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 technology that offers a range of flexible configurations for massive IoT radio access from groups of devices with heterogeneous requirements. A configuration specifies the amount of radio resource allocated to each group of devices for random access and for data transmission. Assuming no knowledge of the traffic statistics, there exists an important challenge in "how to determine the configuration that maximizes the long-term average number of served IoT devices at each Transmission Time Interval (TTI) in an online fashion". Given the complexity of searching for optimal configuration, we first develop real-time 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 (DQN) in the single-parameter single-group scenario. Our results show that the proposed reinforcement learning based approaches considerably outperform the conventional heuristic approaches based on load estimation (LE-URC) in terms of the number of served IoT devices. This result also indicates that LA-Q and DQN can be good alternatives for tabular-Q to achieve almost the same performance with much less training time. We further advance LA-Q and DQN via Actions Aggregation (AA-LA-Q and AA-DQN) and via Cooperative Multi-Agent learning (CMA-DQN) for the multi-parameter multi-group scenario, thereby solve the problem that Q-learning agents do not converge in high-dimensional configurations. In this scenario, the superiority of the proposed Q-learning approaches over the conventional LE-URC approach significantly improves with the increase of configuration dimensions, and the CMA-DQN approach outperforms the other approaches in both throughput and training efficiency.

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) [2]. NB-IoT is expected to provide reliable wireless access for IoT devices with various types of data traffic, and to meet the requirement of extended coverage. As most mIoT applications favor delay-tolerant uplink data traffic with small size, such as data from

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) (Corresponding author: Yansha Deng).

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

alarms, and meters, monitors, the key target of NB-IoT design is to deal with the sporadic uplink transmissions of massive IoT devices [3].

NB-IoT is built from legacy Long-Term Evolution (LTE) design, but only deploys in a narrow bandwidth (180 KHz) for Coverage Enhancement (CE) [4]. Different from the legacy LTE, NB-IoT only defines two uplink physical channel resource to perform all the uplink transmission, including the Random Access CHannel (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 of IoT devices sharing the uplink resource in the same band. Each group serves IoT devices with different coverage requirements distinguishing based on a same broadcast signal (i.e., using Narrowband Physical Broadcast CHannel, a.k.a. NPBCH) from the evolved Node B (eNB) [4]. At the beginning of each uplink Transmission Time Interval (TTI), eNB selects a system configuration that specifies the radio resource allocated to each group in order to accommodate the RACH procedure along with the remaining resource for data transmission. The key challenge is to optimally balance the allocations of channel resource between the RACH procedure and data transmission so as to provide maximum success accesses and transmissions in NB-IoT networks. For instance, allocating too much resource for RACH to enhance access performance can result in insufficient resource for data transmission, while allocating too less RACH resource may lead to long access delay and uplink data resource over-allocation.

Unfortunately, dynamic RACH and data transmission resource configuration optimization is an untreated problem in NB-IoT. Generally speaking, the eNB observes the transmission receptions of both RACH (e.g., number of successfully received preambles and collisions) and data transmission (e.g., number of successful scheduling and unscheduling) for all groups at the end of each TTI. This historical information can be potentially used to predict traffic from all groups and to facilitate the optimization of future TTIs' configurations. Even if one knew all the relevant statistics, tackling this problem in an exact manner would result in a Partially Observable Markov Decision Process (POMDP) with large state and action spaces, which would be generally intractable. The complexity of the problem is compounded by the lack of a prior knowledge at the

eNB regarding the stochastic traffic and unobservable channel statistics (i.e., random collision, and effects of physical radio including path-loss as well as fading).

In this paper, we develop Reinforcement Learning (RL) based uplink resource configuration approaches to dynamically optimize the number of served IoT devices in NB-IoT networks. To showcase the efficiency, we compare the proposed RL-based approaches with the conventional heuristic uplink resource allocation approaches. The contributions can be summarized as follows:

- We develop an RL-based framework to optimize the number of served IoT devices by adaptively configuring uplink resource in NB-IoT networks. The uplink communication procedure in NB-IoT is simulated by taking into account the heterogeneous IoT traffics, the CE group selection, the RACH procedure, and the uplink data transmission resource scheduling. This generated simulation environment is used for training the RL-based agents before deployment, and these agents will be updated according to the real traffic in practical NB-IoT networks in an online manner.
- We first study a simplified NB-IoT scenario considering the single parameter and the single CE group, where a basic tabular Q-learning (tabular-Q) was developed to compare with the revised conventional Load Estimation based Uplink Resource Configuration (LE-URC) scheme. The tabular-Q is further advanced by implementing function approximators with different computational complexities, namely, Linear Approximator (LA-Q) and Deep Neural Networks (Deep Q-Network, a.k.a. DQN) to elaborate their capability and efficiency in dealing with high-dimensional state space.
- We then study a more practical NB-IoT scenario with multiple parameters and multiple CE groups, where direct implementation of LA-Q or DQN is not feasible due to the increasing size of the parameter combinations. To solve it, we propose Action Aggregation approaches based on LA-Q and DQN, namely, AA-LA-Q and AA-DQN, which guarantee convergence capability by sacrificing certain accuracy in the parameters selection. Finally, a Cooperative Multi-Agent learning based on DQN (CMA-DQN) is developed to break down the selection in high-dimensional parameters into multiple parallel sub-tasks by using that a number of DQN agents are cooperatively trained to produce each parameter for each CE group.
- In the simplified scenario, our results show that the number of served IoT devices with tabular-Q considerably outperforms that with LE-URC, while LA-Q and DQN achieve almost the same performance as that of tabular-Q using much less training time. In the practical scenario, the superiority of Q-learning based approaches over LE-URC significantly improves. Especially, CMA-DQN outperforms all other approaches in terms of both throughput and training efficiency, which is mainly due to the use of DQN enabling operation over a large state

space and the use of multiple agents dealing with the large dimensionality of parameters selection.

The rest of the paper is organized as follows. Section II presents related works. Section III provides the problem formulation and system model. Section IV illustrates the preliminary and the conventional LE-URC. Section V proposes Q-learning based uplink resource configuration approaches in the single-parameter single-group scenario. Section VI presents the advanced Q-learning based approaches in the multi-parameter multi-group scenario. Section VII elaborates the numerical results, and finally, Section VIII summarizes the conclusion and future work.

II. RELATED WORKS

1) Real-time optimization in cellular-based networks: In light of the proposed POMDP challenge, prior works [5, 6] studied real-time resource configuration of RACH procedure and/or data transmission by proposing dynamic Access Class Barring (ACB) schemes to optimize the number of served IoT devices. These optimization problems have been tackled under the simplified assumptions that at most two configurations are allowed and that the optimization is executed for a single group without considering errors due to wireless transmission. In order to consider more complex and practical formulations, RL emerges as a natural solution given its capability in interacting with the practical environment and feedback in the form of the number of successful and unsuccessful transmissions per TTI. Distributed RL based on tabular-Q has been proposed in [7–10]. In [7–9], the authors studied distributed tabular-Q in slotted-Aloha networks, where each device learns how to avoid collisions by finding a proper time slot to transmit packets. In [10], the authors implemented tabular-Q agents at the relay nodes for cooperatively selecting its transmit power and transmission probability to optimize the total number of useful received packets per consumed energy. Centralized RL has also been studied in [11–13], where the RL agent was implemented at the base station site. In [11], a learning-based scheme was proposed for radio resource management in multimedia wide-band code-division multiple access systems to improve spectrum utilization. In [12, 13], the authors studied the tabular-Q based ACB schemes in cellular networks, where a Q-agent was implemented at an eNB aiming at selecting the optimal ACB factor to maximize the access success probability of RACH procedure. Unfortunately, the tabular-Q framework in [12, 13] cannot be used to solve the multi-parameter multi-group optimization problem in uplink resource configuration of NB-IoT networks, due to their inability to address high-dimensional state space and variable selection. More importantly, whether their proposed RL-based resource configuration approaches [12, 13] outperform the conventional resource configuration approaches [5, 6] is still unknown.

2) Optimization in NB-IoT networks: In NB-IoT networks, most existing studies either focused on the resource allocation during RACH procedure [14, 15], or that during the data transmission [16, 17]. For RACH procedure, the access success probability was statistically optimized in [14] using exhaustive
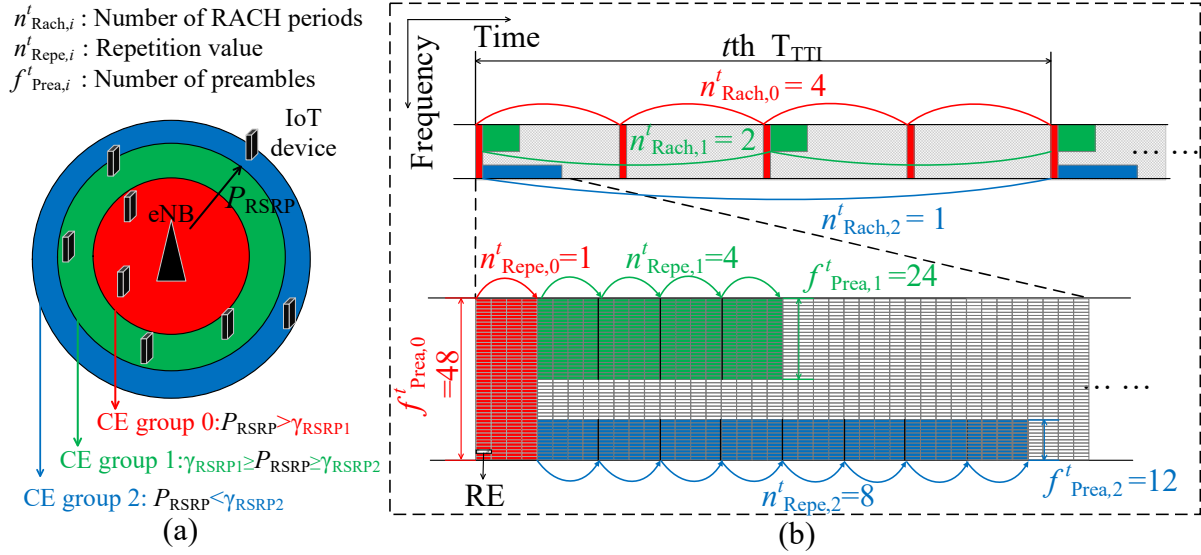
Fig. 1: (a) Illustration of system model; (b) Uplink channel frame structure.

search, and the authors in [15] studied fixed-size data resource scheduling for various resource requirements. For data transmission, [16] presented an uplink data transmission time slot and power allocation scheme to optimize the overall channel gain, and [17] proposed a link adaptation scheme, which selects modulation and coding level, and the repetition value according to the acknowledgment/negative-acknowledgment feedback to reduce the uplink data transmission block error ratio. More importantly, these works ignore the time-varied heterogeneous traffic of massive IoT devices, and considered a snap shot [14, 16, 17] or steady-state behavior [15] of NB-IoT networks. To our knowledge, the most relevant work is [18], where the authors studied the steady-state behavior of NB-IoT networks from the perspective of a single device. Optimizing some of the parameters of the NB-IoT configuration, namely the repetition value (to be defined below) and time intervals between two consecutive scheduling of NPRACH and NPDCCH, was carried out in terms of latency and power consumption in [18] using a queueing framework.

