A Decoupled Learning Strategy for Massive Access Optimization in Cellular IoT Networks

Nan Jiang, *Member, IEEE*, Yansha Deng, *Member, IEEE*, Arumugam Nallanathan, *Fellow, IEEE*,
and Jinghong Yuan, *Fellow, IEEE*

Abstract— **Cellular-based networks are expected to offer connectivity for massive Internet of Things (mIoT) systems. However, their Random Access CHannel (RACH) procedure suffers from unreliability, due to the collision from the simultaneous massive access. Despite that this collision problem has been treated in existing RACH schemes, these schemes usually organize IoT devices' transmission and re-transmission along with fixed parameters, thus can hardly adapt to time-varying traffic patterns. Without adaptation, the RACH procedure easily suffers from high access delay, high energy consumption, or even access unavailability. With the goal of improving the RACH procedure, this paper targets to optimize the RACH procedure in real-time by maximizing a long-term hybrid multi-objective function, which consists of the number of access success devices, the average energy consumption, and the average access delay. To do so, we first optimize the long-term objective in the number of access success devices by using Deep Reinforcement Learning (DRL) algorithms for different RACH schemes, including Access Class Barring (ACB), Back-Off (BO), and Distributed Queuing (DQ). The converging capability and efficiency of different DRL algorithms including Policy Gradient (PG), Actor-Critic (AC), Deep Q-Network (DQN), and Deep Deterministic Policy Gradient (DDPG) are compared. Inspired by the results from this comparison, a decoupled learning strategy is developed to jointly and dynamically adapt the access control factors of those three access schemes. This decoupled strategy integrates predicted traffic into the learning process to improve training efficiency, where a Recurrent Neural Network (RNN) model is first employed to predict the real-time traffic values of the network environment, and then multiple DRL agents are employed to cooperatively configure parameters of each RACH scheme. Our results demonstrate that the decoupled strategy remarkably accelerate the training speedy.**

Index Terms—**RACH, overload control, energy-delay tradeoff, traffic prediction, deep reinforcement learning.**

I. INTRODUCTION

Cellular-based radio access technologies are required to support massive Internet of Things (mIoT) ecosystem, due to its high reliability, security, and scalability. In most state-of-the-art IoT systems, including enhanced Machine-Type Communication (eMTC), NarrowBand (NB)-IoT, and 5G New Radio (NR), Random Access CHannel (RACH) procedures have been adopted to establish synchronization between IoT devices and Base Stations (BSs). In these systems, time is organized into frames (a.k.a transmission time interval), where each frame consisting of multiple preambles (a.k.a. RACH channels) for IoT devices to request access. In each frame, IoT devices are activated by the requests from their upper layer applications that is unknown to the BS, in other words, these devices try to connect to the associated BS in an uncoordinated manner by selecting preambles at random. Due to this uncoordination, collisions occur when IoT devices select the same preamble at the same frame, which inevitably increases the access delay and the energy consumption, or even leads to service unavailability. To solve this problem, several access control schemes have been proposed in traditional access control works, including Access Class Baring (ACB) [2–4], Back-Off (BO) [5], Distributed Queuing (DQ) [6], and etc.. These schemes aim to improve the access success probability of RACH by intelligently organizing preamble transmission and re-transmission of IoT devices, but, its efficiency strongly depends on the incoming traffic of the IoT devices, which is generally intractable and dynamic over time.

With the goal of improving the access success, majority of efforts in previous works [2–9] have been devoted to formulate a mathematical model to describe the regularities of the practical communication environment as well as the traffic access pattern, so as to explicitly optimize the access control factor of each RACH scheme. Classical dynamic ACB optimizations have been studied in [2–4, 8, 9], where the ACB factor was configured based on the approximated future backlog estimation conducted by the methods of drift analysis [2], Method of Moments (MoM) [3], and Maximum-Likelihood Estimation (MLE) [4], respectively. In [8], a Bayesian traffic estimator is proposed by using the joint probability distribution function of the number of successful and collided access requests within the same frame, and its computational complexity is reduced by simplifying its recursion progress. In [9], a combination of the extended access barring and the access class barring is proposed to solve severe congestion that occurs when massive devices access simultaneously.

