COOPERATIVE DEEP REINFORCEMENT LEARNING FOR MULTIPLE-GROUP NB-IOT NETWORKS OPTIMIZATION

Nan Jiang[1] Yansha Deng[2] Osvaldo Simeone[2] Arumugam Nallanathan[1]

[1]Queen Mary University of London, School of Electronic Engineering and Computer Science, Mile End Road, London E1 4NS, UK

[2]King's College London, Department of Informatics, Strand, London WC2R 2LS, UK

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 resources allocated to each group of devices for random access and for data transmission. Assuming no knowledge of the traffic statistics, the problem is to determine, in an online fashion at each Transmission Time Interval (TTI), the configurations that maximizes the long-term average number of IoT devices that are able to both access and deliver data. Given the complexity of optimal algorithms, a Cooperative Multi-Agent Deep Neural Network based Q-learning (CMA-DQN) approach is developed, whereby each DQN agent independently control a configuration variable for each group. The DQN agents are cooperatively trained in the same environment based on feedback regarding transmission outcomes. CMA-DQN is seen to considerably outperform conventional heuristic approaches based on load estimation.

Index Terms— Deep Reinforcement Learning, Multi-Agent, NB-IoT, Resource Configuration, Random Access

1. INTRODUCTION

To effectively support the emerging massive Internet of Things (IoT) ecosystem, the 3GPP has standardized NarrowBand-IoT (NB-IoT), a new radio access technology designed to coexist with Long-Term Evolution [1]. NB-IoT supports up to three groups of IoT devices, known as Coverage Enhancement (CE) groups. Each group shares a similar average received Signal-to-Noise Ratio (SNR), as measured based on a broadcast signal, and traffic characteristics (see Fig.1(a)) [2]. At the beginning of each uplink Transmission Time Interval (TTI), the evolved Node B (eNB) selects a system configuration that specifies the radio resources allocated to each group in order to accommodate the Random Access CHannel (RACH) procedure with the remaining resources used for data transmission. The key challenge is to optimally balance the allocations of channel resources between the RACH procedure and data transmission so as to provide reliable connections: Allocating too many resources for RACH enhances the random access performance, while leaving insufficient resources for data transmission.

This work was supported by the European Research Council (ERC) under the European Union Horizon 2020 research and innovative programme (grant agreement No 725731).

The eNB observes the number of successful transmissions and collisions on the RACH for all groups at the end of any TTI. This historical information can be potentially used to predict traffic and to aid the optimization of future TTIs' configurations. Even if one knew all the relevant statistics, tackling this problem in an exact manner can 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 traffic and channel statistics.

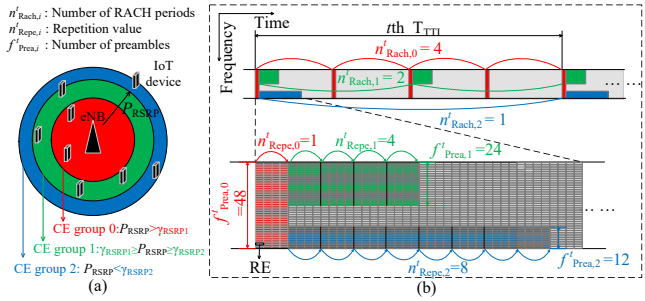

Fig. 1: (a) Illustration of system model; (b) Uplink channel frame structure.

In light of these challenges, prior works [3, 4] have tackled the problem under the simplifying assumptions that at most two configurations are allowed and that the optimization is done separately for each group without considering wireless transmission errors. In order to consider more complex and practical formulations, Reinforcement Learning (RL) emerges as a natural solution given the availability of feedback in the form of number of successful and unsuccessful transmissions per TTI. Q-learning based Access Class Barring (ACB) schemes using a tabular approach have been proposed in [5, 6] with the aim of optimizing the access success probability of the RACH. Finally, optimizing some of the parameters in NB-IoT, namely the repetition value (to be defined below), was carried out from the perspective of a single device in terms of latency and power consumption in [7] using a queuing framework.

In this paper, we develop a Cooperative Multi-Agent Deep Neural Network based Q-learning (CMA-DQN) algorithm for online uplink resource configuration in NB-IoT systems. In the proposed approach, a number of DQN agents are cooperatively trained to produce the configurations for the three CE groups. The reliance on Deep Neural Networks (DNNs) addresses the problem of tabular approaches [5, 6] in enabling operation over a large state space, while the use of multiple agents deals with the large dimensionality of the output space, corresponding to the configurations for the three CE groups.