III. PROBLEM FORMULATION AND SYSTEM MODEL

As illustrated in Fig. 1(a), we consider a single-cell NB-IoT network composed of an eNB located at the center of the cell, and a set of static IoT devices randomly located in an area of the plane $\mathbb{R}^2$, and remain spatially static once deployed. The devices are divided into three CE groups as further discussed below, and the eNB is unaware of the status of these IoT devices, hence no uplink channel resource is scheduled to them in advance. In each IoT device, uplink data is generated according to random inter-arrival processes over the TTIs, which are Markovian and possibly time-varying.

A. Problem Formulation

With packets waiting for service, an IoT device executes the contention-based RACH procedure in order to establish a Radio Resource Control (RRC) connection with the eNB.

The contention-based RACH procedure consists of four steps, where an IoT device transmits a randomly selected preamble, for a given number of times according to the repetition value $n_{\text{Repe},i}^t$ [2], to initial RACH procedure in step 1, and exchanges control information with the eNB in the next three steps [19]. The RACH process can fail if: (i) a collision occurs when two or more IoT devices selecting the same preamble; or (ii) there is no collision, but the eNB cannot detect a preamble due to low SNR. Note that a collision can be still detected in step 3 of RACH when the collided preambles are not detected in step 1 of RACH following 3GPP report [20]. This assumption is different from our previous works [21, 22], which only focuses on the preamble detection analysis in step 1 of RACH.

As shown in Fig. 1(b), for each TTI t and for each CE group $i = 0, 1, 2$, in order to reduce the chance of a collision, the eNB can increase the number $n_{\text{Rach},i}^t$ of RACH periods in the TTI or the number $f_{\text{Prea},i}^t$ of preambles available in each RACH period [23]. Furthermore, in order to mitigate the SNR outage, the eNB can increase the number $n_{\text{Repe},i}^t$ of times that a preamble transmission is repeated by a device in group i in one RACH period [23] of the TTI.

After the RRC connection is established, the IoT device requests uplink channel resource from the eNB for control information and data transmission. As shown in Fig. 1(b), given a total number of resource R_{Uplink} for uplink transmission in the TTI, the number of available resource for data transmission R_{DATA}^t is written as $R_{\text{DATA}}^t = R_{\text{Uplink}} - R_{\text{RACH}}^t$, where R_{RACH}^t is the overall number of Resource Elements (REs)[1] allocated for the RACH procedure. This can be computed as $R_{\text{RACH}}^t = B_{\text{RACH}} \sum_{i=0}^{2} n_{\text{Rach},i} n_{\text{Repe},i} f_{\text{Prea},i}$, where B_{RACH} measures the number of REs required for one preamble transmission.

---

[1] The uplink channel consists of 48 sub-carriers within 180 kHz bandwidth. With a 3.75 kHz tone spacing, one RE is composed of one time slot of 2 ms and one sub-carrier of 3.75 kHz [2]. 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 [2].

In this work, we tackle the problem of optimizing the RACH configuration defined by parameters $A^t = \{n_{\text{Rach},i}^t, f_{\text{Prea},i}^t, n_{\text{Repe},i}^t\}_{i=0}^2$ for each ith group in an online manner for every TTI t. In order to make this decision at the beginning of every TTI t, the eNB accesses all prior history $U^{t'}$ in TTIs $t' = 1, ..., t-1$ consisting of the following variables: the number of the collided preambles $V_{\text{cp},i}^{t'}$, the number of the successfully received preambles $V_{\text{sp},i}^{t'}$, and the number of idle preambles $V_{\text{ip},i}^{t'}$ of the ith CE group in the tth TTI for the RACH, as well as the number of IoT devices that have successfully sent data $V_{\text{su},i}^{t'}$ and the number of IoT devices that are waiting for being allocated data resource $V_{\text{un},i}^{t'}$. We denote $O^t = \{A^{t-1}, U^{t-1}, A^{t-2}, U^{t-2}, \cdots, A^1, U^1\}$ as the observed history of all such measurements and past actions.

The eNB aims at maximizing the long-term average number of devices that successfully transmit data with respect to the stochastic policy π that maps the current observation history O^t to the probabilities of selecting each possible configuration A^t. This problem can be formulated as the optimization

$$(\text{P1}): \max_{\{\pi(A^t|O^t)\}} \sum_{k=t}^{\infty} \sum_{i=0}^{2} \gamma^{k-t} \mathbb{E}_\pi[V_{\text{su},i}^k], \qquad (1)$$

where $\gamma \in [0, 1)$ is the discount rate for the performance in future TTIs and index i runs over the CE groups. Since the dynamics of the system is Markovian over the TTI and is defined by the NB-IoT protocol to be further discussed below, this is a POMDP problem that is generally intractable. Approximate solutions will be discussed in Sections IV, V, and VI.

B. NB-IoT Access Network

We now provide additional details on the model and on the NB-IoT protocol. 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. Fig. 2 presents the uplink data transmission procedure from the perspective of an IoT device in NB-IoT networks, which consists of the four stages that are explained in the following four subsections to introduce the system model.

1) Traffic Inter-Arrival: We consider two types of IoT devices with different traffic models, including periodical traffic and bursty traffic, which is a heterogeneous traffic scenario for diverse IoT applications [24, 25]. The periodical traffic coming from periodic uplink reporting tasks, such as metering or environmental monitoring, is the most common traffic model in NB-IoT networks [26]. The bursty traffic due to emergency events, such as fire alarms and earthquake alarms, captures the complementary scenario in which a massive number of IoT devices tries to establish RRC connection with the eNB [20]. 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 an 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

$$\mu_{\text{period}}^t = \frac{T_{\text{TTI}}}{n_{\text{Rach},i}^t} \times \frac{1}{T_{\text{periodic}}}, \qquad (2)$$

where $n_{\text{Rach},i}^t$ is the number of RACH periods in the tth TTI, $\frac{T_{\text{TTI}}}{n_{\text{Rach},i}^t}$ 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 jth RACH period of the tth TTI is given by

$$\mu_{\text{bursty}}^{t,j} = \int_{\tau_{j-1}}^{\tau_j} p(\tau)d\tau, \qquad (3)$$

where τ_j is the starting time of the jth RACH period in the tth TTI, $\tau_j - \tau_{j-1} = \frac{T_{\text{TTI}}}{n_{\text{Rach},i}^t}$, and the distribution $p(\tau)$ follows the time limited Beta profile given as [20, Section 6.1.1]

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

In (4), $\text{Beta}(\alpha, \beta)$ is the Beta function with the constant parameters α and β [27].

2) CE Group Determination: Once an IoT device is backlogged, it first determines its associated CE group by comparing the received power of the broadcast signal P_{RSRP} to the Reference Signal Received Power (RSRP) thresholds $\{\gamma_{\text{RSRP1}}, \gamma_{\text{RSRP2}}\}$ according to the rule [28]

$$\begin{cases} \text{CE group 0}, & \text{if } P_{\text{RSRP}} > \gamma_{\text{RSRP1}}, \\ \text{CE group 1}, & \text{if } \gamma_{\text{RSRP1}} \geq P_{\text{RSRP}} \geq \gamma_{\text{RSRP2}}, \\ \text{CE group 2}, & \text{if } P_{\text{RSRP}} < \gamma_{\text{RSRP2}}. \end{cases} \qquad (5)$$

In (5), the received power of broadcast signal P_{RSRP} is expressed as

$$P_{\text{RSRP}} = P_{\text{NPBCH}}u^{-\eta}, \qquad (6)$$

where u is the device's distance from the eNB, and P_{NPBCH} is the broadcast power of eNB [28]. Note that P_{RSRP} is obtained by averaging the small-scale Rayleigh fading of the received power [28].

3) RACH Procedure: After CE group determination, each backlogged IoT device in group i repeats a randomly selected preamble $n_{\text{Repe},i}^t$ times in the next RACH period by using a pseudo-random frequency hopping schedule. The pseudo-random hopping rule is based on the current repetition time as well as the Narrowband Physical Cell ID, and in one repetition, a preamble consists of four symbol groups, which are transmitted with fixed size frequency hopping [2, 21, 29]. Therefore, a preamble is successfully detected if *at least* one preamble repetition succeeds, which in turn happens if *all* of its four symbol groups are correctly decoded [21]. Assuming that correct detecting is determined by the SNR level $\text{SNR}_{\text{sg},j,k}^t$ for
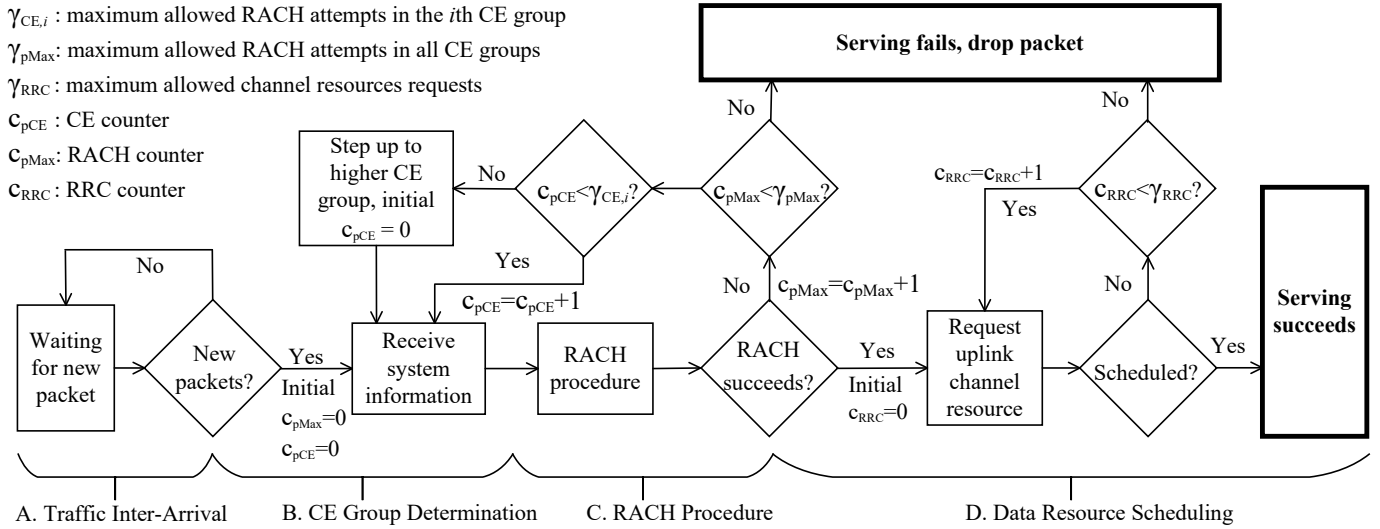
Fig. 2: Uplink data transmission procedure from the perspective of an IoT device in NB-IoT networks.

the jth repetition and the k symbol group, the correct detecting event S_{pd} can be expressed as

$$S_{pd} \triangleq \bigcup_{j=1}^{n_{\text{Repe},i}^{t}} \left(\bigcap_{k=1}^{4} \left\{ \text{SNR}_{\text{sg},j,k}^{t} \geq \gamma_{th} \right\} \right), \quad (7)$$

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

$$\text{SNR}_{\text{sg},j,k}^{t} = P_{\text{RACH},i} u^{-\eta} h / \sigma^2. \quad (8)$$

In (8), u is the Euclidean distance between the IoT device and the eNB, η 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

$$P_{\text{RACH},i} = \begin{cases} \min\left\{ \rho u^{\eta},\ P_{\text{RACHmax}} \right\}, & i = 0, \\ P_{\text{RACHmax}}, & i = 1 \text{ or } 2. \end{cases} \quad (9)$$

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 [28], which maintains the received signal power at the eNB from IoT devices with different distance equalling to the same threshold ρ, 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 [28].