In [5], the class-dependent BO scheme has been proposed to guarantee the acceptable access delay of IoT devices with different priorities. In [6], a DQ scheme based on tree-splitting algorithm has been proposed to perfectly solve collisions, but

N. Jiang is with the Telecommunications Research Laboratory, Toshiba Research Europe Ltd., Bristol BS1 4ND, UK. This work was done while N. Jiang was working at Queen Mary University of London. (e-mail: nan.jiang@toshiba-bril.com).

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

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

J. Yuan is with the School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney NSW 2052, Australia (e-mail: j.yuan@unsw.edu.au).

how to select the tree-splitting factor for access optimization has not been discussed due to its analytical intractability. More recently, to fulfill mission-critical IoT applications, the network is expected to provide not only reliable wireless access, but also the low access delay and energy consumption. In [10–12], the energy-delay tradeoff in the RACH optimization has been studied from the perspectives of optimized extended ACB [10], power saving mode [11], and repetition values [12], respectively. However, due to its mathematical complexity, all these works balance the energy-delay tradeoff using queuing frameworks based on a static analytical model by ignoring the dynamic of the network system.

To deal with more complex communication environment and practical formulations, Machine Learning (ML), specifically Reinforcement Learning (RL), emerges as a promising tool to optimize RACH, due to that it solely relies on the self-learning of the environment interaction, without the need to derive explicit optimization solutions based on a complex mathematical model. Recent work [7] optimized the ACB scheme based on a tabular Q-learning algorithm, but this tabular method can not be used to solve other access optimization problems, due to its inefficiency in handling large state and action space. The most relevant work is our prior work [13, 14], where we developed a cooperative DQN algorithm for uplink resource configuration, including repetition value and RACH opportunities, to optimize the number of served IoT devices in NB-IoT networks. However, these learning-based RACH optimizations only focused on maximizing the number of access success devices at the BS side, whereas ignored the delay and energy consideration at the user side. Knowing the importance of the energy-delay tradeoff for mission critical IoTs [10–12], there is a need for an dynamically optimized RACH satisfying the access, energy, and delay requirements simultaneously in order to support massive access with delay and energy consideration in industrial IoT scenario.

To solve this problem, in this paper, we aim to develop a novel learning strategy to efficiently optimize the hybrid performance metric, taking into account the number of success accesses, the energy consumption of IoT devices, and the access delay of IoT devices for three main access control schemes. Our main contributions can be summarized as follows:

- To effectively optimize the existing RACH schemes, we first propose four DRL algorithms, including Policy Gradient (PG), Actor-Critic (AC), Deep Q-Network (DQN), and Deep Deterministic Policy Gradients (DDPG), where the PG, AC, and DQN target to optimize the BO and the DQ schemes with discrete action space, and the DDPG aims at optimizing the ACB scheme with continuous action space. All the DRL algorithms leverage the Recurrent Nerual Network (RNN) model, specifically, the Gated Recurrent Unite (GRU) architecture, to approximate their value function/policy. RACH schemes can be fairly compared, as they are efficiently optimized by using the similar DRL agents. Our results show that our proposed DRL-based RACH schemes significantly outperform conventional heuristic schemes in terms of the number of success accesses. The results also demonstrate that the DRL-based ACB scheme always outperforms the DRL-based BO, and DQ schemes in terms of the number of success accesses, but consumes much more energy in transceiving control signaling.

- In order to efficiently train DRL agents, we innovatively integrate domain knowledge from the communication, that is "the historical and present traffic statistics in the network are directly correlated with the future performance", into learning agents. To do so, we propose a novel decoupled learning strategy, where an RNN predictor is first employed to predict traffic statistics, and then several DRL agents are employed in parallel to configure RACH parameters by using those predicted traffic statistics as a belief state. The proposed learning strategy is able to be updated in an online manner, thus it can be pre-trained in the simulation environment during implementation, and then be transferred to practical environment. The proposed decoupled learning strategy can achieve better performance than the conventional DRL methods, and uses much less training time.