The rest of the paper is organized as follows. Section 2 illustrates the system model. Section 3 discusses the conventional solution. Section 4 presents the CMA-DQN approach. Section 5 provides the numerical results and discussion.

2. 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 $\mathbb{R}^2$. The devices are divided into three CE groups as further discussed below. In each IoT device, uplink data is generated according to random inter-arrival processes over the TTIs, which are Markovian as defined in [8, Ch. 6.1].

2.1. Problem Formulation

Once backlogged, an IoT device executes the contention-based RACH procedure in order to establish a Radio Resource Control (RRC) connection with the eNB. This is done by transmitting a randomly selected preamble for a given number of times within the next RACH period of the current TTI. The RACH process can fail if: (*i*) a collision occurs between 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. 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 $n^t_{\text{Rach},i}$ of RACH periods in the TTI or the number $f^t_{\text{Prea},i}$ of preambles available in each RACH period [9]. Furthermore, in order to mitigate the SNR outage, the eNB can increase the number $n^t_{\text{Repe},i}$ of times that a preamble transmission is repeated by a device in group i in one RACH period [9] of the TTI.

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

In this work, we tackle the problem of optimizing the RACH configuration defined by parameters $A^t = \{n^t_{\text{Rach},i}, f^t_{\text{Prea},i}, n^t_{\text{Repe},i}\}_{i=0}^{2}$ 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 has available for all prior TTIs $t' = 1, ..., t-1$ the collection $U^{t'}$ consisting of the following variables: the number $V^{t'}_{\text{cp}}$ of the collided preambles the number $V^{t'}_{\text{sp},i}$ of the successfully received preambles, and the number $V^{t'}_{\text{ip},i}$ of idle preambles for RACH, as well as the number $V^{t'}_{\text{su},i}$ of IoT devices that have successfully sent data and the number $V^{t'}_{\text{un},i}$ of IoT devices waiting for scheduling. We denote as O^t=$\{U^1, A^1, ..., U^{t-1}, A^{t-1}\}$ the 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

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

where $\gamma \in [0, 1)$ is the discount rate for the performance accrued 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 3 and 4.

2.2. NB-IoT Access Network

We now provide additional details on the model and on the NB-IoT protocol. For the wireless channels, we consider the standard power-law path-loss model with path-loss exponent η and Rayleigh flat-fading. Once an IoT device becomes 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 [10]

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

where the received power $P_{\text{RSRP}} = P_{\text{NPBCH}} u^{-\eta}$ is averaged over small-scale Rayleigh fading, u is the device's distance from the eNB, and P_{NPBCH} is the broadcast power of eNB [10].

After CE group determination, each IoT device in group i repeats a randomly selected preamble $n^t_{\text{Repe},i}$ times in the next RACH period by using a pseudo-random frequency hopping schedule. The preamble consists of four so-called symbol groups, each occupying one RE [1, 11, 12]. 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 [12]. Assuming that correct detecting is determined by the SNR level $\text{SNR}^t_{\text{sg},j,k}$ for the jth repetition and the k symbol group, the correct detecting event S_{pd} can be expressed as

$$S_{\text{pd}} \triangleq \bigcup_{j=1}^{n^t_{\text{Repe},i}} \Big(\bigcap_{k=1}^{4} \big\{ \text{SNR}^t_{\text{sg},j,k} \geq \gamma_{\text{th}} \big\} \Big), \tag{3}$$

where γ_{th} is the SNR threshold, and the SNR can be written as $\text{SNR}^t_{\text{sg},j,k} = P_{\text{RACH},i} u^{-\eta} h / \sigma^2$ given the preamble transmit power $P_{\text{RACH},i} = \min\{\rho u^{\eta},\ P_{\text{RACHmax}}\}$ for $i = 0$ (CE group 0), and $P_{\text{RACH},i} = P_{\text{RACHmax}}$ for $i = 1$ or 2. Here, P_{RACHmax} is the maximal transmit power of IoT devices. Note that the preamble is transmitted using full path-loss inversion power control for CE group 0 [10], which ensures an average received power of ρ unless the power constraint is violated.

[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 [1].

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 are exceeded, the device switches to a higher CE group if possible [13]. Moreover, the IoT device is allowed to attempt the RACH procedure no more than γ_{pMax} times before dropping a packet.

