^t_{\text{Prea},0}$ in single CE group scenario.

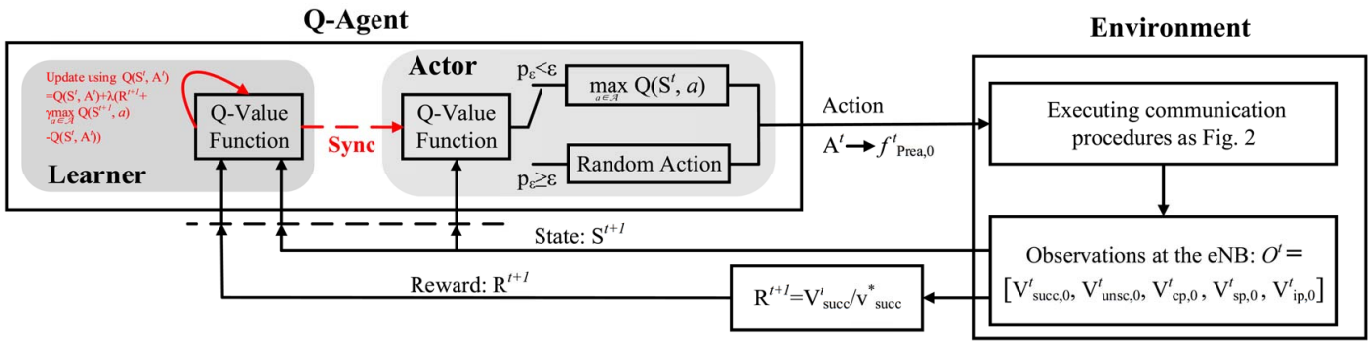

Fig. 3: The Tabular-Q agent and environment interaction in the MDP.

As shown in Fig. 3, we consider a basic state function in the single CE group scenario, where S^t is a set of indices mapping to the current observed information $O^{t-1} = [V^{t-1}_{\text{succ}}, V^{t-1}_{\text{unsc}}, V^{t-1}_{\text{cp}}, V^{t-1}_{\text{sp}}, V^{t-1}_{\text{ip}}]$. With the knowledge of the state S^t, the Q-agent chooses an action A^t from the set $\mathcal{A}$, which is a set of indexes mapped to the set of the number of available preambles $\boldsymbol{\mathcal{N}}_{\text{Prea}}$. Once an action A^t is performed, the Q-agent will receive a scalar reward R^{t+1}, and observe a new state S^{t+1}. The reward R^{t+1} indicates to what extent the executed action A^t can achieve the optimization goal, which is determined by the new observed state S^{t+1}. As the optimization goal is to maximize the number of the successfully served IoT devices, we define

the reward R^{t+1} as a function that positively proportional to the observed number of successfully served IoT devices $V_{\text{succ}}^t \in O^t$, which is defined as

$$R^{t+1} = V_{\text{succ}}^t / \mathrm{V}_{\text{succ}}^*, \tag{26}$$

where $\mathrm{V}_{\text{succ}}^*$ is constant used to normalize the reward function referring to the maximum number of successfully served IoT devices.

Q-learning is a value-based RL approach [17, 31], where the policy of states to actions mapping $\pi(s) = a$ is learned using a state-action value function $Q(s, a)$ to determine an action for the state s. We first use a *lookup table* to represent the state-action value function $Q(s, a)$ (tabular-Q), which consists of value scalars for all the state and action spaces. To obtain an action A^t, we need to select the highest value scalar from the numerical value vector $Q(S^t, a)$, which maps all possible actions under S^t to the Q-value table $Q(s, a)$.

Accordingly, our objective is to derive an optimal Q-value table $Q^*(s, a)$ with optimal policy π^* that can select actions to dynamically optimize the number of served IoT devices. To do so, we train a initial Q-value table $Q(s, a)$ in the environment using Q-Learning algorithm, where $Q(s, a)$ is immediately updated using the current observed reward $\mathcal{R}^{t+1}$ after each action as

$$Q(S^t, A^t) = Q(S^t, A^t) + \lambda\big[R^{t+1} + \gamma \max_{a \in \mathcal{A}} Q(S^{t+1}, a) - Q(S^t, A^t)\big], \tag{27}$$

where λ is a constant step-size learning rate that affects how fast the algorithm adapt to a new environment, $\gamma \in [0, 1)$ is the discount rate that determines how current rewards affects the value function updating, $\max_{a \in \mathcal{A}} Q(S^{t+1}, a)$ approximates the value in optimal Q-value table $Q^*(s, a)$ via the up-to-date Q-value table $Q(s, a)$ and the obtained new state S^{t+1}. Note that $\max_a Q(S^{t+1}, a)$ is obtained with the policy of choosing the optimal value scalar, which is independent of the action actually chosen for S^{t+1} in the $(t+1)$th TTI (i.e., in the $(t+1)$th TTI, the IoT device may randomly choose an action for exploitation, rather than the action corresponding to $\max_{a \in \mathcal{A}} Q(S^{t+1}, a)$). Note that $Q(S^t, A^t)$ in Eq. (27) is a scalar, which means that we can only update one value scalar in the Q-value table $Q(s, a)$ with one received reward R^{t+1}. Therefore, each element of Q-value table $Q(s, a)$ is exactly updated using the combination of state S^t and action A^t, and thus it is expected to find the exact optimal value function and optimal policy.

As shown in Fig. 3, we consider ϵ-greedy approach to balance *exploitation* and *exploration* in the Actor of the Q-Agent, where ϵ is a positive real number and $\epsilon \leq 1$. In each TTI, the Q-agent randomly generates a probability p_ϵ to compare with ϵ. Then, with the probability ϵ, the algorithm randomly chooses an action from the remaining feasible actions to improve the estimate of the non-greedy action's value. With the probability $1 - \epsilon$, the algorithm exploits the current knowledge of the Q-value table to choose the action that maximizes the expected reward.

Algorithm 2: Tabular-Q Based Uplink Resource Configuration

input : Valid numbers of preambles $\mathcal{N}_{\text{Prea}}$, Number of IoT devices D, Operation Iteration I.

Algorithm hyperparameters: learning rate $\lambda \in (0, 1]$, discount rate $\gamma \in [0, 1)$, ϵ-greedy rate $\epsilon \in (0, 1]$;
Initialization of the Q-value table $Q(s, a)$ with 0 value scalars;
for *Iteration* $\leftarrow 1$ **to** I **do**
    Initialization of S^1 by executing a random action A^0 and bursty traffic arrival rate $\mu^0_{bursty} = 0$;
    for $t \leftarrow 1$ **to** T **do**
        Update μ^t_{bursty} using Eq. (3);
        if $p_\epsilon < \epsilon$ **then** select a random action A^t from $\mathcal{A}$;
        else select $A^t = \underset{a \in \mathcal{A}}{\text{argmax}}\, Q(S^t, a)$;
        The eNB broadcasts $\mathcal{N}_{\text{Prea}}(A^t)$ and backlogged IoT devices attempt communication in the tth TTI;
        The eNB observes S^{t+1}, calculate the related R^{t+1} using Eq. (26), and update $Q(S^t, A^t)$ using Eq. (27).
    end
end

Particularly, the learning rate λ is suggested to be set to a small number (e.g., $\lambda = 0.01$) to guarantee the stable convergence of Q-value table in this NB-IoT communication system. This is due to a single reward in a specific TTI can be severely biased, because the reward function is defined based on the observed state function, which is composed of multiple unpredictable distributions (e.g., an action allows for the setting with large number of preambles N_{preamble}, but massive random collisions accidentally occur, which leads to an unusual low reward). In the following, the implementation of uplink resource configuration using Q-learning based real-time optimization is shown in **Algorithm 2**.