As shown in the RACH procedure of Fig. 2, if a RACH fails, the IoT device reattempts the procedure until receiving a positive acknowledgement that RRC connection is established, or exceeding $\gamma_{\text{pCE},i}$ RACH attempts while being part of one CE group. If these attempts exceeds $\gamma_{\text{pCE},i}$, the device switches to a higher CE group if possible [30]. Moreover, the IoT device is allowed to attempt the RACH procedure no more than γ_{pMax} times before dropping its packets. These two features are counted by c_{pCE} and c_{pMax}, respectively.

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_{DATA}^{t} to the associated IoT device for control information and data transmission as shown in Fig 1(b). To allocate data resource among these devices, we adopt a basic random scheduling strategy, whereby an ordered list of all devices that have successfully completed the RACH procedure but have not received a data channel is compiled using a random order. In each TTI, devices in the list are considered in order for access to the data channel until the data resource are insufficient to serve the next device in the list. The remaining RRC connections between the unscheduled IoT devices and the eNB will be preserved within at most γ_{RRC} subsequent TTIs counting by c_{RRC}, and attempts will be made to schedule the device's data during these TTIs [30, 31]. The condition that the data resource are sufficient in TTI t is expressed as

$$R_{\text{DATA}}^{t} \geq \sum_{i=0}^{2} r_{\text{DATA},i}^{t} V_{\text{sch},i}^{t}, \quad (10)$$

where $\sum_{i=0}^{2} V_{\text{sch},i}^{t} \leq \sum_{i=0}^{2} (V_{\text{sp},i}^{t} + V_{\text{un},i}^{t-1})$ is the number of scheduled devices limited by the upper bound denoted by IoT devices with successful RACH $V_{\text{sp},i}^{t}$ in the current TTI t as well as unscheduled IoT devices $V_{\text{un},i}^{t-1}$ in the last TTI $(t-1)$, $r_{\text{DATA},i}^{t} = B_{\text{DATA}} \times n_{\text{Repe},i}^{t}$ is the number of required REs for serving one IoT device within the ith CE group, and B_{DATA} is the number of REs per repetition for control signal and data transmission[2]. Note that $n_{\text{Repe},i}^{t}$ is the repetition value for the ith CE group in the tth TTI, which is the same as for preamble transmission [2].

---

[2]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 [4, 23].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2019.2904366, IEEE Journal on Selected Areas in Communications

6

IV. PRELIMINARY AND CONVENTIONAL SOLUTIONS

A. Preliminary

The optimized number of served IoT devices over the long term given in Eq. (1) is really complicated, which cannot be easily solved via the conventional uplink resource approach. Therefore, most prior 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 [5, 6], which is expressed as

$$(P2): \quad \max_{\pi(x|O^t)} \quad \mathbb{E}_\pi[V_{\mathrm{su},0}^t], \tag{11}$$

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_{RACH}^t, 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 [6], the authors designed a dynamic ACB scheme to optimize the problem given in Eq. (1) via adjusting the ACB factor. The ACB factor is adapted based on the knowledge of traffic load, which is estimated via moment matching. The estimated number of RACH attempting IoT devices in the tth TTI $\hat{D}_{\mathrm{RACH}}^t$ is expressed as:

$$\hat{D}_{\mathrm{RACH}}^t = \max\Big\{0, \hat{D}_{\mathrm{RACH}}^{t-1} + \max\big\{-f_{\mathrm{Prea},0}^{t-1}, \hat{D}_{\mathrm{RACH}}^t - \hat{D}_{\mathrm{RACH}}^{t-1}\big\}\Big\} \tag{12}$$

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

$$\hat{D}_{\mathrm{RACH}}^{t-1} = f_{\mathrm{Prea},0}^{t-1}\Big/\Big[\min\Big\{1, p_{\mathrm{ACB}}^{t-1}\big(1 + \frac{(V_{\mathrm{cp},0}^{t-1} - u_{M,p^*})e}{2f_{\mathrm{Prea},0}^{t-1}}\big)\big)^{-1}\Big\}\Big]. \tag{13}$$

In Eq. (13), p_{ACB}^{t-1}, $f_{\mathrm{Prea},0}^{t-1}$, and $V_{\mathrm{cp},0}^{t-1}$ 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 [6, Eq. (32)].

In Eq. (12), $\hat{D}_{\mathrm{RACH}}^t - \hat{D}_{\mathrm{RACH}}^{t-1} \approx \hat{D}_{\mathrm{RACH}}^{t-1} - \hat{D}_{\mathrm{RACH}}^{t-2}$ 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 [6].

This dynamic control approach is designed for an ACB scheme, which is only triggered when the exact traffic load is bigger than the number of preambles (i.e., $D_{\mathrm{RACH}}^t > f_{\mathrm{Prea},0}^t$). Accordingly, the related backlog estimation approach is only used when $D_{\mathrm{RACH}}^t > f_{\mathrm{Prea},0}^t$. However, it cannot estimate the load when $D_{\mathrm{RACH}}^t < f_{\mathrm{Prea},0}^t$, which is required in our problem.

B. Resource Configuration in Single Parameter Single CE Group Scenario

In this subsection, we modify the load estimation approach given in [6] via estimating based on the last number of the collided preambles $V_{\mathrm{cp},0}^{t-1}$ and the previous numbers of idle preambles $V_{\mathrm{ip},0}^{t-1}, V_{\mathrm{ip},0}^{t-2}, \cdots$. And then, we propose an uplink resource configuration approach based on this revised load estimation, namely, LE-URC.

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

$$\mathbb{P}\{d_j = 0|D_{\mathrm{RACH},0}^{t-1} = n\} = \big(1 - \frac{1}{f_{\mathrm{Prea},0}^{t-1}}\big)^n. \tag{14}$$

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

$$\mathbb{E}\{\mathcal{V}_{\mathrm{ip},0}^{t-1}|D_{\mathrm{RACH},0}^{t-1} = n\} = \Big(\sum_{j=1}^{f_{\mathrm{Prea},0}^{t-1}} \mathbb{P}\{d_j = 0|D_{\mathrm{RACH}}^{t-1} = n\}\Big)$$
$$= f_{\mathrm{Prea},0}^{t-1}\big(1 - \frac{1}{f_{\mathrm{Prea},0}^{t-1}}\big)^n. \tag{15}$$

Due to that the actual number of preambles experiencing idles $V_{\mathrm{ip},0}^{t-1}$ 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

$$\zeta^{t-1} = f^{-1}(\mathbb{E}\{V_{\mathrm{ip},0}^{t-1}|D_{\mathrm{RACH},0}^{t-1}) = \log_{\big(\frac{f_{\mathrm{Prea},0}^{t-1} - 1}{f_{\mathrm{Prea},0}^{t-1}}\big)}\big(\frac{V_{\mathrm{ip},0}^{t-1}}{f_{\mathrm{Prea},0}^{t-1}}\big), \tag{16}$$

To obtain the estimated number of RACH attempting IoT devices in the tth TTI $\tilde{D}_{\mathrm{RACH},0}^t$, 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 $\delta^t = \tilde{D}_{\mathrm{RACH},0}^t - \tilde{D}_{\mathrm{RACH},0}^{t-1}$ for $t = 1, 2, \cdots$, and $\tilde{D}_{\mathrm{RACH},0}^0 = 0$. However, $\tilde{D}_{\mathrm{RACH},0}^t$ cannot be obtained before the tth TTI. To solve this, we can assume $\delta^t \approx \delta^{t-1}$ according to [6]. 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 [6]. Therefore, the number of RACH attempting IoT devices in the tth time slot is estimated as

$$\tilde{D}_{\mathrm{RACH},0}^t = \max\big\{2V_{\mathrm{cp},0}^{t-1}, \zeta^{t-1} + \delta^{t-1}\big\}, \tag{17}$$

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

2) Uplink Resource Configuration Based on Load Estimation: In the following, we propose LE-URC by taking into account the resource condition given in Eq. (10). 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

$$\mathbb{P}\{d_j = 1 | \tilde{D}_{\text{RACH},0}^t = n\} = \left(\begin{array}{c} n \\ 1 \end{array}\right) \frac{1}{f_{\text{Prea},0}^t}\left(1 - \frac{1}{f_{\text{Prea},0}^t}\right)^{n-1}. \tag{18}$$

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

$$\mathbb{E}\{\mathcal{V}_{\text{RACH},0}^t | \tilde{D}_{\text{RACH},0}^t = n\} = \sum_{j=1}^{f_{\text{Prea},0}^t} \mathbb{P}\{d_j = 1 | \tilde{D}_{\text{RACH},0}^t = n\}$$
$$= n\left(1 - \frac{1}{f_{\text{Prea},0}^t}\right)^{n-1}, \tag{19}$$

Based on (19), the expected number of IoT devices requesting uplink resource in the tth TTI is derived as

$$\mathbb{E}\{\mathcal{V}_{\text{reqs}}^t | \tilde{D}_{\text{RACH},0}^t = n\} = \mathbb{E}\{\mathcal{V}_{\text{RACH},0}^t | \tilde{D}_{\text{RACH},0}^t = n\} + V_{\text{un},0}^t$$
$$= n\left(1 - \frac{1}{f_{\text{Prea},0}^t}\right)^{n-1} + V_{\text{un},0}^{t-1}, \tag{20}$$

where $V_{\text{un},0}^{t-1}$ is the number of unscheduled IoT devices in the last TTI. Note that $V_{\text{un},0}^{t-1}$ can be observed.