To allocate data resources among the devices that have correctly completed the RACH procedure, 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 resources 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, and attempts will be made to schedule the device's data during these TTIs [14]. The condition that the data resources are sufficient in TTI t is expressed as

$$R^t_{\text{DATA}} \geq \sum_{i=0}^{2} r^t_{\text{DATA},i} V^t_{\text{sch},i}, \tag{4}$$

where $\sum_{i=0}^{2} V^t_{\text{sch},i} \leq \sum_{i=0}^{2} (V^t_{\text{sp},i} + V^{t-1}_{\text{unsc},i})$ is the number of scheduled devices; $r^t_{\text{DATA},i} = B_{\text{DATA}} \times n^t_{\text{Repe},i}$ is the number of required REs for serving a device within the ith CE group; and B_{DATA} is the number of REs per repetition. Note that the number of repetitions is the same as for preamble transmission [1].

3. CONVENTIONAL SOLUTIONS

Due to its complexity, most previous works simplify the optimization in (1) by considering the greedy formulation

$$(\text{P2}): \quad \max_{\pi(A^t|O^t)} \quad \mathbb{E}_\pi[V^t_{\text{succ},i}], \tag{5}$$

for some group i, whereby only the performance in the current TTI is considered. Furthermore, the expectation in (5) is approximated based on an estimate of the load in TTI t as discussed below; and the action space for A_t is typically reduced to include only some parameters such as the number $f^t_{\text{Prea},i}$ of preambles in each RACH period [3, 4].

To elaborate, we now briefly describe a solution based on [4] that follows the outlined simplifying principles. We drop the group index i in order to avoid unnecessary notation. At the beginning of each TTI, the scheme first estimates the number $\hat{D}^t_{\text{RACH}}$ of IoT devices that will attempt RACH access in the tth TTI, and then adjusts only the parameters $A_t = f^t_{\text{Prea}}$ according to the estimated load. The estimate is given as

$$\hat{D}^t_{\text{RACH}} = \max\{2V^{t-1}_{\text{coll}}, \zeta^{t-1} + \delta^t\}, \tag{6}$$

where the term $2V^{t-1}_{\text{coll}}$ reflects the fact that there are at least $2V^{t-1}_{\text{coll}}$ IoT devices colliding in the last TTI; $\delta^t \approx \delta^{t-1} = \hat{D}^{t-1}_{\text{RACH}} - \hat{D}^{t-2}_{\text{RACH}}$ is the difference between the estimated numbers of RACH attempting IoT devices in the $(t-1)$th and the tth TTIs [4]; and $\zeta^{t-1} = \log_{(f^{t-1}_{\text{Prea}}-1)/f^{t-1}_{\text{Prea}}}(V^{t-1}_{\text{idle}}/f^{t-1}_{\text{Prea}})$ is an estimate of the number of RACH-attempting IoT devices in the $(t-1)$th TTI obtained via moment matching [4].

Using the estimated load given in (6), the approach, which is referred to as Load Estimation based Uplink Resource Configuration (LE-URC), attempts to solve problem (5) by approximating the objective as

$$\mathbb{E}_\pi[V^t_{\text{succ}}] \approx \min\{\mathbb{E}\{V^t_{\text{reqs}} \big| \hat{D}^t_{\text{RACH}}\}, V^t_{up}\}, \tag{7}$$

where $\mathbb{E}\{V^t_{\text{reqs}} \big| \hat{D}^t_{\text{RACH}} = n\} = n\big(1 - \frac{1}{f^t_{\text{Prea}}}\big)^{n-1} + V^{t-1}_{\text{unsc}}$ is the expected number of IoT devices requesting uplink resource in the tth TTI; and $V^t_{\text{up}} = \frac{R_{\text{Uplink}} - R^t_{\text{RACH}}}{r^t_{\text{DATA}}}$ is an upper bound on the number of IoT devices can be scheduled.

4. COOPERATIVE MULTI-AGENT DQN APPROACH

We now introduce an RL-based approach to tackle problem (1). A direct application of the DQN approach [15] or of its enhancement proposed in [16], whereby the policy $\pi(A^t|O^t)$ for all t is modelled by a DNN, is not feasible due to the increasing size of the action A^t. In order to overcome this issue, we break up the action space by considering separately each of the nine action variables in A^t. Recall that we have three variables for each group i, namely $n_{\text{Rach},i}$, $n_{\text{Repe},i}$, and $f_{\text{Prea},i}$.