B. Value Function Approximation

Since tabular-Q needs its each element to be updated to converge, searching for an optimal policy can be difficult in limited time and computational resource. To solve this problem, we use a value function approximator instead of Q-value table to find a sub-optimal approximated policy. Generally, selecting a efficient approximation approach to represent the value function for different learning scenarios is a usual problem within the RL [31–34]. A variety of function approximation approaches can be conducted, such as linear approximation, deep neural networks (DNNs), tree search, and which approach to select can critically influence the successful learning [31, 33, 34]. The function approximation should fit the complexity of the desired value function, and be efficient to obtain good solutions. Unfortunately, most function approximation approaches require specific design for different learning problems, and there is no basis function, which is both reliable and efficient to satisfy all learning problems.

In this subsection, we first focus on the linear function approximation for Q-learning approach (LA-Q), due to its simplicity, efficiency, and convergence [31, 35, 36]. We then conduct the DNN for Q-learning (DNN-Q) as a more effective but complicated function approximator, which is also known as deep Q-network (DQN) [17]. The reasons we propose DNN-Q are that: 1) the DNN function approximation is able to deal with several kinds of partially observable problems [17, 31]; 2) DNN-Q has the potential to accurately approximate the desired value function while addressing a problem with very large state and action spaces [17], which is favored for the learning in the multiple CE group scenarios; 3) DNN-Q is with high scalability, where the scale of its value function can be easily fit to a more complicated problem; 4) a variety of libraries have been established to facilitate building DNN architectures and accelerate experiments, such as TensorFlow, Pytorch, Theano, Keras, and etc..

1) Linear Approximation: LA-Q uses a linear weight matrix $\mathbf{w}$ to approximate the value function $Q(s,a)$ with feature vector $\vec{x} = \mathbf{x}(s)$ corresponding to the state S^t. The dimensions of weight matrix $\mathbf{w}$ is $|\mathcal{A}| \times |\vec{x}|$, where $|\mathcal{A}|$ is the total number of all available actions and $|\vec{x}|$ is the size of feature vector $\vec{x}$. Here, we consider polynomial regression (as [31, Eq. 9.17]) to construct the real-valued feature vector $\mathbf{x}(s)$ due to its efficiency[5].In the training process, the exploration is the same as the tabular Q-learning by generating random actions, but the exploitation is not based on table lookup, but to calculate the value function scalars using the weight matrix $\mathbf{w}$. In detail, to predict an action using the LA value function $Q(S^t, a, \mathbf{w})$ with state S^t in the tth TTI, the approximated value function scalars for each action $\mathbf{a}$ is obtained by inner-producting between the weight matrix $\mathbf{w}$ and the features vector $\mathbf{x}(s)$ as:

$$Q(S^t, a, \mathbf{w}) = \mathbf{w} \cdot \mathbf{x}(S^t)^T = \Big[\sum_{j=0}^{|\vec{x}|-1} w_{(0,j)} x_j(S^t), \sum_{j=0}^{|\vec{x}|-1} w_{(1,j)} x_j(S^t), \cdots, \sum_{j=0}^{|\vec{x}|-1} w_{(|\mathcal{A}|-1,j)} x_j(S^t) \Big]^T. \tag{28}$$

By searching for the maximal value function scalar in $Q(S^t, a, \mathbf{w})$ given in Eq. (28), we can obtain the matched action A^t to maximize future rewards.

To obtain the optimal policy, we update the weigh matrix $\mathbf{w}$ in the value function $Q(s,a;\mathbf{w})$ using stochastic gradient descent (SGD) [31, 38]. SGD minimizes the error on predictions of observed samples after each example, where the error is reduced by a small amount following the direction to the optimal target policy $Q^*(s,a)$. In practice, it is infeasible to obtain optimal target policy by summing over all states, due to that the LA value function requires to be updated in each TTI. Instead, we estimate the desired

[5]The polynomial case is the most well understood feature constructor and always performs well in practice with appropriate setting [31, 32]. Furthermore, the results in [37] shows that there is a rough correspondence between a fitted neural network and a fitted ordinary parametric polynomial regression model. These reasons encourage us to compare the polynomial based LA-Q with DNN-Q

action-value function by simply considering one learning sample $Q^*(s,a) \approx Q^*(S^t, a, \mathbf{w}^t)$ [31]. In each TTI, the weigh matrix $\mathbf{w}$ is updated following

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha \nabla L(\mathbf{w}^t), \tag{29}$$

where λ is the learning rate. Considering the least squares estimator, the loss function $\nabla L(\mathbf{w}^t)$ is defined as

$$\nabla L(\mathbf{w}^t) = \left(R^{t+1} + \gamma \max_a Q(S^{t+1}, a; \mathbf{w}^t) - \mathbf{w}^t \cdot \mathbf{x}(A^t, S^t)^T\right) \cdot \mathbf{w}^t \tag{30}$$

where $\mathbf{w}^t$ is the weight matrix, $\mathbf{x}(A^t, S^t)$ is the features matrix with the same shape of $\mathbf{w}^t$ (i.e., $|\mathbf{a}| \times d_{\mathrm{f}}$). $\mathbf{x}(A^t, S^t)$ is constructed by zeros and the feature vector located in the row corresponding to the index of the action selected in tth TTI A_t. Note that $Q^*(S^{t+1}, a; \mathbf{w}^t)$ is a scalar. The learning procedure follows **Algorithm 2** by changing the Q-table $Q(s,a)$ to the LA value function $Q(s,a;\mathbf{w})$ with linear weigh vector $\mathbf{w}$, and updating $Q(s,a;\mathbf{w})$ with SGD given in (30) in step 10 of **Algorithm 2**.

2) Deep Q-Network: DNN is implemented by multiple hidden layers and one output layer, where each layer is composed of different number of neuron units. We consider the regular architecture of neural networks, where neurons between two adjacent layers are fully pairwise connected, namely fully-connected layers. The output layer is composed of linear units[6], which are in one-to-one correspondence with all available actions in $\mathbf{a}$. All hidden layers are equipped with rectifier linear units (ReLUs), which computes the function $f(x) = \max(0, x)$.

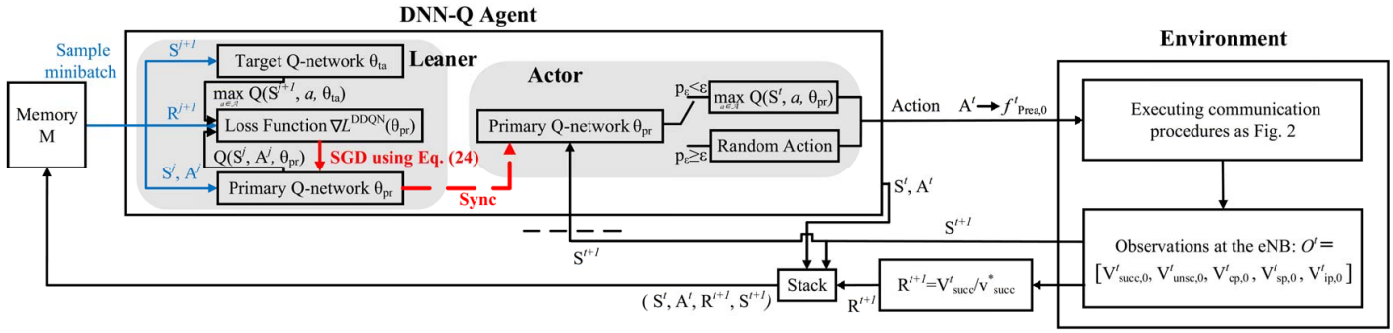

Fig. 4: The DNN-Q agent and environment interaction in the MDP.

Considering DNN-Q is with the value function $Q(s,a;\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ refers to weights for the whole Q-network. The value function $Q(s,a;\boldsymbol{\theta})$ will also be trained using SGD same as the LA approach. To improve the convergent reliability, we use minibatch, instead of a single sample to update the value function

[6]Linear activation is used here according to [17]. Note that, in this learning problem, the desired value function given in Eq. (15) can be bigger than 1, rather than a probability, and thus the activation function with return value limited in $[-1, 1]$ (such as sigmoid function and tanh function) can lead to convergence difficulty.