However, if the data resource is not sufficient (i.e., occurs when Eq. (10) 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

$$\mathcal{V}_{\text{up},0}^t = \frac{R_{\text{DATA}}^t}{r_{\text{DATA},i}^t} = \frac{R_{\text{Uplink}} - R_{\text{RACH}}^t}{r_{\text{DATA},i}^t}. \tag{21}$$

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

According to (20) and (21), the expected number of the successfully served IoT devices is given by

$$\mathcal{V}_{suss}^t(f_{\text{Prea},0}^t) = \mathbf{min} \quad \{\mathbb{E}\{\mathcal{V}_{\text{reqs}}^t | \tilde{D}_{\text{RACH},0}^t = n\}, \mathcal{V}_{\text{up},0}^t\}. \tag{22}$$

The maximal expected number of the successfully served IoT devices is obtained by selects the number of preamble $f_{\text{Prea},0}^{t*}$ using

$$f_{\text{Prea},0}^{t*} = \underset{f \in \mathcal{N}_{\text{Prea}}}{\mathbf{argmax}} \quad \mathcal{V}_{suss}^t(f). \tag{23}$$

The LE-URC approach based on the estimated load $\tilde{D}_{\text{RACH},0}^t$ is detailed in **Algorithm 1**. For comparison, we consider an ideal scenario that the actual number of RACH requesting IoT devices D_{RACH}^t is available at the eNB, namely, Full State Information based URC (FSI-URC). FSI-URC configures $f_{\text{Prea},0}^{t*}$ still using the approach given in Eq. (23), while the load estimation approach given in Section III.B.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 $\mathcal{F}_{\text{Prea},0}$, Operation Iteration I.
>
> **1** **for** *Iteration* $\leftarrow 1$ **to** I **do**
> **2**   Initialization of $V_{\text{ip},0}^0 := 12$, $V_{\text{cp},0}^0 := 0$, $\tilde{D}_{\text{RACH},0}^0 := 0$, $\delta^1 := 0$, and bursty traffic arrival rate $\mu_{bursty}^0 = 0$;
> **3**   **for** $t \leftarrow 1$ **to** T **do**
> **4**   Generate μ_{bursty}^t using Eq. (3);
> **5**   The eNB observes $V_{\text{ip},0}^{t-1}$ and $V_{\text{cp},0}^{t-1}$, and calculate ζ^{t-1} using Eq. (16);
> **6**   Estimate the number of RACH requesting IoT devices $\tilde{D}_{\text{RACH},0}^t$ using Eq. (17);
> **7**   Select the number of preambles $f_{\text{Prea},0}^{t*}$ using Eq. (23) based on the estimated load $\tilde{D}_{\text{RACH},0}^t$;
> **8**   The eNB broadcasts $f_{\text{Prea},0}^{t*}$, and backlogged IoT devices attempt communication in the tth TTI;
> **9**   Update $\delta^{t+1} := \tilde{D}_{\text{RACH},0}^t - \tilde{D}_{\text{RACH},0}^{t-1}$.
> **10**   **end**
> **11** **end**

---

3) LE-URC for Multiple CE Groups: We slightly revise the introduced single-parameter single-group LE-URC approach (given in Section III.B) 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. (17). 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 $R_{\text{uplink}} = R_{\text{DATA}}^t + R_{\text{RACH}}^t$. In the multiple CE groups scenarios, the resource R_{DATA}^t are allocated to IoT devices in any CE groups without bias, but R_{RACH}^t is specifically allocated to each CE group.

Under this condition, the expected number of successfully served IoT devices $\mathcal{V}_{\text{suss},i}^t$ given in Eq. (22) needs to be modified by taking into account multiple variables, which extremely complicates the optimization problem. To solve it, we use a sub-optimal solution by artificially setting uplink resource constraint $R_{\text{Uplink},i}$ for each CE group ($R_{\text{Uplink}} = \sum_{i=0}^2 R_{\text{Uplink},i}$). Each CE group can independently allocate the resource between $R_{\text{DATA},i}^t$ and $R_{\text{RACH},i}^t$ according to the approach given in Eq. (23).

V. Q-LEARNING BASED RESOURCE CONFIGURATION IN SINGLE-PARAMETER SINGLE-GROUP SCENARIO

The RL approaches are well-known in addressing dynamic control problem in complex POMDPs [32]. 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-group scenario first, in order to be fairly compared with conventional heuristic approaches. In this section, we consider one single CE group with the *fixed RACH periods* $n_{\text{Rach},0}$ as well as the *fixed repetition value* $n_{\text{Repe},0}$, and only dynamically configuring the number of preambles
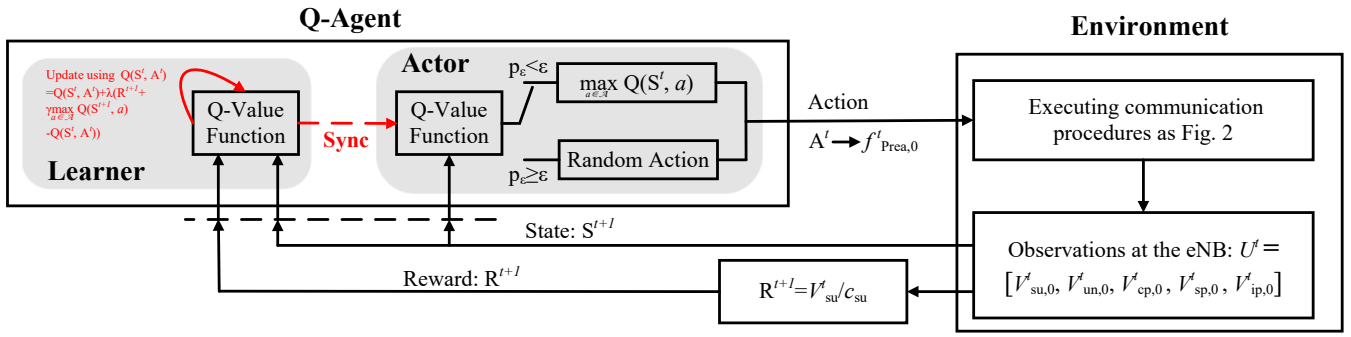
Fig. 3: The Tabular-Q agent and environment interaction in the POMDP.

$f_{\text{Prea},0}$ at the beginning of each TTI. In the following, We first study tabular-Q based on the tabular representation of the value function, which is the simplest Q-learning form with guaranteed convergence [32], but requires extremely long training time. We then study Q-learning with function approximators to improve training efficiency, where LA-Q and DQN will be used to construct an approximation of the desired value function.

A. Q-Learning and Tabular 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 ($O^t = \{U^{t-1}, U^{t-2}, \cdots, U^1\}$) 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 $f_{\text{Prea},0}^t$ in single CE group scenario.

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 $U^{t-1} = [V_{\text{su},0}^{t-1}, V_{\text{un},0}^{t-1}, V_{\text{cp},0}^{t-1}, V_{\text{sp},0}^{t-1}, V_{\text{ip},0}^{t-1}]$. 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 $\mathcal{F}_{\text{Prea}}$. 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 $V_{\text{su}}^t \in O^t$, which is defined as

$$R^{t+1} = V_{\text{su}}^t/c_{\text{su}}, \tag{24}$$

where c_{su} is constant used to normalize the reward function.

Q-learning is a value-based RL approach [32, 33], 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 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 find 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

$$Q(S^t, A^t) = Q(S^t, A^t) + \lambda \big[R^{t+1} + \gamma \max_{a \in \mathcal{A}} Q(S^{t+1}, a) - Q(S^t, A^t) \big], \tag{25}$$

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 $Q(S^t, A^t)$ in Eq. (25) 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}.

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 t, the Q-agent randomly generates a probability p_ϵ^t 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.

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 that a single reward in a specific TTI can be severely biased, because state function is composed of multiple unobserved information with unpredictable distributions (e.g., an action allows for the setting with large number

of preambles f^t_{prea}, but massive random collisions accidentally occur, which leads to an unusual low reward). In the following, the implementation of uplink resource configuration using tabular-Q based real-time optimization is shown in **Algorithm 2**.

Algorithm 2: Tabular-Q Based Uplink Resource Configuration

input : Valid numbers of preambles $\mathcal{F}_{\text{Prea}}$, Operation Iteration I.

1   Algorithm hyperparameters: learning rate $\lambda \in (0, 1]$, discount rate $\gamma \in [0, 1)$, ϵ-greedy rate $\epsilon \in (0, 1]$;
2   Initialization of the Q-value table $Q(s, a)$ with 0 value scalars;
3   **for** *Iteration* $\leftarrow 1$ **to** I **do**
4   Initialization of S^1 by executing a random action A^0 and bursty traffic arrival rate $\mu^0_{bursty} = 0$;
5   **for** $t \leftarrow 1$ **to** T **do**
6   Update μ^t_{bursty} using Eq. (3);
7   **if** $p^t_\epsilon < \epsilon$ **then** select a random action A^t from $\mathcal{A}$;
8   **else** select $A^t = \underset{a \in \mathcal{A}}{\text{argmax}}\, Q(S^t, a)$;
9   The eNB broadcasts $f^t_{\text{Prea}} = \mathcal{F}_{\text{Prea}}(A^t)$ and backlogged IoT devices attempt communication in the tth TTI;
10   The eNB observes S^{t+1}, calculate the related R^{t+1} using Eq. (24), and update $Q(S^t, A^t)$ using Eq. (25).
11   **end**
12 **end**

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 [32, 34–36]. A variety of function approximation approaches can be conducted, such as LA, DNNs, tree search, and which approach to be selected can critically influence the successful learning [32, 35, 36]. 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, due to its simplicity, efficiency, and guaranteed convergence [32, 37, 38]. We then conduct the DNN for Q-learning as a more effective but complicated function approximator, which is also known as DQN [33]. The reasons we conduct DQN are that: 1) the DNN function approximation is able to deal with several kinds of partially observable problems [32, 33]; 2) DQN has the potential to accurately approximate the desired value function while addressing a problem with very large state spaces [33], which can be favored for the learning in the multiple CE group scenarios; 3) DQN 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 [32, Eq. 9.17]) to construct the real-valued feature vector $\mathbf{x}(s)$ due to its efficiency[3]. In the training process, the exploration is the same as the tabular Q-learning by generating random actions, but the exploitation is calculated using the weight matrix $\mathbf{w}$ of the value function. 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:

$$
\begin{aligned}
Q(S^t, a, \mathbf{w}) = \quad & \mathbf{w} \cdot \mathbf{x}(S^t)^T \\
= \quad & \Big[\sum_{j=0}^{|\vec{x}|-1} w_{(0,j)} x_j(S^t), \sum_{j=0}^{|\vec{x}|-1} w_{(1,j)} x_j(S^t), \\
& \cdots, \sum_{j=0}^{|\vec{x}|-1} w_{(|\mathcal{A}|-1,j)} x_j(S^t) \Big]^T .
\end{aligned}
\tag{26}
$$