- We present a novel method to balance the energy-delay tradeoff in an online manner, where each RACH scheme is jointly optimized in terms of a hybrid Key Performance Indicators (KPIs) objective, including the number of access success, the energy consumption, and the access delay. The importance of each KPI can be adapted by configuring their weights, so as to fulfil different performance requirements of the network. To handle numerous action space conducted by multiple RACH schemes, we leverage the cooperative multi-agent DRL, where each DRL agent independently handles a single scheme, and shares their historical action selections to enable cooperation. The proposed hybrid scheme outperforms other single schemes in most combinations of KPIs.

- Finally, our proposed decoupled learning strategy is applied to optimize the number of served IoT devices in a practical case, namely, multi-group NB-IoT network. The simulation environment takes into account the RACH as well as the uplink channel resource scheduling based on the 3GPP reports [15, 16]. The result shown that our proposed decoupled learning strategy speeds up the training about 10 times than the conventional schemes in this case.

The rest of the paper is organized as follows. Section II formulates the problem and illustrates the system model. Section III provides preliminary and DRL-based single RACH scheme optimization. Section IV presents the hybrid scheme optimization and the decoupled learning strategy. Finally, Section V concludes the paper.

II. SYSTEM MODEL AND PROBLEM FORMULATION

We consider a cellular-based mIoT network system consisting of an arbitrary number of IoT devices and a single BS,

TABLE I: Notation Table

Notations	Physical means	Notations	Physical means
j	Index of IoT device	t	Index of frame
τ	Time	j	Index of device
f_{ACB}	ACB factor *	f_{BO}	BO factor *
f_{TD}	Depth of DQ scheme *	f_{TB}	Degree of DQ scheme *
V_{s}	Number of the access success devices	V_{c}	Number of the collided preambles
V_{i}	Number of the idle preambles	V_{e}	Average energy consumption of each success device
V_{d}	Average access delay of each success device	E_j	Energy consumption of device j
D_j	Access delay of device j	U^t	Transmission receptions at frame t
A^t	Actions at frame t	O^t	Observations at frame t
H^t	Historical observations from frame 1 to frame t	R^t	Reward at frame t
R_{s}^t	Success accesses reward component at frame t	R_{d}^t	Access delay reward component at frame t
R_{e}^t	Energy consumption reward component at frame t	x_s	Weight of success accesses reward component *
x_d	Weight of access delay reward component *	x_e	Weight of energy consumption reward component *
λ	learning rate	γ	Discount factor
k_j^i	Preamble group ID of CRQ i in device j	N^t	Exact traffic statistic at frame t
$\hat{N}^t$	Estimated traffic statistic at frame t	$\boldsymbol{\theta}$	Weights matrix of a neural network *
Remarks	The variables marked with * are configurable parameters.		

where they are in-synchronized, thus are unaware of the status of each other. For each IoT device, there is an random process for the generation of uplink data packets, which is unknown to the BS. We consider the time is divided into frames, and the packets inter-arrival processes are independent and identically distributed over the time frames, which are Markovian as defined in [15, 17].

A. Problem Formulation

We model the grant-based RACH procedure, where every IoT device has only two possible states[1], either *inactive* or *active*, while an IoT device with uplink data packets to be transmitted is in the latter case. Once active, an IoT device executes the RACH procedure in order to establish the synchronization with the BS. Without loss of generality, we focus on the RACH, and assume that a packet would always be completely transmitted if the IoT device succeeds in access. This assumption simplifies the study of RACH, which was specified in 3GPP report [17], and was considered in most prior works [2–4, 14, 18–20].