A separate DQN agent is introduced for each output variable in A^t. We define as A^t_k the action selected by the kth agent. Each kth agent is responsible to update the value $Q(S^t, A^t_k)$ of action A^t_k in state S^t, where the state variable $S^t = [U^{t-M_o}, A^{t-M_o}, ..., U^{t-1}, A^{t-1}]$ only includes information about the last M_o TTIs. All agents receive the same reward signal $R^t = \sum_{i=0}^{i=2} V^t_{\text{su},i}$ 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 [17] for multi-agent systems.

The DQN agents are trained in parallel. Each agent k parameterizes the action-state value function $Q(S^t, A^t_k)$ by using a function $Q(S, A_k; \theta_k)$, where $\boldsymbol{\theta}_k$ represents the weights matrix of a DNN with fully-connected layers. The input of the DNN is given by the variables in state S^t; the intermediate layers are Rectifier Linear Units (ReLUs); while the output layer is composed of linear units. Each output neurons provides the value of one of the actions in A^t_k as in [15]. The weights matrix θ_k is updated online along each training episode by using double deep Q-learning (DDQN) [16]. Accordingly, learning takes place over multiple training episodes, with each episode of duration N_{TTI} TTI periods. In each TTI, the parameters θ_k of the Q-function approximator $Q(S^t, A^t_k; \theta_k)$ are updated using Stochastic Gradient Descent at all agents k as

$$\theta^{t+1}_k = \theta^t_k - \alpha \nabla L_k(\theta^t_k), \tag{8}$$

where α is RMSProp learning rate [18], $\nabla L_k(\theta_k)$ is the gradient of the loss function $L_k(\theta^t_k)$ used to train the Q-function approximator. This is given as

$$\begin{aligned}\nabla L_k(\theta^t_k) =& \mathbb{E}_{S^i, A^i_k, R^{i+1}, S^{i+1}}\big[\big(R^{i+1} + \gamma \max_{A_k} Q(S^{i+1}, \\ & A_k; \bar{\theta}^t_k) - Q(S^i, A^i_k; \theta^t_k)\big)\nabla_{\theta_k} Q(S^i, A^i_k; \theta^t_k)\big],\end{aligned} \tag{9}$$

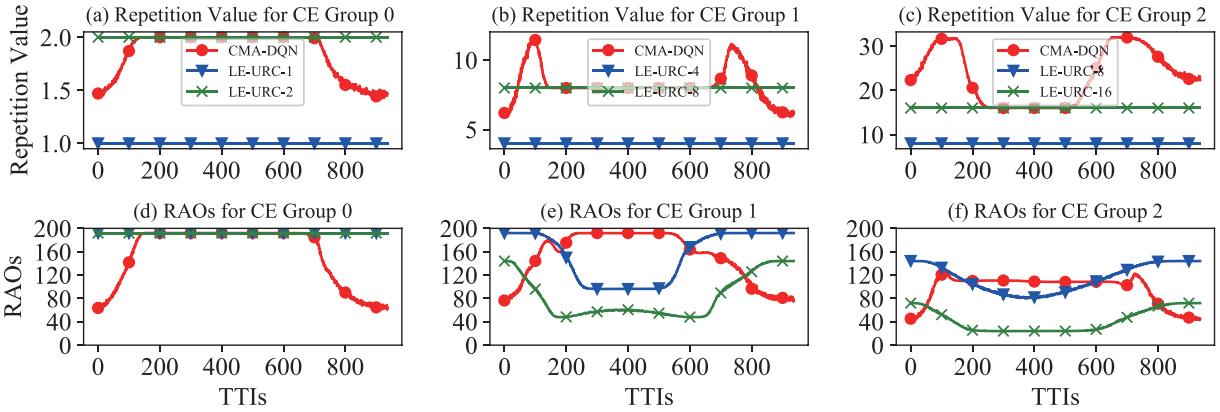

Fig. 3: The allocated repetition value $n^t_{\text{Repe},i}$, and RAOs producted by $n^t_{\text{Rach},i} \times f^t_{\text{Prea},i}$.

where the expectation is taken with respect to randomly selected previous samples $(S^i, A^i_k, S^{i+1}, R^{i+1})$ for some $i \in \{t - M_r, ..., t+1\}$, with M_r being the replay memory [15]. When $t - M_r$ is negative, this is to be intended as including samples from the previous episode. Following DDQN [16], in (9), $\bar{\theta}^t_k$ is a so-called target parameter 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 θ^t_k and kept fixed for a number of episodes.