$Q(s,a;\boldsymbol{\theta})$ [17], where a number of experience are randomly sampled to approximate the desired value function $Q^*(s,a) \approx Q^*(s,a,\boldsymbol{\theta}^-)$. Furthermore, to avoid overestimation[7], we use double deep Q-networks (DDQNs) algorithm [18]. As shown in Fig. 4, the primary Q-network $Q(s,a;\boldsymbol{\theta}_{\text{pr}})$ is used to choose the action and is updated every training step, and the target Q-network $Q(s,a;\boldsymbol{\theta}_{\text{ta}})$ is used to generate the maximal value for a state when computing the target-Q value $Q^*(s,a,\boldsymbol{\theta})$ for each training step. In the training process, the target Q-network $Q(s,a;\boldsymbol{\theta}_{\text{ta}})$ remains unchanged for a dedicated duration, and then be periodically updated by the primary Q-network $Q(s,a;\boldsymbol{\theta}_{\text{pr}})$. Considering the least squares estimator, the weights of primary Q-network is updated following

$$\boldsymbol{\theta}_{pr}^{t+1} = \boldsymbol{\theta}_{\text{pr}}^{t} - \alpha \nabla L(\boldsymbol{\theta}_{\text{pr}}^{t}), \tag{31}$$

where the loss function $\nabla L^{\text{DDQN}}(\boldsymbol{\theta}_{\text{pr}}^{t})$ is defined as

$$\nabla L^{\text{DDQN}}(\boldsymbol{\theta}_{\text{pr}}^{t}) = \mathbb{E}_{S^j,A^j,R^{j+1},S^{j+1}}\Big[\big(R^{j+1} + \gamma \max_{a} Q(S^{j+1},a;\boldsymbol{\theta}_{\text{ta}}^{-}) - Q(S^j,A^j;\boldsymbol{\theta}_{\text{pr}}^{-})\big)\nabla_{\boldsymbol{\theta}_{\text{pr}}} Q(s,a;\boldsymbol{\theta}_{\text{pr}})\Big]. \tag{32}$$

Note that the error backpropagation is used to calculate the gradient of the weights in the DNN-Q [38], and RMSProp is used to adjust learning rate λ [39]. The algorithm of uplink resource configuration using real-time optimization based on DNN-Q is presented in **Algorithm 3**.

V. Multi-Parameters Multiple CE Groups Scenario

In this section, we study the uplink resource configuration with N_{CE} number of CE groups, where each CE group shares the uplink resource in the same bandwidth. At the beginning of a TTI, the eNB separately configure RACH resource to each CE group, and then it schedules data resource to all RRC connected IoT devices without the CE group bias. To optimize the number of served IoT devices in real-time, the eNB should not only balance the uplink resource between RACH and data, but also balance them among each CE group.

A. Conventional Uplink Resource Configuration Based on the Load Estimation

We slightly revise the introduced single parameter single CE group LE-URC approach (given in Section III.C) to dynamically configure resource for multiple CE groups. Note that the repetition value $n_{\text{Repe},i}$ in the LE-URC approach is still defined as a constant to enable the availability of load estimation in Eq. (20). Remind that the principle of LE-URC approach is to optimize the expectation of the number of successful served IoT devices while balancing R_{RACH}^{t} and R_{DATA}^{t} with limited uplink resource $R_{\text{uplink}} = R_{\text{DATA}}^{t} + R_{\text{RACH}}^{t}$.

[7]Overestimation refers to that some suboptimal actions regularly were given higher Q-values than optimal actions, which can negatively influence the convergence capability and training effieciency of the algorithm [18, 33].

Algorithm 3: DNN-Q Based Uplink Resource Configuration

input : The set of numbers of preambles in each RACH period $\boldsymbol{\mathcal{N}}_{\text{Prea}}$, the number of IoT devices D, and operation iteration I.

Algorithm hyperparameters: learning rate $\lambda \in (0, 1]$, discount rate $\gamma \in [0, 1)$, ϵ-greedy rate $\epsilon \in (0, 1]$, target network update frequency K;
Initialization of replay memory M to capacity C, the primary Q-network $Q(s, a; \boldsymbol{\theta}_{\text{pr}})$, and the target Q-network $Q(s, a; \boldsymbol{\theta}_{\text{ta}})$;
for *Iteration* $\leftarrow 1$ **to** I **do**
    Initialization of S^1 by executing a random action A^0 and bursty traffic arrival rate $\mu^0_{bursty} = 0$;
    for $t \leftarrow 1$ **to** T **do**
        Update μ^t_{bursty} using Eq. (3);
        if $p_\epsilon < \epsilon$ **then** select a random action A^t from $\mathcal{A}$;
        else select $A^t = \underset{a \in \mathcal{A}}{\text{argmax}}\, Q(S^t, a, \boldsymbol{\theta}_{\text{pr}})$;
        The eNB broadcasts $\boldsymbol{\mathcal{N}}_{\text{Prea}}(A^t)$ and backlogged IoT devices attempt communication in the tth TTI;
        The eNB observes S^{t+1}, and calculate the related R^{t+1} using Eq. (26);
        Store transition $(S^t, A^t, R^{t+1}, S^{t+1})$ in M;
        Sample random minibatch of transitions $(S^j, A^j, R^{j+1}, S^{j+1})$ from M;
        Perform a gradient descent for $Q(s, a; \boldsymbol{\theta}_{\text{pr}})$ using Eq. (32);
        Every K steps update target Q-network $Q(s, a; \boldsymbol{\theta}_{\text{ta}}) = Q(s, a; \boldsymbol{\theta}_{\text{pr}})$.
    end
end

In the multiple CE groups scenarios, the resource R^t_{DATA} are allocated to IoT devices in any CE groups without bias, but R^t_{RACH} is specifically allocated to each CE group.

Under this condition, the expected number of successfully served IoT devices $\mathcal{V}^t_{\text{suss},i}$ given in Eq. (25) needs to be modified by taking into account N_{CE} number of variables, which becomes non-convex, and extremely complicates the optimization problem. To solve it, we use a sub-optimal solution by artificially setting uplink resource constrain $R_{\text{Uplink},i}$ for each CE group ($R_{\text{Uplink}} = \sum_{i=0}^{N_{\text{CE}}} R_{\text{Uplink},i}$). Each CE group can independently allocate the resource between $R^t_{\text{DATA},i}$ and $R^t_{\text{RACH},i}$ according to the approach given in Eq. (25).

B. Reinforcement Learning Approach

The Q-learning algorithms with the single CE group provided in Section IV.B and IV.C are model-free, and thus their learning structure can be directly used in the multiple configurations multiple CE groups scenario. However, the space of observations increases with increasing the number of configured CE groups n_{CE}, this exponentially increases the size of state spaces. To train Q-agent with this expanded state spaces, the requirements of time and computational resource greatly increase. In such case, the Q-learning with tabular approach would be extremely inefficient, as not only the state-action value table requires a big memory, but also impossible to experience every state to achieve convergence with limited time. In view of this, we only

consider Q-learning with value function approximation (i.e., LA-Q and DNN-Q) to design uplink resource configuration approaches for the multiple CE groups scenarios.

LA-Q and DNN-Q are with high capability to handle massive state spaces, and thus we can considerably improve the state spaces with more observed information to support the optimization of Q-agent. Here, we define the current state S^t stores the observations in the last $M_{\rm o}$ TTIs ($O^{t-1}, O^{t-2}, O^{t-3}, \cdots, O^{t-M_{\rm o}}$). This design improves Q-agent by enabling it to estimate the trend of observations. As our goal is to optimize the number of served IoT devices, the reward function should be defined according to the number of successfully served IoT devices $V_{{\rm succ},i}$ of each CE group. To satisfy different fairness criterion for each group, we introduce $\alpha - fairness$[8] to balance the reward function received from different CE groups, which is expressed as

$$R^{t+1} = \sum_{i=1}^{N_{\rm CE}} w_{{\rm r},i} \frac{\left(V_{{\rm succ},i}^{t}\right)^{1-\alpha}}{1-\alpha}, \tag{33}$$

where $w_{{\rm r},i}$ is the reward weight for each CE group, and α determines the utility of fairness criteria.