By searching for the maximal value function scalar in $Q(S^t, a, \mathbf{w})$ given in Eq. (26), 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) [32, 40]. SGD minimizes the error on predictions of observation after each example, where the error is reduced by a small amount following the direction to the optimal target policy $Q^*(s, a)$. As it is infeasible to obtain optimal target policy by summing over all states, we instead estimate the desired action-value function by simply considering one learning sample $Q^*(s, a) \approx Q^*(S^t, a, \mathbf{w}^t)$ [32]. In each TTI, the weigh matrix $\mathbf{w}$ is updated following

$$
\mathbf{w}^{t+1} = \mathbf{w}^t - \lambda \nabla L(\mathbf{w}^t),
\tag{27}
$$

where λ is the learning rate. $\nabla L(\mathbf{w}^t)$ is the gradient of the loss function $L(\mathbf{w}^t)$ used to train the Q-function approximator. This is given as

$$
\begin{aligned}
\nabla L(\mathbf{w}^t) = \big(& R^{t+1} + \gamma \max_a Q(S^{t+1}, a; \mathbf{w}^t) \\
& - Q(S^t, a, \mathbf{w}^t) \cdot \mathbf{x}(A^t, S^t)^T \big) \cdot \nabla_{\mathbf{w}} Q(S^t, A^t, \mathbf{w}^t)
\end{aligned}
\tag{28}
$$

where $\mathbf{w}^t$ is the weight matrix, $\mathbf{x}(A^t, S^t)$ is the features matrix with the same shape of $\mathbf{w}^t$. $\mathbf{x}(A^t, S^t)$ is constructed by zeros and the feature vector located in the row corresponding to

---

[3]The polynomial case is the most well understood feature constructor and always performs well in practice with appropriate setting [32, 34]. Furthermore, the results in [39] 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 DQN.
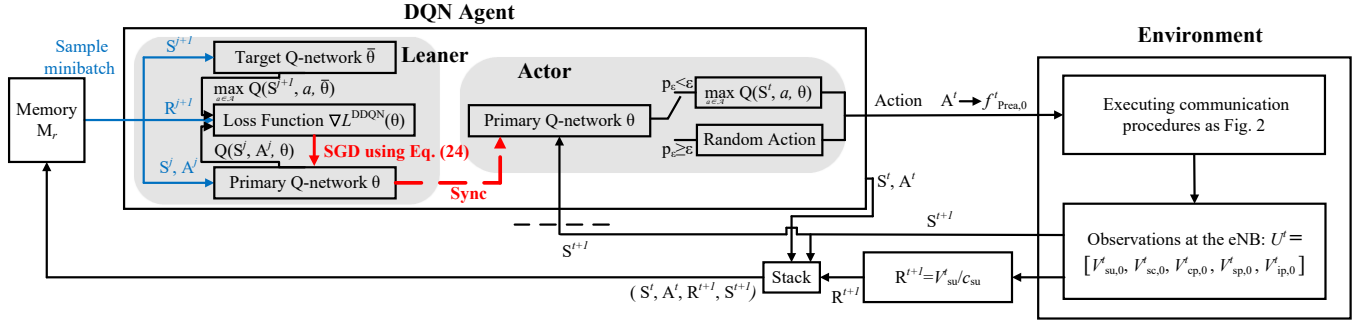
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2019.2904366, IEEE Journal on Selected Areas in Communications

10

Fig. 4: The DQN agent and environment interaction in the POMDP.

the index of the action selected in the 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 matrix $\mathbf{w}$, and updating $Q(s, a; \mathbf{w})$ with SGD given in (28) in step 10 of **Algorithm 2**.

2) Deep Q-Network: The DQN agent parameterizes the action-state value function $Q(s, a)$ by using a function $Q(s, a; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ represents the weights matrix of a DNN with multiple layers. We consider the conventional DNN, where neurons between two adjacent layers are fully pairwise connected, namely fully-connected layers. The input of the DNN is given by the variables in state S^t; the intermediate hidden layers are Rectifier Linear Units (ReLUs) by using the function $f(x) = \max(0, x)$; while the output layer is composed of linear units[4], which are in one-to-one correspondence with all available actions in $\mathcal{A}$.

The exploitation is obtained by performing forward propagation of Q-function $Q(s, a; \boldsymbol{\theta})$ with respect to the observed state S^t. The weights matrix $\boldsymbol{\theta}$ is updated online along each training episode by using double deep Q-learning (DDQN) [41], which to some extent reduce the substantial overestimations[5] of value function. Accordingly, learning takes place over multiple training episodes, with each episode of duration N_{TTI} TTI periods. In each TTI, the parameter $\boldsymbol{\theta}$ of the Q-function approximator $Q(s, a; \boldsymbol{\theta})$ is updated using RMSProp optimizer [42] as

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \lambda_{\text{RMS}} \nabla L^{\text{DDQN}}(\boldsymbol{\theta}^t), \qquad (29)$$

where λ_{RMS} is RMSProp learning rate [42], $\nabla L(\theta)$ is the gradient of the loss function $L(\theta^t)$ used to train the Q-function approximator. This is given as

$$\nabla L^{\text{DDQN}}(\theta^t) = \mathbb{E}_{S^i, A^i, R^{i+1}, S^{i+1}} \Big[\big(R^{i+1} + \gamma \max_a Q(S^{i+1}, a; \bar{\boldsymbol{\theta}}^t) \\ - Q(S^i, A^i; \boldsymbol{\theta}^t) \big) \nabla_{\boldsymbol{\theta}} Q(S^i, A^i; \boldsymbol{\theta}^t) \Big], \qquad (30)$$

where the expectation is taken with respect to a so-called minibatch, which are randomly selected previous samples

---

[4]Linear activation is used here according to [33]. Note that Q-learning is value-based, thus 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.

[5]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 efficiency of the algorithm [35, 41].

Algorithm 3: DQN Based Uplink Resource Configuration

input : The set of numbers of preambles in each RACH period $\mathcal{F}_{\text{Prea}}$, and operation iteration I.

1 Algorithm hyperparameters: learning rate $\lambda_{\text{RMS}} \in (0, 1]$, discount rate $\gamma \in [0, 1)$, ϵ-greedy rate $\epsilon \in (0, 1]$, target network update frequency K;

2 Initialization of replay memory M to capacity C, the primary Q-network $\boldsymbol{\theta}$, and the target Q-network $\bar{\boldsymbol{\theta}}$;

3 **for** *Iteration* $\leftarrow 1$ **to** I **do**

4   Initialization of S^1 by executing a random action A^0 and bursty traffic arrival rate $\mu_{bursty}^0 = 0$;

5   **for** $t \leftarrow 1$ **to** T **do**

6   Update μ_{bursty}^t using Eq. (3);

7   **if** $p_\epsilon < \epsilon$ **then** select a random action A^t from $\mathcal{A}$;

8   **else** select $A^t = \underset{a \in \mathcal{A}}{\text{argmax}}\, Q(S^t, a, \boldsymbol{\theta})$;

9   The eNB broadcasts $\mathcal{F}_{\text{Prea}}(A^t)$ and backlogged IoT devices attempt communication in the tth TTI;

10   The eNB observes S^{t+1}, and calculate the related R^{t+1} using Eq. (24);

11   Store transition $(S^t, A^t, R^{t+1}, S^{t+1})$ in replay memory M;

12   Sample random minibatch of transitions $(S^j, A^j, R^{j+1}, S^{j+1})$ from replay memory M;

13   Perform a gradient descent for $Q(s, a; \boldsymbol{\theta})$ using Eq. (30);

14   Every K steps update target Q-network $\bar{\boldsymbol{\theta}} = \boldsymbol{\theta}$.

15   **end**

16 **end**

$(S^i, A^i, S^{i+1}, R^{i+1})$ for some $i \in \{t - M_r, ..., t\}$, with M_r being the replay memory [33]. When $t - M_r$ is negative, this is interpreted as including samples from the previous episode. The use of minibatch, instead of a single sample, to update the value function $Q(s, a; \boldsymbol{\theta})$ improves the convergent reliability of value function [33]. Furthermore, following DDQN [41], in (30), $\bar{\boldsymbol{\theta}}^t$ is a so-called target Q-network that is used to estimate the future value of the Q-function in the update rule. This parameter is periodically copied from the current value $\boldsymbol{\theta}^t$ and kept fixed for a number of episodes [41].

VI. Q-LEARNING BASED RESOURCE CONFIGURATION IN MULTI-PARAMETER MULTI-GROUP SCENARIO

Practically, NB-IoT is always deployed with multiple CE groups to serve IoT devices with various coverage requirements. In this section, we study the problem (1) of optimizing the resource configuration for three CE groups each with parameters $A^t = \{n_{\text{Rach},i}^t, f_{\text{Prea},i}^t, n_{\text{Repe},i}^t\}_{i=0}^2$. This joint optimization by configuring each parameter in each CE group can

improve the overall data access and transmission performance. Note that each CE group shares the uplink resource in the same bandwidth, and the eNB schedules data resource to all RRC connected IoT devices without the CE group bias as introduced in Sec. III.B.4. 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.

The Q-learning algorithms with the single CE group provided in Sec. V are model-free, and thus their learning structure can be directly used in this multi-parameter multi-group scenario. However, considering multiple CE groups results in the increment of observations space, which exponentially increases the size of state space. To train Q-agent with this expansion, the requirements of time and computational resource greatly increase. In such case, the tabular-Q would be extremely inefficient, as not only the state-action value table requires a big memory, but it is impossible to repeatedly experience every state to achieve convergence with limited time. In view of this, we only study Q-learning with value function approximation (LA-Q and DQN) to design uplink resource configuration approaches for the multi-parameter multi-group scenario.

LA-Q and DQN 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 includes information about the last M_o TTIs $(U^{t-1}, U^{t-2}, U^{t-3}, \cdots, U^{t-M_o})$. This design improves Q-agent by enabling it to estimate the trend of traffic. 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 $V_{\mathrm{su},i}$ of each CE group, which is expressed as

$$R^{t+1} = \Big(\sum_{i=0}^{2} V_{\mathrm{su},i}^t \Big) / c_{\mathrm{su}}. \qquad (31)$$

Same as the state spaces, the available action spaces also exponentially increases with the increment of the adjustable configurations. The number of available actions corresponds to the possible combinations of configurations $|\mathcal{A}| = \prod_{i=0}^{2} (|\mathcal{N}_{\mathrm{Rach},i}| \times |\mathcal{N}_{\mathrm{Repe},i}| \times |\mathcal{F}_{\mathrm{Prea},i}|)$ (i.e., $|\cdot|$ denotes the number of elements in any vector $\cdot$, $\mathcal{A}$ is the set of actions, $\mathcal{N}_{\mathrm{Rach},i}$, $\mathcal{N}_{\mathrm{Repe},i}$, and $\mathcal{F}_{\mathrm{Prea},i}$ are the sets of the number of RACH periods, the repetition value, and the number of preambles in each RACH period). Unfortunately, it is extremely hard to optimize the system under such numerous action spaces (i.e., $|\mathcal{A}|$ can be over fifty thousands.), 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. To solve this problem, we then provide two approaches that can reduce the dimension of action space to enable the LA and DQN in the multi-parameter multi-group scenario.