5. SIMULATION RESULTS AND DISCUSSION

In this section, we evaluate the performance of the proposed CMA-DQN and compare it with the conventional LE-URC described in Sec. 3 via numerical experiments. The eNB is assumed to be located at the center of a circle area with 12 km radius, and we adopt the standard network parameters listed in Table 1 following [1, 2, 9, 13, 19]. Accordingly, one epoch consists of 937 TTIs (i.e., 10 minutes). Throughout epoch, each device have a bursty traffic profile, where the packet generation probability is given by the time limited Beta profile defined in [8, Ch. 6.1] with parameters $(3, 4)$, which has a peak around the 400th TTI. The resulting average number of generated packets is shown as dashed line in Fig. 2. The DQNs used by CMA-DQN have three hidden layers, each with 128 ReLU units, where other hyperparameters are listed in Table 1. All results are obtained by averaging over 1000 training episodes.

Table 1: Simulation Parameters and Q-learning hyperparameters

Simulation Parameters	Setting	Simulation Parameters	Setting
Path-loss exponent η	4	Noise power σ^2	-138 dBm
Received SNR threshold γ_{th}	0 dB	Power control threshold ρ	120 dB
eNB broadcast power P_{NPBCH}	35 dBm	TTI	640 ms
Bursty traffic duration	10 mins	IoT devices	30000
Maximum transmit power P_{RACHmax}	23 dBm	Set of number of preambles $\mathcal{F}_{\text{Prea}}$	{12,24,36,48}
Maximum resource requests γ_{RRC}	5	Set of repetition value $\mathcal{N}_{\text{Repe}}$	{1,2,4,8,16,32}
Maximum RACH in one CE $\gamma_{\text{pCE},i}$	5	Set of RACH periods $\mathcal{N}_{\text{Rach}}$	{1,2,4}
Maximum RACH attempts γ_{pMax}	10	RSRP threshold $\{\gamma_{\text{RSRP1}}, \gamma_{\text{RSRP2}}\}$	{0,-5} dB
REs required for B_{RACH}	4	REs required for B_{DATA}	32
Q-learning Hyperparameters	**Value**	**Q-learning Hyperparameters**	**Value**
Exploration ϵ	[0.1,1]	RMSProp Learning rate λ_{RMS}	0.0001
Discount rate γ	0.5	Minibatch size	32
Replay memory	10000	Target Q-network update frequency	1000

Fig. 2 compares the number of successfully served IoT devices V_{succ} during one epoch using 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 CMA-DQN slightly outperforms LE-URC in the light traffic regions at the beginning and end of the epoch, but it substantially outperforms LE-URC in the period of heavy traffic in the middle of the epoch. This demonstrates the capability of CMA-DQN to better manage the scarce channel resources 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.

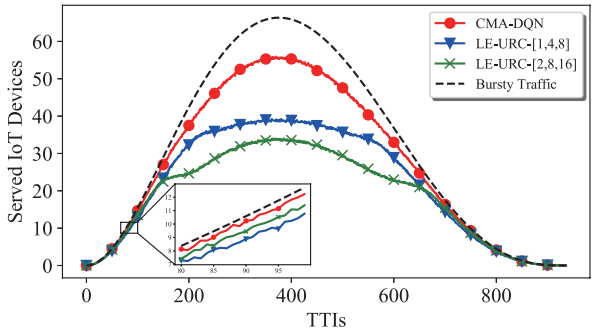

Fig. 2: The average number of successfully served IoT devices V_{succ} per TTI during one bursty traffic duration. The dashed line represents the average number of generated packets per TTI.

To gain more insight into the operation of CMA-DQN, Fig. 3 plots the average number $n^t_{\text{Repe},i}$ of repetitions and the average number of Random Access Opportunities (RAOs), defined as the product $n^t_{\text{Rach},i} \times f^t_{\text{Prea},i}$, for each CE group i that are selected by CMA-DQN over the training episodes. As seen in Fig. 3(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. As illustrated in Fig. 3(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. 2.

6. REFERENCES

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

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

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

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

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

[11] 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.

[12] 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.

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

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

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

[16] 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.

[17] 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.

[18] 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.

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