Same as the state spaces, the available action spaces also exponentially increases with the increase of the adjustable configurations. The number of available actions corresponds to the number of available configurations

$$|\mathcal{A}| = \prod^{N_{\rm CE}} |\boldsymbol{\mathcal{N}}_{\rm Rach}| \times |\boldsymbol{\mathcal{N}}_{\rm Repe}| \times |\boldsymbol{\mathcal{F}}_{\rm Prea}|, \tag{34}$$

where $|\cdot|$ denotes the number of elements in vector $\cdot$, and $\mathcal{A}$ is the set of actions. In Eq. (34), $\boldsymbol{\mathcal{N}}_{{\rm Rach},i}$, $\boldsymbol{\mathcal{N}}_{{\rm Repe},i}$, and $\boldsymbol{\mathcal{F}}_{{\rm Prea},i}$ are the sets of the number of RACH periods, the repetition value, and the number of preambles in each RACH period, respectively.

Unfortunately, it is extremely hard to optimize the system under such numerous action spaces (i.e., $|\mathcal{A}|$ can be over ten thousands for the 3 CE groups.), due to that the system will fall into updating policy with only a small part of the action in $\mathcal{A}$, and finally leads to convergence difficulty. In the following, we provide two approaches to dynamically optimize uplink resource configuration in multiple configuration multiple CE groups scenario.

1) Actions Simplified Approach: We first provide the actions simplified approach (ASA), which guarantees convergent capability by sacrificing the accuracy of action selection, namely, action aggregation[9]. In detail, the specific action selection can be converted to the increasing or decreasing trend selection. Instead of

[8] $\alpha - fairness$ is a well-known criteria framework to balance the network resource [40]. Here, we leverage $\alpha - fairness$ to balance the reward function received from different CE group, which can indirectly balance the uplink resource for each CE group.

[9] The action aggregation has been rarely evaluated, but the same idea, namely, state aggregation has been well studied, which is a basic function approximation approach [31].

selecting the exact values from the sets of $\boldsymbol{\mathcal{N}}_{\text{Rach},i}$, $\boldsymbol{\mathcal{N}}_{\text{Repe},i}$, and $\boldsymbol{\mathcal{F}}_{\text{Prea},i}$, we convert it to single step ascent/descent based on the last action, which is represented by $A^t_{\text{Rach},i} \in \{0,1\}$, $A^t_{\text{Repe},i} \in \{0,1\}$, and $A^t_{\text{Prea},i} \in \{0,1\}$ corresponding to the number of RACH periods $n^t_{\text{Rach},i}$, the repetition values $n^t_{\text{Repe},i}$, and the number of preambles in each RACH period $f^t_{\text{Prea},i}$ in the tth TTI. Consequently, the size of total action spaces for N_{CE} CE groups is reduced to $|\mathcal{A}| = 2^{2N_{\text{CE}}}$. By doing so, the algorithms for training with LA function approximator and DQN in the multiple configurations multiple CE groups scenario can be deployed following **Algorithm 2** and **Algorithm 3**, respectively.

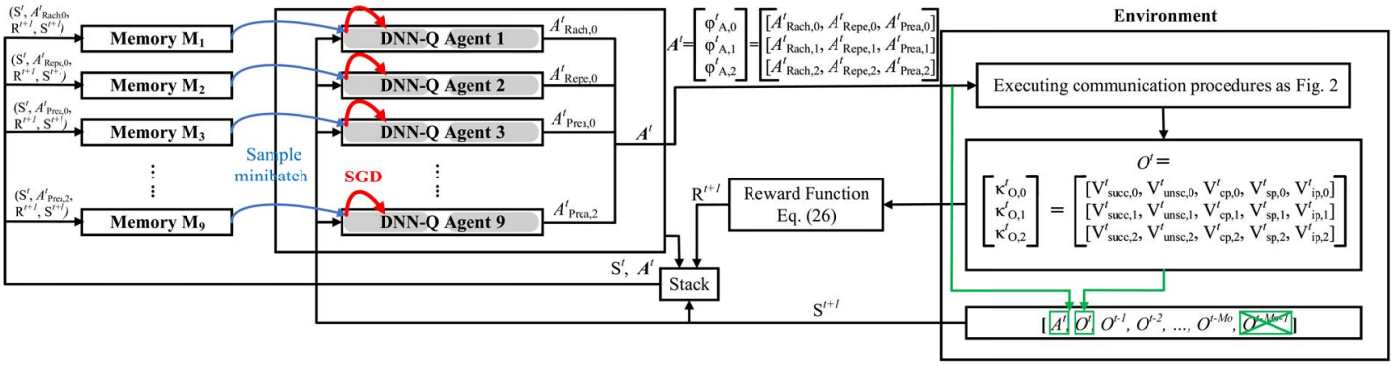

Fig. 5: The CMC-DNN-Q agents and environment interaction in the MDP.

2) Cooperative Learning Approach: Despite that the uplink resource configuration is managed by a central authority, identifying the control of each parameter as one sub-task that is cooperatively handled by independent Q-agents is promising to deal with the problem with unsolvable action spaces [41]. As shown in Fig. 5, we consider a centralized multi-agent cooperation deep Q-learning algorithm (CMC-DNN-Q), where multiple DNN-Q agents are implemented at the eNB with the same structure of value function approximator[10] following Section IV.C.2), and be cooperatively trained at the same times. We set the same goal to optimize the performance of the whole network, but be trained to handle different sub-tasks, which means each DNN-Q agent controls their own parameter as shown in Fig. 5.

In the training processes, each DNN-Q agent independently chooses its own action and explores the related actions space, which means that DNN-Q agents are parallel trained using the same input observations and different actions. Specifically, all DNN-Q agents share a common reward function, which is generated from the input signal receptions given by Eq. (33). This common reward function is a summary of signal receptions indicating the performance of the whole network, which makes all DNN-Q agents aiming at cooperatively

[10]The structures of value function approximator can also be specifically designed for RL agents with sub-tasks of significantly different complexity. However, there is no such requirement in our problem, so it will not be considered in this work.

increase the reward without conflicts of interests. Furthermore, the current state S^t is influenced by all the actions during the $(t-1)$th TTI, and thus each Q-agent has only limited capability to influence the common reward R^t.

The challenge of this approach is how to evaluate each action according to the common reward function. For each DNN-Q agent, the received reward is corrupted by massive noise, where its own effect on the reward is deeply hidden in the effects of all other DNN-Q agents. For instance, a positive action can receive a mismatched low reward due to other DNN-Q agents' negative actions. Fortunately, in our scenario, all DNN-Q agents are centralized at the eNB, which means that all DNN-Q agents can have full information among each other. To improve the observed information of each DNN-Q agent, we modify their state function that can contain the whole signal reception as before along with all selected actions[11] as seen in Fig. 5. For instance, considering the 3 CE groups scenario, the state in tth TTI is represented as

$$S^t = \left[[\phi_{A,0}^{t-1}, \phi_{A,1}^{t-1}, \phi_{A,2}^{t-1}, \kappa_{O,0}^{t-1}, \kappa_{O,1}^{t-1}, \kappa_{O,2}^{t-1}], \cdots, [\phi_{A,0}^{t-M_\mathrm{o}}, \phi_{A,1}^{t-M_\mathrm{o}}, \phi_{A,2}^{t-M_\mathrm{o}}, \kappa_{O,0}^{t-M_\mathrm{o}}, \kappa_{O,1}^{t-M_\mathrm{o}}, \kappa_{O,2}^{t-M_\mathrm{o}}]\right] \tag{35}$$

where M_o is the number of stored observations, the $\phi_{A,i}^{t-1} = [A_{\text{Rach},i}, A_{\text{Repe},i}, A_{\text{Prea},i}]$ is the set of selected actions of the ith CE group corresponding to the number of RACH periods $n_{\text{Rach},i}$, the repetition value $n_{\text{Repe},i}$, as well as the number of preambles in each RACH period $f_{\text{Prea},i}$, and $\kappa_{o,i}^{t-1} = [V_{\text{succ},i}^{t-1}, V_{\text{unsc},i}^{t-1}, V_{\text{cp},i}^{t-1}, V_{\text{sp},i}^{t-1}, V_{\text{ip},i}^{t-1}]$ is the set of observed signal receptions of the ith CE group in the $(t-1)$th TTI.