Briefly speaking, RACH is based on a framed-ALOHA principle, where an IoT device is allowed to transmit a randomly selected preamble during the first step of its procedure (details to be discussed in Sec. II-B). During this transmission, the RACH can fail if a collision occurs among two or more IoT devices selecting the same preamble. Once collided, IoT devices requires to retransmit in the following frames, which increases the backlog of the whole network. When massive collisions occur simultaneously, the network can be overloaded, which results in growing access latency and energy consumption. To tackle this challenge, one can allocate the transmission and re-transmission of these IoT devices using access control schemes to spread the traffic loads.

[1]The *grant-based* RACH procedure separates random access and grant-based data transmission, where packets in an IoT device will be completely transmitted if it access succeeds (assuming there are enough uplink channel resource). The *grant-free* access integrates access preamble and data into one sequence, where multiple sequences might be accumulated in the queue of an IoT device waiting for transmission. The proposed model in this paper can also support the study of *grant-free* access, with the minor modification by considering multiple packets queuing.

In this paper, we focus on the most common access control schemes, which are ACB, BO, and DQ schemes. A BS can adapt the intensity of these schemes in an online manner, each with a set of control factors, including f_{ACB}, f_{BO}, and $\{f_{\text{TD}}, f_{\text{TB}}\}$, respectively. Briefly speaking, increasing any control factor increases the intensity of this scheme. Given an arbitrary traffic scenario, a proper choice of the control factors can postpone transmissions into suitable future frames to release the network overload, while overusing these schemes can increase the waste of channel resources, the access delay, and the energy consumption. To enable efficient control, a BS may encompass mixtures of these schemes [21], as each is with different capability in overload control as well as different energy consumption during execution. We roughly summarize that the overload control capabilities of these schemes follows ACB>DQ>BO, while, in opposite, their energy consumptions follow BO>DQ>ACB. The key challenge is to optimally choose control factors of the BO, the DQ, and the ACB schemes so as to balance their utilization that provides sufficient overload control as well as consumes as little as energy consumption.

In this work, we aim to tackle the problem of optimizing the access control factors defined as $A^t = \{f_{\text{ACB}}^t, f_{\text{BO}}^t, f_{\text{TD}}^t, f_{\text{TB}}^t\}$ in an online manner for every frames. At the beginning of each frame t, the decision is made by the BS according to the transmission receptions $U^{t'}$ for all prior frames $t' = 1, ..., t-1$, which consists of the following variables: the number of the access success devices $V_{\text{s}}^{t'}$, the number of the collided preambles $V_{\text{c}}^{t'}$, the number of idle preambles $V_{\text{i}}^{t'}$, the average energy consumption of each success device $V_{\text{e}}^{t'}$, and the average access delay of each success device $V_{\text{d}}^{t'}$. Without loss of generality, we assume that each device j would report their energy consumption E_j and access delay D_j to the BS if their access succeeded. The access delay D_j of device j is normalized by counting the number of frames that this device was used for RACH, while the energy consumption E_j summarizes the energy that the device j was used for receiving system information and executing RACH (to be detailed in Sec. II-B3). Note that the observation in each

frame t includes all histories of such measurements and past actions, which is denoted as $H^t = \{O^1, O^2, ..., O^{t-1}\}$, each with $O^{t-1} = \{U^{t-1}, A^{t-1}\}$.

At each frame t, the BS aims at maximizing a long-term objective R^t (reward) related to the average access successes V_s^t, the average energy consumption V_e^t, and the average access delay V_d^t. The optimization relies on the selection of parameters in A^t according to the current historical observation O^t with respect to the stochastic policy π. This optimization problem can be formulated as:

$$(\text{P1}) : \max_{\pi(A^t|O^t)} \quad \sum_{k=t}^{\infty} \gamma^{k-t} R^k, \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor for the performance accrued in future frames, and the objective function R^t is formulated as

$$R^t = x_s R_s^t + x_d R_d^t + x_e R_e^t. \tag{2}$$

In (2), x_s, x_d and x_e are the weights of the success accesses reward component R_s^t, the access delay reward component R_d^t, and the energy consumption reward component R_e^t, respectively. Note that these three sub rewards are obtained by normalizing the observation of the average success accesses V_s^t, the average access delay reward V_d^t of each succeeded device, and the average energy consumption V_e^t of each succeeded device, respectively. The using of weighted rewards are inspired by the study of multi-criteria RL [22, 23], which targets to solve the problem that typically without available trade-offs among different objectives a priori. The derivation of these three sub rewards will be detailed in Sec. IV-A.