A. Actions Aggregated Approach

We first provide AA based Q-learning approaches, which guarantee convergent capability by sacrificing the accuracy of action selection[6]. In detail, the specific action selection can be converted to the increasing or decreasing trend selection. Instead of selecting the exact values from the sets of $\mathcal{N}_{\mathrm{Rach},i}$, $\mathcal{N}_{\mathrm{Repe},i}$, and $\mathcal{F}_{\mathrm{Prea},i}$, we convert it to single step ascent/descent based on the last action, which is represented by $A_{\mathrm{Rach},i}^t \in \{0, 1\}$, $A_{\mathrm{Repe},i}^t \in \{0, 1\}$, and $A_{\mathrm{Prea},i}^t \in \{0, 1\}$ for the number of RACH periods $n_{\mathrm{Rach},i}^t$, the repetition values $n_{\mathrm{Repe},i}^t$, and the number of preambles in each RACH period $f_{\mathrm{Prea},i}^t$ in the tth TTI. Consequently, the size of total action spaces for the three CE groups is reduced to $|\mathcal{A}|=2^9=512$. 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.

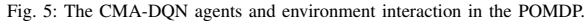### B. Cooperative Multi-agent Learning Approach

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 sufficient to deal with the problem with unsolvable action spaces [43]. As shown in Fig. 5, we consider multiple DQN agents are centralized at the eNB with the same structure of value function approximator[7] following Section IV.B.2). We break down the action space by considering nine separate action variables in A^t, where each DQN agent controls their own action variable as shown in Fig. 5. Recall that we have three variables for each group i, namely $n_{\mathrm{Rach},i}$, $n_{\mathrm{Repe},i}$, and $f_{\mathrm{Prea},i}$.

We introduce a separate DQN agent for each output variable in A^t defined as action A_k^t selected by the kth agent, where each kth agent is responsible to update the value $Q(S^t, A_k^t; \boldsymbol{\theta}_k)$ of action A_k^t in shared state S^t. The DQN agents are trained in parallel and receive the same reward signal given in Eq. (31) at the end of each TTI as per problem (1). The use of this common reward signal ensures that all DQN agents aim at cooperatively increase the objective in (1). Note that the approach can be interpreted as applying a factorization of the overall value function akin to the approach proposed in [44] for multi-agent systems.

The challenge of this approach is how to evaluate each action according to the common reward function. For each DQN 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 DQN agents. For instance, a positive action can receive a mismatched low reward due to other DQN agents' negative actions. Fortunately, in our scenario, all DQN agents

---

[6]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 [32].

[7]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.

Fig. 5: The CMA-DQN agents and environment interaction in the POMDP.

are centralized at the eNB, which means that all DQN agents can have full information among each other. Accordingly, we adopt the action selection histories of each DQN agent as part of state function[8], thus they are able to know how reward is influenced by different combinations of actions. To do so, we define state variable S^t as

$$S^t = [A^{t-1}, U^{t-1}, A^{t-2}, U^{t-2}, \cdots, A^{t-M_o}, U^{t-M_o}], \quad (32)$$

where M_o is the number of stored observations, A^{t-1} is the set of selected action of each DQN agent in the $(t-1)$th TTI corresponding to $n_{\text{Rach},i}$, $n_{\text{Repe},i}$, and $f_{\text{Prea},i}$ for the ith CE group, and U^{t-1} is the set of observed transmission receptions.

In each TTI, the parameters θ_k of the Q-function approximator $Q(S^t, A^t_k; \theta_k)$ are updated using RMSProp optimizer at all agents k as Eq. (29). The learning algorithm can be implemented following **Algorithm 3**. Different from the single-parameter single-group scenario, we need to first initialize nine primary networks θ_k, target networks $\bar{\theta}_k$, and replay memories M_k for each DQN agent. In step 11 of **Algorithm 3**, the current transactions of each DQN 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 DQN agent.

VII. SIMULATION RESULTS

In this section, we evaluate the performance of the proposed Q-learning approaches and compare them with the conventional LE-URC and FSI-URC described in Sec. IV via numerical experiments. We adopt the standard network parameters listed in Table I following [2, 4, 23, 26, 30], and hyperparameters for Q-learning listed in Table II. Accordingly, one epoch consists of 937 TTIs (i.e., 10 minutes). The RL agents will first be trained in a so-called learning phase, and after convergence, their performance will be compared with LE-URC and FSI-URC in a so-called testing phase. All testing performance results are obtained by averaging over 1000 episodes. In the following, we present our simulation results of the single-parameter single-group scenario and the multi-parameter multi-group scenario in Section VII-A and Section VII-B, respectively.

[8]The state function can be designed to collect more information according to the complexity requirements, such as sharing the value function between each DQN agent [43].

TABLE I: Simulation Parameters

Parameters	Value
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_{RACHmax}	23 dBm
The received SNR threshold γ_{th}	0 dB
Duration of periodic traffic T_{periodic}	1 hour
Period of TTI T_{TTI}	640ms
Duration of bursty traffic T_{bursty}	10 minutes
Set of number of preambles $\mathcal{F}_{\text{Prea}}$	$\{12, 24, 36, 48\}$
Set of repetition value $\mathcal{N}_{\text{Repe}}$	$\{1, 2, 4, 8, 16, 32\}$
Set of number of RACH periods $\mathcal{N}_{\text{Rach}}$	$\{1, 2, 4\}$
Maximum RACH attempts γ_{pMax}	10
Maximum allowed resource requests γ_{RRC}	5
Maximum allowed RACH in one CE $\gamma_{\text{pCE},i}$	5
REs required for B_{RACH}	4
REs required for B_{DATA}	32
Bursty traffic parameter Beta(α, β)	(3,4)

TABLE II: Q-learning Hyperparameters

Hyperparameters	Value
Learning rate λ for Tabular-Q and LA-Q	0.01
Learning rate by RMSProp λ_{RMS} for DQN	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

A. Single-Parameter Single-Group Scenario

In the single-parameter single-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 $D_{\text{periodic}} = 10000$, and the number of bursty IoT devices to be $D_{\text{bursty}} = 5000$. The DQN is set with three hidden layers, each with 128 ReLU units. Tabular-Q, LA-Q, and DQN approaches are proposed in Sec. V.A, V.B.1, and V.B.2, respectively. The conventional LE-URC and FSI-URC approaches are proposed in Sec. IV.B.

Throughout epoch, each device has a periodical traffic profile (i.e., Uniform distribuion given in Eq. (2)), or a bursty traffic

Fig. 6: The real-time traffic load and V_{su} for FSI-URC, LE-URC, and DQN.

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

profile (i.e., the time limited Beta profile defined in Eq. (4) with parameters $(3, 4)$) that has a peak around the 400th TTI. The resulting average number of newly generated packets is shown as dashed line in Fig. 6(a). Fig. 6(b) plot the number of successfully served IoT devices V_{su} with the proposed FSI-URC, LE-URC, and DQN approaches. In Fig. 6(b), V_{su} first increases gradually with the increasing of traffic shown in Fig. 6(a), until it reaches the serving capacity of eNB. Then, V_{su} decreases slowly due to the increasing collisions and scheduling failures with the increase of traffic. After that, V_{su} increases gradually as the collisions and scheduling failures decrease with the decreasing of traffic. Finally, V_{su} 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 V_{su}^t at one time instance as Eq. (11). DQN 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_{su}^t at one time instance, whereas DQN 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. (22), while the optimization over long-term period (DQN) 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 DQN approach can well capture these hidden features to optimize the long-term performance of V_{su} as Eq. (1).

Fig. 7(a) compares the number of successfully served IoT devices V_{su} under Tabular-Q, LA-Q, and DQN approaches. We observe that all these three approaches achieve similar values of V_{su}, which indicates that both LA-Q and DQN can well estimate the optimal value function $Q^*(s, a)$ as the converged Tabular-Q in this low-complexity single CE group scenario. Fig. 7(b) plots the average received reward over each bursty
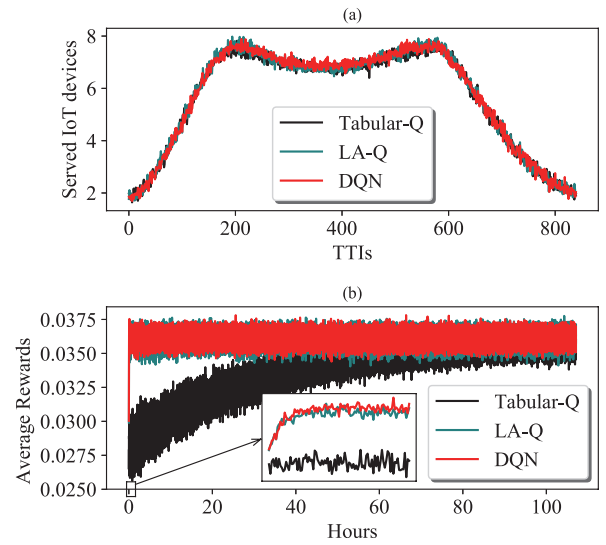
duration $\mathbb{E}\{R\} = \frac{1}{T_{bursty}} \sum_{t=0}^{T_{bursty}} R^t$ (i.e., one epoch consists of one bursty duration T_{bursty}) from the beginning of the training versus the required training time. It can be seen that LA-Q and DQN converge to the optimal value function $Q^*(s, a)$ (about 10 minutes) much faster than that of Tabular-Q (about 5 days). The observations in Fig. 7 demonstrate that LA-Q and DQN can be good alternatives for tabular-Q to achieve almost the same number of served IoT devices with much less training time.
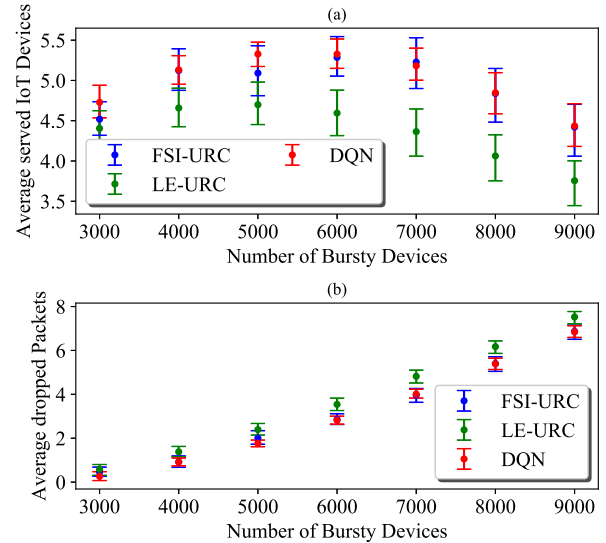

Fig. 8: $\mathbb{E}\{V_{su}\}$ and $\mathbb{E}\{V_{drop}\}$ for FSI-URC, LE-URC, and DQN.