The learning algorithm can be implemented following **Algorithm 3**. Different from the single parameter single CE group scenario, we need to first initialize $2N_{\text{CE}}$ number of primary networks $Q(s, a_j; \boldsymbol{\theta}_{\text{pr},j})$, target networks $Q(s, a_j; \boldsymbol{\theta}_{\text{ta},j})$, and replay memories M_j. In step 11 of **Algorithm 3**, the current transactions of each Q-agent should be stored in their own memory separately. In step 12 and 13 of **Algorithm 3**, the minibatch of transaction should separately sampled from each memory to train the corresponding Q-agent.

VI. Simulation Results

In this section, we present the simulation results of proposed Q-learning based uplink resource configuration approaches in NB-IoT networks. The simulation parameters are listed in Table I [1, 3, 21, 22, 30], and the hyperparameters for Q-learning are shown in Table II. All the simulations are based on Python, and the DNN-Q is built based on Keras library [42]. In the following, we present our simulation results of the single parameter single CE group scenario and the multi-parameters multiple CE groups scenario in Section VI-A and Section VI-B, respectively.

[11]The state function can be designed to collect more information according to the complexity requirements, such as sharing the value function between each DNN-Q agent [41].

A. Single Parameter Single CE Group Scenario

In the single CE group scenario, eNB is located at the center of a circular area with a 10 km radius, and the IoT devices are randomly located within the cell. We set the number of RACH periods as $n_{\text{Rach}} = 1$, the repetition value as $n_{\text{Repe}} = 4$, and the limited uplink resource as $R_{\text{uplink}} = 1536$ REs (i.e., 32 slots with 48 sub-carriers). Unless otherwise stated, we consider the number of periodical IoT devices to be $D_{\text{periodical}} = 10000$, and the number of bursty IoT devices to be $D_{\text{bursty}} = 5000$. In the training of DNN-Q, we consider the Q-network with three hidden layers each of which is consist of 128 ReLU units. The results of tabular-Q, LA-Q, and DNN-Q approaches are given in Section III.A, III.B.1), and III.B.2), respectively. The results of conventional LE-URC and FSI-URC approaches are given in Section III.C.

TABLE I: Simulation Parameters

Parameters	**Setting**	**Parameters**	**Setting**
Path-loss exponent η	4	noise power σ^2	-138 dBm
eNB broadcast power P_{NPBCH}	35 dBm	Path-loss inverse power control threshold ρ	120 dB
Maximal preamble transmit power $P_{\text{NPRACHmax}}$	23 dBm	The received SNR threshold γ_{th}	0 dB
Duration of periodic traffic T_{periodic}	1 hour	TTI	640ms
Duration of bursty traffic T_{bursty}	10 minutes	Set of number of preambles $\mathcal{N}_{\text{Prea}}$	$\{12, 24, 36, 48\}$
Maximum allowed resource requests γ_{RRC}	5	Set of repetition value $\mathcal{N}_{\text{Repe}}$	$\{1, 2, 4, 8, 16, 32\}$
Maximum allowed RACH attempts γ_{pMax}	10	Set of number of RACH periods $\mathcal{N}_{\text{Rach}}$	$\{1, 2, 4\}$
REs required for B_{RACH}	4	REs required for B_{DATA}	32

TABLE II: Q-learning Hyperparameters

Hyperparameters	**Value**	**Hyperparameters**	**Value**
Learning rate λ for Tabular-Q and LA-Q	0.01	Learning rate by RMSProp λ_{RMS} for DNN-Q	0.0001
Initial exploration ϵ	1	Final exploration ϵ	0.1
Discount rate γ	0.5	Minibatch size	32
Replay memory	10000	Target Q-network update frequency	1000

Fig. 6(a) plots the actual number of IoT devices executing the first RACH procedure, which illustrates the trend of IoT devices received a newly inter-arrival packet based on the proposed traffic models given in Eq. (2) and Eq. (4). Fig. 6(b) plot the number of successfully served IoT devices V_{succ} in our proposed FSI-URC, LE-URC, and DNN-Q approaches. In Fig. 6(b), with the time increases, V_{succ} first increases gradually with the increasing of traffic shown in Fig. 2(a), until it reaches the serving capacity of eNB. Then, V_{succ} decreases slowly due to the increasing collisions and scheduling failures with the increase of traffic. After that, V_{succ} increases gradually as the collisions and scheduling failures decrease with the decreasing of traffic. Finally, V_{succ} decreases slowly with the decreasing of traffic.

In Fig. 6(b), we see that the ideal FSI-URC approach outperforms the LE-URC approach, due to that the FSI-URC approach uses the actual network load to perfectly optimize the V_{succ}^t at one time instance as Eq. (14). DNN-Q not only always outperforms LE-URC, but also exceeds the ideal DSI-URC approach in most

of TTIs. This is due to that both LE-URC and FSI-URC only optimize V_{succ}^{t} at one time instance, whereas DNN-Q optimizes the long-term performance of the number of served IoT devices. The optimization in one time instance (LE-URC and FSI-URC) only takes into account the current trade-off between RACH resource and DATA resource given in Eq. (25), while the optimization over long-term period (DNN-Q) also accounts for some long-term hidden features, such as the dropping packets due to exceeding them maximum RACH attempts γ_{pMax} or maximum resource requests γ_{RRC}. The DNN-Q approach can well capture these hidden features to optimize the long-term performance of V_{succ} as Eq. (13).

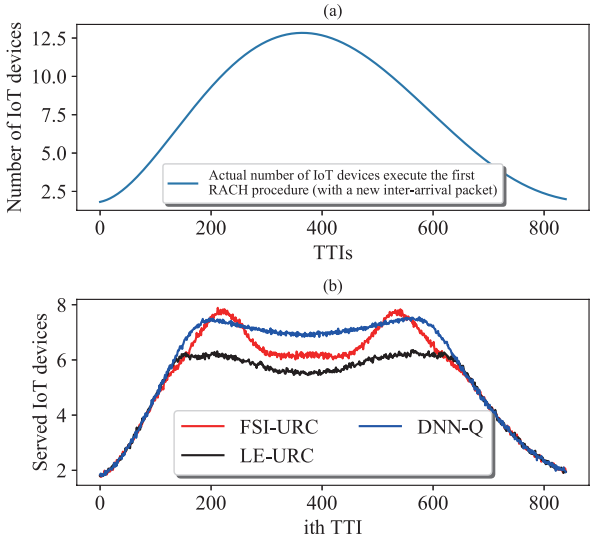

Fig. 6: The actual number of IoT devices executing the first RACH procedure and V_{succ} for FSI-URC, LE-URC, and DNN-Q.

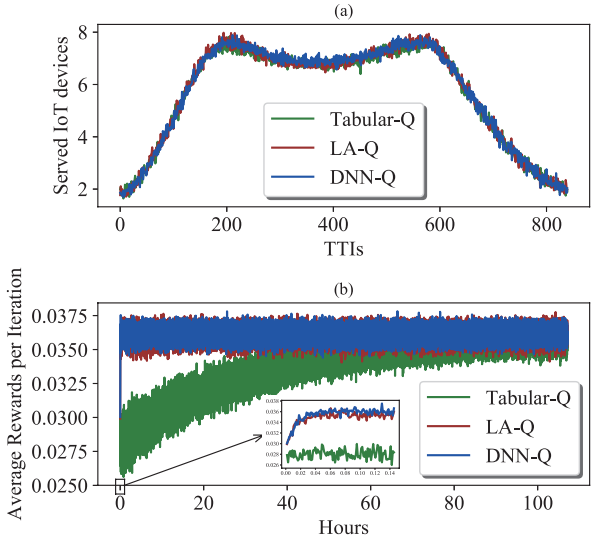

Fig. 7: V_{succ} and the average received reward for Tabular-Q, LA-Q, and DNN-Q.

Fig. 7(a) compares the number of successfully served IoT devices V_{succ} under Tabular-Q, LA-Q, and DNN-Q approaches. We observe that all these three approaches achieve similar values of V_{succ}, which indicates that both LA and DNN approaches can well estimate the optimal value function $Q^*(s,a)$ as the converged Tabular-Q approach in this low complexity single CE group scenario. Fig. 7(b) plots the average received reward over each bursty duration $\mathbb{E}\{R\} = \frac{1}{T_{\text{bursty}}} \sum_{t=0}^{T_{\text{bursty}}} R^t$ (i.e., considering one bursty duration T_{bursty} as an iteration) from the beginning of the training versus the required training time. It can be seen that LA-Q and DNN-Q converge to the optimal value function $Q^*(s,a)$ (1 hour) much faster than that of Tabular-Q (5 days). The observations in Fig. 7 demonstrate that LA-Q and DNN-Q can be good alternatives for tabular-Q to achieve almost same number of served IoT devices with much less training time.

Fig. 8(a) and Fig. 8(b) plot the average number of successfully served IoT devices $\mathbb{E}\{V_{\text{succ}}\}$ and the average number of dropped packets $\mathbb{E}\{V_{\text{drop}}\}$ over a bursty duration T_{bursty} versus the number of bursty IoT devices D_{bursty}. In Fig. 8(a), we observe that $\mathbb{E}\{V_{\text{succ}}\}$ first increases and then decreases with increasing the number of bursty devices, the decreasing trend starts when eNB can not afford to serve the increasing

IoT device number due to the increasing collisions and scheduling failures. These collisions and scheduling failures also result in the increasing number of packet drops with increasing traffics as shown in Fig. 8(b). In Fig. 8, we also notice that 8 DNN-Q always outperforms LE-DRA (especially for relatively large D_{bursty}), which indicates the superiority of DNN-Q approach in handling massive bursty IoT devices. Interestingly, DNN-Q provides better performance of the number of served IoT devices and smaller mean errors than the ideal FSI-DRA approach in most cases, which thanks to the long-term optimization capability of DNN-Q.

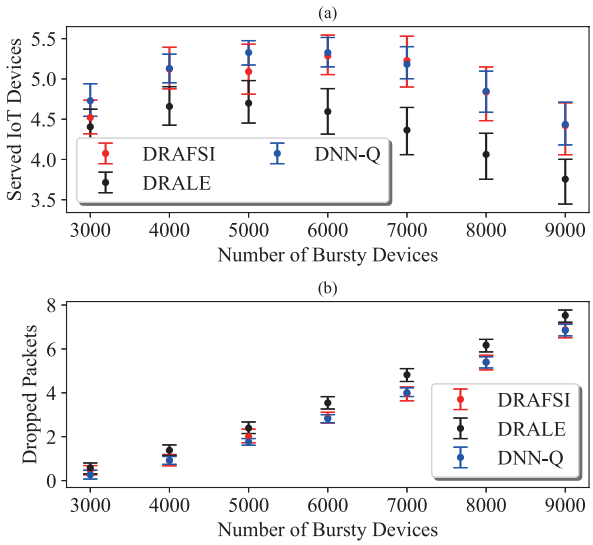

Fig. 8: $\mathbb{E}\{V_{\text{succ}}\}$ and $\mathbb{E}\{V_{\text{drop}}\}$ for FSI-URC, LE-URC, and DNN-Q.

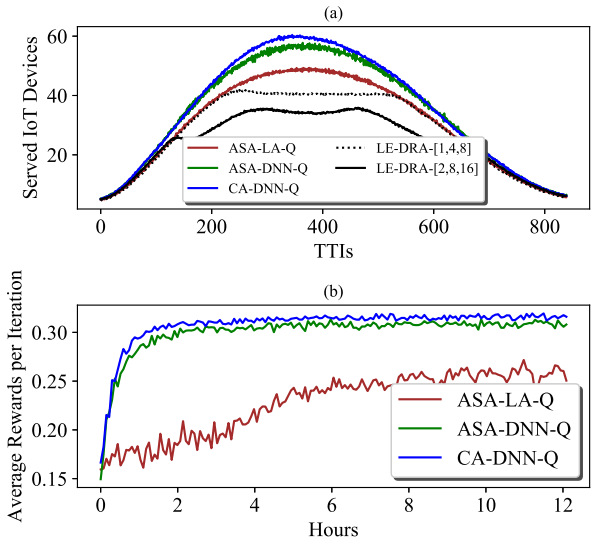

Fig. 9: V_{succ} and the average received reward.

B. Multiple Configuration Multiple CE Groups Scenario

In the multiple CE group scenario, eNB is located at the center of a circle area with 12 km radius. We consider RSRP thresholds for CE group choosing $\{\gamma_{\text{RSRP1}}, \gamma_{\text{RSRP2}}\} = \{0, -5\}$dB, the uplink resource $R_{\text{uplink}} = 15360$ REs (i.e., 320 slots with 48 sub-carriers), and the NPUSCH constrains for LE-DRA is $\{R_{\text{uplink},0}, R_{\text{uplink},0}, R_{\text{uplink},0}\} = \{3072, 6144, 6144\}$. To model the massive IoT traffic, both the number of periodical IoT devices $D_{\text{periodical}}$ and the number of bursty IoT devices D_{bursty} increase to 30000.

For the training of Q-agent, we consider $\alpha = 1$ in $\alpha - fairness$, and each CE group uses the same reward weight ($w_{r,0} = w_{r,1} = w_{r,2} = 1$ in Eq. (33)). In ASA-DNN-Q, we consider the Q-network with 3 hidden layers each of which is consist of 2048 ReLU units. In CA-DNN-Q, we consider 6 Q-networks with 9 hidden layers each of which is consist of 128 ReLU units, where each Q-network independently controls the number of RACH periods $n_{\text{Rach},i}$, the repetition value $n_{\text{Repe},i}$, and the number of preambles in each RACH period $f_{\text{Prea},i}$. The results of ASA-LA-Q and ASA-DNN-Q approaches are given in Section V.B.1), and results of CA-DNN-Q approach is given in Section IV.B.2).

Fig. 9(a) plots the the number of successfully served IoT devices V_{succ} versus the number of TTIs. LE-DRA-[1,4,8] and LE-DRA-[2,8,16] represent LE-DRA approach with the repetition values $\{N_{\text{r},0}, N_{\text{r},1}, N_{\text{r},2}\}$ equaling to $\{1, 4, 8\}$ and $\{2, 8, 16\}$, respectively. We observes that the number of successfully served IoT devices V_{succ} follows CA-DNN-Q > ASA-DNN-Q > ASA-LA-Q $\gg$ LE-DRA-[1,4,8] $\gg$ LE-DRA-[2,8,16]. The reason for LE-DRA-[1,4,8] $\gg$ LE-DRA-[2,8,16] is that the small repetition values ([1,4,8]) consumes fewer channel resource, which reduces scheduling failures in heavy traffic scenarios. As can be seen, all Q-learning based approaches substantially outperform LE-DRA approaches, due to that the Q-learning based approaches can dynamically optimize the number of served IoT devices by accurately configuring both repetition value $N_{\text{r},i}$ and RAOs.