Fig. 8(a) and Fig. 8(b) plot the average number of successfully served IoT devices $\mathbb{E}\{V_{su}\}$ and the average number of dropped packets $\mathbb{E}\{V_{drop}\}$ (i.e., this system performance can only be summarized in simulation) over a bursty duration T_{bursty} versus the number of bursty IoT devices D_{bursty}. In Fig. 8(a), we observe that $\mathbb{E}\{V_{su}\}$ 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 DQN always outperforms LE-URC (especially for relatively large D_{bursty}), which indicates the superiority of DQN approach in handling massive bursty IoT devices. Interestingly, DQN provides better performance of the number of served IoT devices and smaller mean errors than the ideal FSI-URC approach in most cases, which results from the long-term optimization capability of DQN.

B. Multi-Parameter Multi-Group Scenario

Considering eNB is located at the center of a circle area with 12 km radius, we set 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 constraints for LE-URC following $R_{\text{uplink},0} : R_{\text{uplink},1} : R_{\text{uplink},2} = 1 : 1 : 1$. To model massive IoT traffic, both the number of periodical IoT devices D_{periodic} and the number of bursty IoT devices D_{bursty} increase to 30000. In AA-DQN, we use one Q-network with three hidden layers each of which is consist of 2048 ReLU units. In CMA-DQN, nine DQNs are used to control each of the nine configuration (i.e., $n_{\text{Rach},i}$, $n_{\text{Repe},i}$, $f_{\text{Prea},i}$ for three CE groups), where each DQN has three hidden layers, each with 128 ReLU units. AA-LA-Q and AA-DQN approaches are proposed in Sec. VI.A, and CMA-DQN approach is proposed in Sec. VI.B.
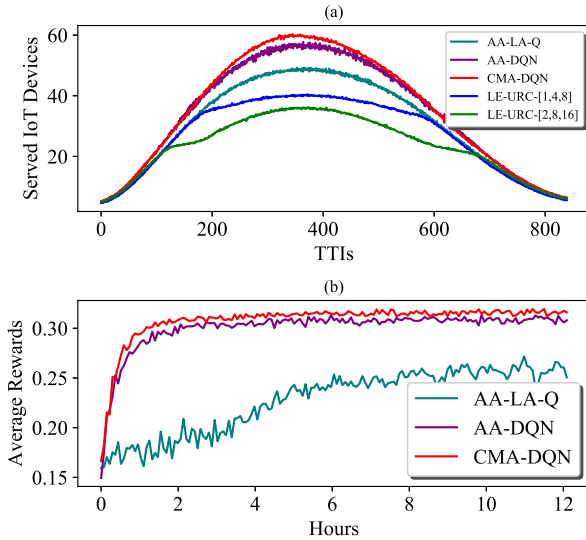
Fig. 9: V_{su} and the average received reward.

Fig. 9(a) compares the number of successfully served IoT devices V_{su} during one epoch using AA-LA-Q, AA-DQN, CMA-DQN and LE-URC. The "LE-URC-[1,4,8]" and "LE-URC-[2,8,16]" curves represent the LE-URC approach with the repetition values $\{n_{\text{Repe},0}, n_{\text{Repe},1}, n_{\text{Repe},2}\}$ set to $\{1, 4, 8\}$ and $\{2, 8, 16\}$, respectively. We observe that the number of successfully served IoT devices V_{su} follows CMA-DQN > AA-DQN > AA-LA-Q $\gg$

LE-URC-[1,4,8] $\gg$ LE-URC-[2,8,16]. As can be seen, all Q-learning based approaches outperform LE-URC approaches, due to that these Q-learning based approaches can dynamically optimize the number of served IoT devices by accurately configuring each parameter. We also observe that CMA-DQN slightly outperforms the others in the light traffic regions at the beginning and end of the epoch, but it substantially outperforms the others in the period of heavy traffic in the middle of the epoch. This demonstrates the capability of CMA-DQN in better managing the scarce channel resource in the presence of heavy traffic. It is also observed that increasing the repetition value of each CE group with LE-URC improves the received SNR, and thus the RACH success rate in the light traffic region, but it degrades the scheduling success rate due to limited channel resource in the heavy traffic region.

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 CMA-DQN and AA-DQN outperform AA-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. We also observe that CMA-DQN achieves higher $\mathbb{E}\{R\}^*$ than AA-DQN, due to that CMA-DQN can accurately select the exact values from the set of actions $\{\mathcal{N}_{\text{Repe}}, \mathcal{N}_{\text{Rach}}, \mathcal{F}_{\text{Prea}}\}$, whereas AA-DQN can only select ascent/descent actions, which leads to a sub-optimal solution.

To gain more insight into the operation of CMA-DQN, Fig. 10 plots the average number of successfully served IoT devices $V_{\text{succ},i}$ for each CE group i, and Fig. 11 plots the average number $n_{\text{Repe},i}^t$ of repetitions and the average number of Random Access Opportunities (RAOs), defined as the product $n_{\text{Rach},i}^t \times f_{\text{Prea},i}^t$, for each CE group i that are selected by CMA-DQN over the testing episodes. As seen in Fig. 10, CMA-DQN substantially outperforms LE-URC approaches for each CE group i, where the reasons for this performance are showcased in Fig. 11. As seen in Fig. 11(a)-(c), CMA-DQN increases the number of repetitions in the light traffic region in order to improve the SNR and reduce RACH failures, while decreasing it in the heavy traffic region so as to reduce scheduling failures. Surprisingly, the CMA-DQN 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 CMA-DQN is aware of the key to optimize the overall performance V_{su} 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. As illustrated in Fig. 11(d)-(f), this allows CMA-DQN to increase the number of RAOs in the high traffic regime mitigating the impact of collisions on the throughput. In contrast, for the CE groups 1 and 2, in the heavy traffic region, LE-URC decreases the number of RAOs in order to reduce resource scheduling failures, causing an overall lower throughput as seen in Fig. 10.
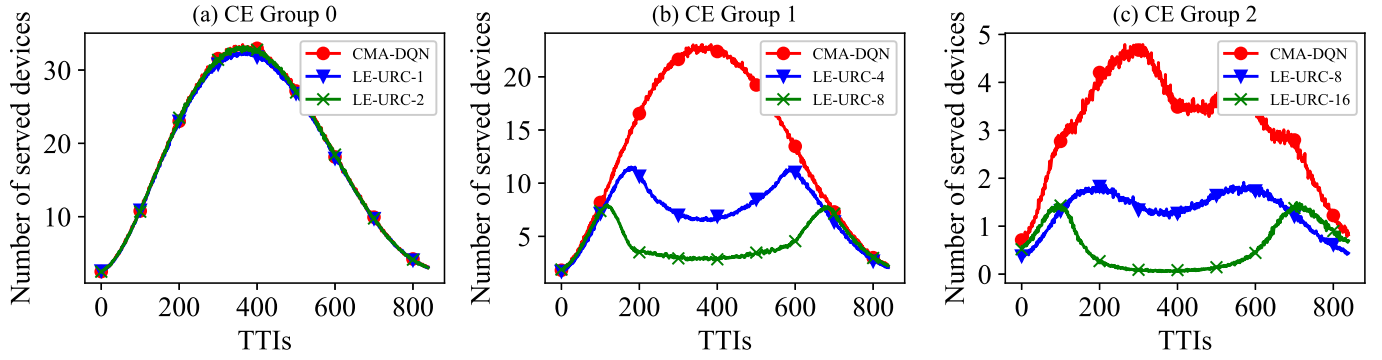
Fig. 10: The average number of successfully served IoT devices $V_{\text{succ},i}$ for each CE group i.
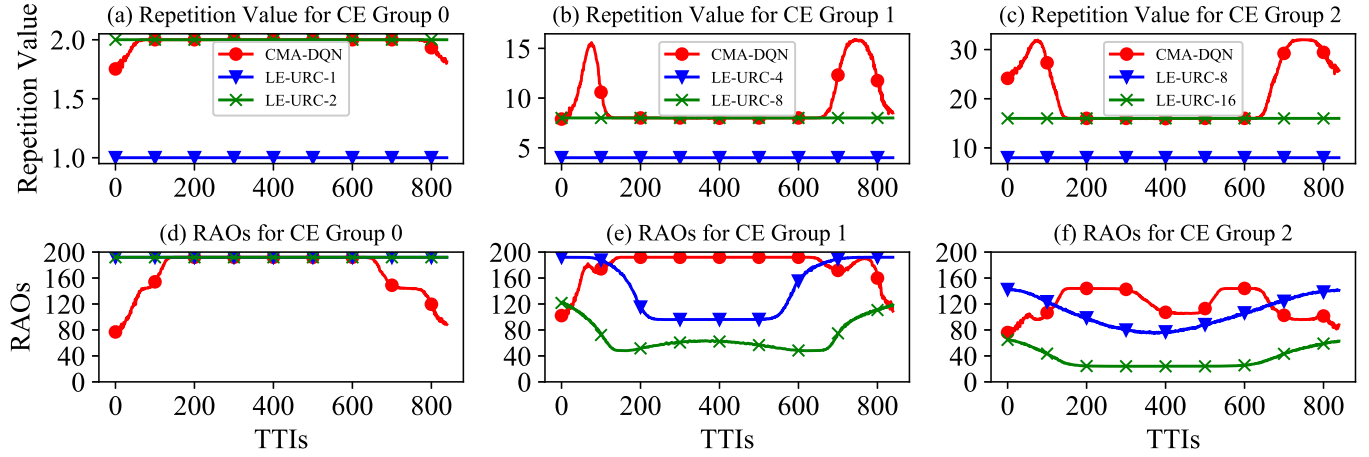

Fig. 11: The allocated repetition value $n_{\text{Repe},i}^{t}$, and RAOs produced by $n_{\text{Rach},i}^{t} \times f_{\text{Prea},i}^{t}$.