Fig. 9(b) plots the average received reward over each bursty duration $\mathbb{E}\{R\} = \frac{1}{T_{\text{bursty}}} \sum_{t=0}^{T_{\text{bursty}}} R^t$ from the beginning of the training versus the consumed training time. It can be seen that CA-DNN-Q and ASA-DNN-Q outperform ASA-LA-Q in terms of less training time. Compared with the results in the single CE group scenario shown in Fig. 7, DNN is a better value function approximator for the 3 CE groups scenario due to its efficiency and capability in solving high complexity problems (i.e., the complexity of the problem increases with the increasing CE groups N_{CE}). We also seen that CA-DNN-Q achieves higher $\mathbb{E}\{R\}^*$ than ASA-DNN-Q, due to that CA-DNN-Q can accurately select the exact values from the set of actions $\{\mathcal{N}_{\text{r}}, \mathcal{N}_{\text{p}}, \mathcal{N}_{\text{d}}\}$, whereas ASA-DNN-Q can only select ascent/descent actions, which leads to a sub-optimal solution.

Fig. 10 plots the number of successfully served IoT devices V_{succ}, the number of dropped packets V_{drop}, and the number of dropped packets V_{drop}, and Fig. 11 plots the configured number of RACH opportunities (i.e., $n_{\text{Rach},i} \times f_{\text{Prea},i}$, a.k.a. RAOs) and the repetition value $n_{\text{Repe},i}$ with CA-DNN-Q during one bursty duration, due to its best performance in terms of V_{succ} (as shown in Fig. 9(a)). Fig. 10 demonstrates that the number of successfully served IoT devices in each CE group follows $V_{\text{succ},0} > V_{\text{succ},1} > V_{\text{succ},2}$. This is due to the larger number of IoT devices with lower SNR outage in group 0 (i. e., IoT devices in group 0 are located much closer to eNB than IoT devices in group 1 and group 2). Fig. 11 presents how the Q-agents select the number of RAOs and the repetition value $n_{\text{Repe},i}$ to optimize the overall number of served IoT devices. It can be seen with the increase of V_{drop} Q-agents reduce the RAOs of group 2, and the repetition values of group 1 $n_{\text{Repe},1}$ and group 2 $n_{\text{Repe},2}$ to save the data resource R^t_{DATA} to schedule more IoT devices. Surprisingly, the Q-agent increases the repetition value of group 0 $n_{\text{Repe},0}$ at the same time, which is completely opposite to the actions of $n_{\text{Repe},1}$ and $n_{\text{Repe},2}$. This is due to that the Q-agent is aware of the key to optimize the overall performance V_{succ} is to guarantee $V_{\text{succ},0}$, as the IoT devices in the CE group 0 are easier to be served, due to they are located close to the eNB and consume less resource. To do so, the Q-agent increases $n_{\text{Repe},0}$ to mitigate that IoT devices waste RACH attempts due to the SNR outage, which mitigate the dropped packets

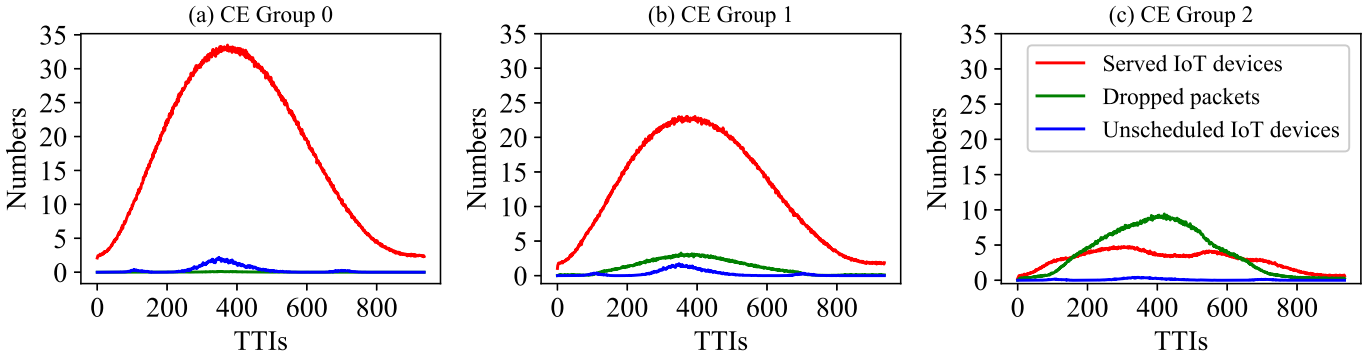

Fig. 10: The number of received packets per BS.

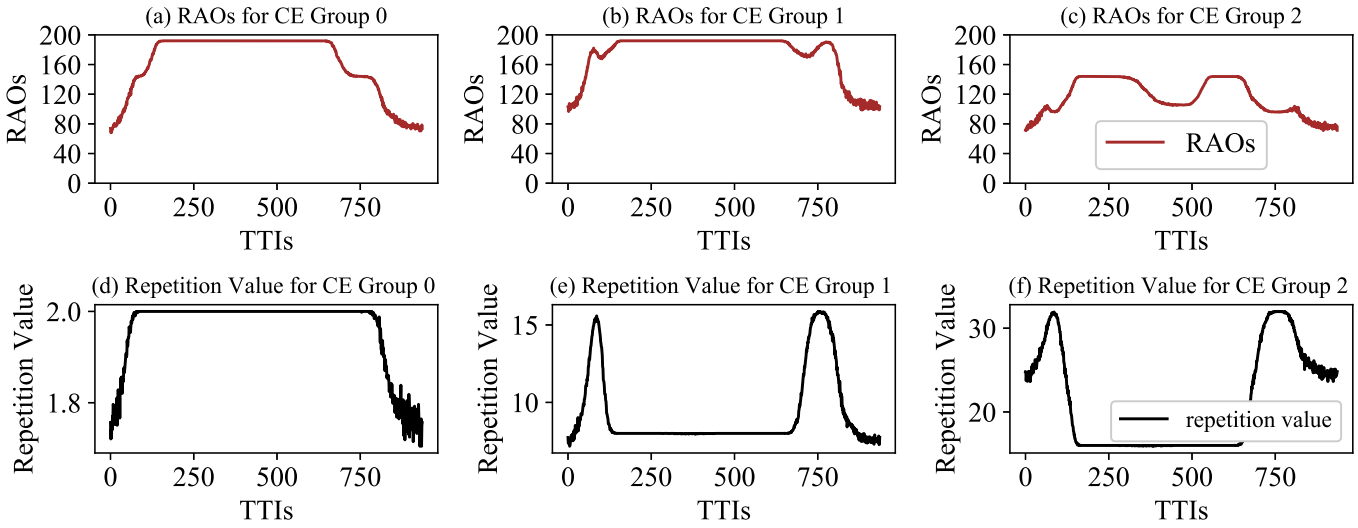

Fig. 11: Preamble transmission success probability.

caused by massive RACH failures in the CE group 0. Overall, the proposed Q-learning based uplink resource configuration approaches can dynamically optimize the number of served IoT devices in the high-complexity scenarios, which intelligently well-serve massive IoT devices with extended coverage in NB-IoT networks.

VII. CONCLUSION

In this paper, we developed six Q-learning based uplink resource configuration approaches, which can optimize the number of served IoT devices in real-time in NB-IoT networks. We first developed the tabular-Q, LA-Q, and DNN-Q based approaches for the single parameter single CE group, which outperform the conventional LE-URC and FSI-URC approaches in terms of the number of served IoT devices. Our results demonstrated that the LA-Q and DNN-Q approaches can be good alternatives for the tabular-Q approach to achieve almost the same number of served IoT devices with much less training time. To support traffic with different coverage requirements, we then studied the multi-parameters multiple CE groups scenario as defined in NB-IoT standard, which introduced the problem of high-dimensional configurations. To solve it,

we advanced the LA-Q and DNN-Q approaches using the actions simplification (AS-LA-Q and AS-DNN-Q), which guarantees its convergent capability by sacrificing the accuracy in resource configuration. We further developed the CMC-DNN-Q approach by dividing high-dimensional configurations into multiple parallel sub-tasks, which achieves the best performance in terms of the number of successfully served IoT devices V_{succ} with the least training time.