The realistic network conditions can be different from the simulation environment, due to that the practical traffic and physical channel vary and can be unpredictable. This difference may lead to inaccurate configuration that can degrade the system performance of each approach. Fortunately, the proposed RL-based approaches can self-update after deployment according to the practical observation in NB-IoT networks in an online manner. To model this, we use the trained CMA-DQN agents given in Fig. 11 (i.e., the bursty traffic is modelled by the time limited Beta profile with parameters $(3, 4)$), and test them in a slightly modified traffic scenario that the bursty traffic is with $\text{Beta}(5, 6)$, and we set the constant exploration rate $\epsilon = 0.001$. Fig. 12 plots the average number of successfully served IoT devices $\mathbb{E}\{V_{\text{su}}\}$ per TTI over each episode versus epochs. Our result shows that, as expected, $\mathbb{E}\{V_{\text{su}}\}$ follows CMA-DQN>LE-URC-[1,4,8]>LE-URC-[2,8,16] at any epoch. More importantly, the performance of CMA-DQN gradually improves along epochs, which sheds light on the online self-updating capability of the proposed RL-based approaches.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we developed Q-learning based uplink resource configuration approaches to optimize the number of served IoT
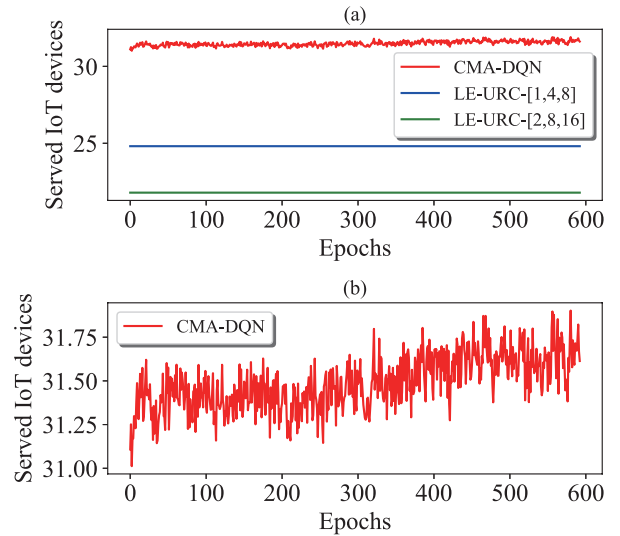

Fig. 12: The average number of successfully served IoT devices per TTI over each epoch in online updating

devices in real-time in NB-IoT networks. We first developed tabular-Q, LA-Q, and DQN based approaches for the single-parameter single-group scenario, which are shown to outperform the conventional LE-URC and FSI-URC approaches in terms of the number of served IoT devices. Our results

demonstrated that LA-Q and DQN can be good alternatives for tabular-Q to achieve almost the same system performance with much less training time. To support traffic with different coverage requirements, we then studied the multi-parameter multi-group scenario as defined in NB-IoT standard, which introduced the high-dimensional configurations problem. To solve it, we advanced the proposed LA-Q and DQN using the Actions Aggregation technique (AA-LA-Q and AA-DQN), which guarantees the convergent capability of Q-learning by sacrificing the accuracy in resource configuration. We further developed CMA-DQN by dividing high-dimensional configurations into multiple parallel sub-tasks, which achieved the best performance in terms of the number of successfully served IoT devices V_{su} with the least training time.

From the results using the simulated traffics in our work, we demonstrated that the proposed RL-based resource configuration approaches significantly outperform the conventional approaches in terms of throughput. With realistic traffic, a direct implementation of DRL may bring computational complexity and processing delay at the NB-IoT BSs, so how to reduce the complexity of DRL algorithms can be considered in future work. Furthermore, online self-updating of each RL-based approach can be deployed either in BSs, network edges, or cloud, where their efficiency could be further evaluated. From the perspective of performance improvement, a promising future direction is to cooperatively optimize network throughput along with the IoT devices' key performance indicators (KPIs), such as power consumption and transmission delay. Such multi-objective optimization is quite challenging and should be addressed in the future.

REFERENCES

[1] N. Jiang, Y. Deng, O. Simeone, and A. Nallanathan, "Cooperative deep reinforcement learning for multiple-group NB-IoT networks optimization," *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP) invited paper*, 2019.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] J. Moon and Y. Lim, "A reinforcement learning approach to access management in wireless cellular networks," *Wireless Commun. Mobile Comput.*, 2017.

[13] 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.

[14] 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.

[15] 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.

[16] 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, Jun. 2018.

[17] 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, 5 2017.

[18] A. Azari, G. Miao, C. Stefanovic, and P. Popovski, "Latency-energy tradeoff based on channel scheduling and repetitions in NB-IoT systems," *arXiv preprint arXiv:1807.05602*, Jul. 2018.

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

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

[21] 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.

[22] N. Jiang, Y. Deng, A. Nallanathan, X. Kang, and T. Q. S. Quek, "Analyzing random access collisions in massive IoT networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 10, pp. 6853–6870, Oct. 2018.

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

[24] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, and J. Wang, "A first look at cellular machine-to-machine traffic: large scale measurement and characterization," *ACM SIGMETRICS Performance Evaluation Rev.*, vol. 40, no. 1, pp. 65–76, Jun. 2012.

[25] J. Kim, J. Lee, J. Kim, and J. Yun, "M2M service platforms: Survey, issues, and enabling technologies." *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 61–76, Jan. 2014.

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

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

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

[29] 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.

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

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

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

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

[34] 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.

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

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

[37] 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, Dec. 2013.

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

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

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

[41] 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.

[42] 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, Oct. 2012.

[43] 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)*, Mar. 2008.

[44] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Pro. Int. Conf. Auton. Agents MultiAgent Syst. (AAMAS)*, Jul. 2018, pp. 2085–2087.
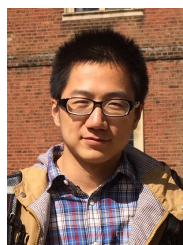
Nan Jiang (S'16) is currently working toward the Ph.D. degree in electronic engineering at Queen Mary University of London, London, U.K.

His research interests include internet of things, machine learning, and radio resource management.

Yansha Deng (S'13-M'18) is currently a Lecturer (Assistant Professor) in department of Informatics, Kings College London, U. K. She received the Ph.D. degree in Electrical Engineering from the Queen Mary University of London, UK in 2015. From 2015 to 2017, she was a Post-Doctoral Research Fellow with Kings College London, UK. Her research interests include molecular communication, internet of things, and 5G wireless networks. She was a recipient of the Best Paper Awards from ICC 2016 and Globecom 2017 as the first author. She is currently an Editor of IEEE Transactions on Communications and IEEE Communication Letters. She also received Exemplary Reviewer for the IEEE Transactions on Communications in year 2016 and 2017. She has also served as TPC member for many IEEE conferences, such as IEEE GLOBECOM and ICC.

Arumugam Nallanathan (S'97-M'00-SM'05-F'17) is Professor of Wireless Communications and Head of the Communication Systems Research (CSR) group in the School of Electronic Engineering and Computer Science at Queen Mary University of London since September 2017. He was with the Department of Informatics at King's College London from December 2007 to August 2017, where he was Professor of Wireless Communications from April 2013 to August 2017 and a Visiting Professor from September 2017. He was an Assistant Professor in the Department of Electrical and Computer Engineering, National University of Singapore from August 2000 to December 2007. His research interests include 5G Wireless Networks, Internet of Things (IoT) and Molecular Communications. He published nearly 400 technical papers in scientific journals and international conferences. He is a co-recipient of the Best Paper Awards presented at the IEEE International Conference on Communications 2016 (ICC'2016) , IEEE Global Communications Conference 2017 (GLOBE-COM'2017) and IEEE Vehicular Technology Conference 2018 (VTC'2018). He is an IEEE Distinguished Lecturer. He has been selected as a Web of Science Highly Cited Researcher in 2016. He published nearly 400 technical papers in scientific journals and international conferences. He is a co-recipient of the Best Paper Awards presented at the IEEE International Conference on Communications 2016 (ICC'2016), IEEE Global Communications Conference 2017 (GLOBECOM'2017) and IEEE Vehicular Technology Conference 2018 (VTC'2018). He is an IEEE Distinguished Lecturer. He has been selected as a Web of Science Highly Cited Researcher in 2016.

He is an Editor for IEEE Transactions on Communications. He was an Editor for IEEE Transactions on Wireless Communications (2006-2011), IEEE Transactions on Vehicular Technology (2006-2017), IEEE Wireless Communications Letters and IEEE Signal Processing Letters. He served as the Chair for the Signal Processing and Communication Electronics Technical Committee of IEEE Communications Society and Technical Program Chair and member of Technical Program Committees in numerous IEEE conferences. He received the IEEE Communications Society SPCE outstanding service award 2012 and IEEE Communications Society RCC outstanding service award 2014.

Jonathon A. Chambers (S'83-M'90-SM'98-F'11) received the Ph.D. and D.Sc. degrees in signal processing from the Imperial College of Science, Technology and Medicine (Imperial College London), London, U.K., in 1990 and 2014, respectively. From 1991 to 1994, he was a Research Scientist with the Schlumberger Cambridge Research Center, Cambridge, U.K. In 1994, he returned to Imperial College London as a Lecturer in signal processing and was promoted to Reader (Associate Professor) in 1998. From 2001 to 2004, he was the Director of the Center for Digital Signal Processing and a Professor of signal processing with the Division of Engineering, King's College London. From 2004 to 2007, he was a Cardiff Professorial Research Fellow with the School of Engineering, Cardiff University, Cardiff, U.K. Between 2007-2014, he led the Advanced Signal Processing Group, within the School of Electronic, Electrical and Systems Engineering at Loughborough University and is now a Visiting Professor. In 2015, he joined the School of Electrical and Electronic Engineering, Newcastle University, where he was Professor of signal and information processing and led the ComS2IP group and is now a Visiting Professor. In 2017 he became the Head of the Department of Engineering at the University of Leicester. He is also an International Honorary Dean and Guest Professor at Harbin Engineering University, China with support from the 1000 Talents Scheme. He is co-author of the books Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability (New York, NY, USA:Wiley, 2001) and EEG Signal Processing (New York, NY, USA: Wiley, 2007). He has advised more than 80 researchers through to Ph.D. graduation and published more than 500 conference papers and journal articles, many of which are in IEEE journals. His research interests include adaptive and blind signal processing and their applications.

Dr. Chambers is a Fellow of the Royal Academy of Engineering, U.K., and the Institution of Electrical Engineers. He was the Technical Program Chair of the 15th International Conference on Digital Signal Processing and the 2009 IEEE Workshop on Statistical Signal Processing, both held in Cardiff, U.K., and a Technical Program Cochair for the 36th IEEE International Conference on Acoustics, Speech, and Signal Processing, Prague, Czech Republic. He received the first QinetiQ Visiting Fellowship in 2007 for his outstanding contributions to adaptive signal processing and his contributions to QinetiQ, as a result of his successful industrial collaboration with the international defense systems company QinetiQ. He has served on the IEEE Signal Processing Theory and Methods Technical Committee for six years and the IEEE Signal Processing Society Awards Board for three years. He is currently a member of the IEEE Signal Processing Conference Board and the European Signal Processing Society Best Paper Awards Selection Panel. He has also served as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING for two terms over the periods 1997-1999, 2004-2007, and as a Senior Area Editor between 2011-2014.