REFERENCES

[1] J. Schlienz and D. Raddino, "Narrowband internet of things whitepaper," *IEEE Microw. Mag.*, vol. 8, no. 1, pp. 76–82, 2016.

[2] H. S. Dhillon, H. Huang, and H. Viswanathan, "Wide-area wireless communication challenges for the internet of things," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 168–174, Feb. 2017.

[3] Y.-P. E. Wang, X. Lin, A. Adhikary, A. Grovlen, Y. Sui, Y. Blankenship, J. Bergman, and H. S. Razaghi, "A primer on 3GPP narrowband internet of things (NB-IoT)," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 117–123, Mar. 2017.

[4] R. Harwahyu, R.-G. Cheng, C.-H. Wei, and R. F. Sari, "Optimization of random access channel in NB-IoT," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 391–402, Feb. 2018.

[5] S.-M. Oh and J. Shin, "An efficient small data transmission scheme in the 3GPP NB-IoT system," *IEEE Commun. Lett.*, vol. 21, no. 3, pp. 660–663, Mar. 2017.

[6] H. Malik, H. Pervaiz, M. M. Alam, Y. Le Moullec, A. Kuusik, and M. A. Imran, "Radio resource management scheme in NB-IoT systems," *IEEE Access*, vol. 6, pp. 15 051–15 064, 2018.

[7] C. Yu, L. Yu, Y. Wu, Y. He, and Q. Lu, "Uplink scheduling and link adaptation for narrowband internet of things systems," *IEEE Access*, vol. 5, pp. 1724–1734, 2017.

[8] D. T. Wiriaatmadja and K. W. Choi, "Hybrid random access and data transmission protocol for machine-to-machine communications in cellular networks," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 33–46, Jan. 2015.

[9] S. Duan, V. Shah-Mansouri, Z. Wang, and V. W. 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.

[10] L. M. Bello, P. Mitchell, and D. Grace, "Application of Q-learning for RACH access to support M2M traffic over a cellular network," in *Proc. European Wireless Conf.*, 2014, pp. 1–6.

[11] Y. Chu, P. D. Mitchell, and D. Grace, "ALOHA and Q-learning based medium access control for wireless sensor networks," in *Int. Symp. Wireless Commun. Syst. (ISWCS)*, 2012, pp. 511–515.

[12] Y. Yan, P. Mitchell, T. Clarke, and D. Grace, "Distributed frame size selection for a Q learning based slotted ALOHA protocol," in *Int. Symp. Wireless Commun. Syst. (ISWCS)*, 2013, pp. 1–5.

[13] G. Naddafzadeh-Shirazi, P.-Y. Kong, and C.-K. Tham, "Distributed reinforcement learning frameworks for cooperative retransmission in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 8, pp. 4157–4162, Oct. 2010.

[14] Y.-S. Chen, C.-J. Chang, and F.-C. Ren, "Q-learning-based multirate transmission control scheme for RRM in multimedia WCDMA systems," *IEEE Trans. Veh. Technol.*, vol. 53, no. 1, pp. 38–48, Jan. 2004.

[15] M. ihun and L. Yujin, "A reinforcement learning approach to access management in wireless cellular networks," in *Wireless Commun. Mobile Comput.*, May. 2017, pp. 1–7.

[16] T.-O. Luis, P.-P. Diego, P. Vicent, and M.-B. Jorge, "Reinforcement learning-based ACB in LTE-A networks for handling massive M2M and H2H communications," in *IEEE Int. Commun. Conf. (ICC)*, May. 2018, pp. 1–7.

[17] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, Feb. 2015.

[18] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning." in *Assoc. Adv. AI (AAAI)*, vol. 2, Feb. 2016, p. 5.

[19] A. Laya, L. Alonso, and J. Alonso-Zarate, "Is the random access channel of LTE and LTE-A suitable for M2M communications? A survey of alternatives." *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 4–16, Jan. 2014.

[20] E. Dahlman, S. Parkvall, and J. Skold, *4G: LTE/LTE-advanced for mobile broadband*. Academic press, 2013.

[21] "Evolved universal terrestrial radio access (E-UTRA); Physical channels and modulation," *3GPP TS 36.211 v.14.2.0*, 2017.

[22] "Cellular system support for ultra-low complexity and low throughput Internet of Things (CIoT)," *3GPP TR 45.820 V13.1.0,*, Nov. 2015.

[23] "Study on RAN improvements for machine-type communications," *3GPP TR 37.868 V11.0.0,*, Sep. 2011.

[24] A. K. Gupta and S. Nadarajah, *Handbook of Beta distribution and its applications*. New York, USA: CRC press, 2004.

[25] "Evolved universal terrestrial radio access (E-UTRA); Requirements for support of radio resource management," *3GPP TS 36.133 v. 14.3.0*, Apr. 2017.

[26] "Evolved universal terrestrial radio access (E-UTRA); Radio resource control protocol specification," *3GPP TS 36.331 v. 14.2.2*, May. 2017.

[27] "Evolved universal terrestrial radio access (E-UTRA); Physical layer measurements," *3GPP TS 36.214 v. 14.2.0*, Apr. 2017.

[28] X. Lin, A. Adhikary, and Y.-P. E. Wang, "Random access preamble design and detection for 3GPP narrowband IoT systems," *IEEE Wireless Commun. Lett.*, vol. 5, no. 6, pp. 640–643, Jun. 2016.

[29] N. Jiang, Y. Deng, M. Condoluci, W. Guo, A. Nallanathan, and M. Dohler, "RACH preamble repetition in NB-IoT network," *IEEE Commun. Lett.*, vol. 22, no. 6, pp. 1244–1247, Jun. 2018.

[30] "Evolved universal terrestrial radio access (E-UTRA); Medium Access Control protocol specification," *3GPP TS 36.321 v.14.2.1*, 2017.

[31] R. Sutton and A. Barto, "Reinforcement learning: An introduction (draft)," *URl: http://www.incompleteideas.net/book/bookdraft2017nov5.pdf*, 2017.

[32] G. Konidaris, S. Osentoski, and P. S. Thomas, "Value function approximation in reinforcement learning using the Fourier basis." in *Assoc. Adv. AI (AAAI)*, vol. 6, Aug. 2011, p. 7.

[33] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," in *Proc. Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

[34] M. Hauskrecht, "Value-function approximations for partially observable markov decision processes," *J. AI Res.*, vol. 13, pp. 33–94, Aug. 2000.

[35] A. Geramifard *et al.*, "A tutorial on linear function approximators for dynamic programming and reinforcement learning," *Found. Trends Mach. Learn.*, vol. 6, no. 4, pp. 375–451, 2013.

[36] F. S. Melo and M. I. Ribeiro, "Q-learning with linear function approximation," in *Springer Int. Conf. Comput. Learn. Theory*, 2007, pp. 308–322.

[37] C. Xi, K. Bohdan, M. Norman, and M. Pete, "Polynomial regression as an alternative to neural nets," *arXiv preprint arXiv:1806.06850*, 2018.

[38] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, USA: Springer print, 2006.

[39] T. Tieleman and G. Hinton, "Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Netw. Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012.

[40] T. Lan, D. Kao, M. Chiang, and A. Sabharwal, *An axiomatic theory of fairness in network resource allocation*. IEEE press, 2010.

[41] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, C, Appl. Rev., 38 (2)*, 2008.

[42] F. Chollet *et al.*, "Keras," *URL: https://keras.io/*, 2015.