It it noted that the energy-delay optimization for access control has never been studied in an online manner. This is due to that, before synchronization, the access delay or the energy consumption of an IoT device is hardly known from the perspective of a BS. To solve it, the BS can collect the reports from those access success IoT devices. But, these reports are generally outdated, due to that they can only be gathered after the delay occurs (i.e., the access success IoT devices may already suffered a long delay or large energy consumption). As the dynamics of the system is Markovian over the frames, using these outdated information to maximize the future performance can be extremely intractable. Thus, the optimization task given in (1) becomes a generally intractable Partially Observable Markov Decision Process (POMDP).

Formally, this POMDP corresponds to a tuple $(\mathcal{S}, \mathcal{P}, \mathcal{A}, \mathcal{O}, \mathcal{Q}, \mathcal{R})$. In details, $\mathcal{S}$ is a set of actual environment's states with variables of transmission receptions $\{V_s, V_c, V_i, V_e, V_d\}$ and traffic statistics; P is a set of transaction probabilities between any two successive states; $\mathcal{A}$ is a set of actions, including all combinations of RACH control factors $\{f_{\text{ACB}}, f_{\text{BO}}, f_{\text{TD}}, f_{\text{TB}}\}$; $\mathcal{O}$ is a set of observations only with variables of transmission receptions $\{V_s, V_c, V_i, V_e, V_d\}$; Q is a set of observation probabilities describe the relationships between actions and observations; and $\mathcal{R}$ is a set of rewards given in Eq. (2) related to success accesses, energy consumption, and access delay. The partial observation refers to the fact that a BS cannot fully observe actual states in $\mathcal{S}$, while it can only access variables of transmission receptions without traffic statistics. The transition between any two successive states relies on the random collision process as well as the traffic generation process. Since neither processes can be observed, even if one knew the exact traffic statistics, calculating the set of transaction probabilities $\mathcal{P}$ and the set of observation probabilities $\mathcal{Q}$ would be generally intractable. In the following, approximate solutions will be discussed in Sections III and IV.

B. System Model

In our system model, we consider all frames have the same length, and each frame contains F available preambles. At the beginning of each frame, every IoT device can be activated according to a packets inter-arrival process. If activated, the IoT device receives the broadcast information from the BS, including the factors $\{f_{\text{ACB}}^t, f_{\text{BO}}^t, f_{\text{TD}}^t, f_{\text{TB}}^t\}$. According to the received information, the IoT device will determine how it executes RACH in the following frames. In the following, we describe the three main parts of system model: inter-arrival traffics, RACH procedure, and RACH schemes.

1) Inter-Arrival Traffics: Considering a bursty traffic scenario, where massive IoT devices are recovered due to an emergency event, e.g., earthquake alarm and fire alarms, and try to establish synchronization with the BS. Every IoT device would be activated at any time τ, according to a time limited Beta probability density function $p(\tau)$ given as [17, Section 6.1.1]

$$p(\tau) = \frac{\tau^{\alpha-1}(T - \tau)^{\beta-1}}{T^{\alpha+\beta-2}\text{Beta}(\alpha, \beta)}, \tag{3}$$

where T is the total time of the bursty traffic, and $\text{Beta}(\alpha, \beta)$ is the Beta function with the constant parameters α and β [24]. Considering that the newly activated devices at frame t only come from those who received an packet within the interval between with the last RACH period (τ^{t-1}, τ^t), the packet inter-arrival rate measured in each RACH period at each IoT device is hence described by

$$\mu^t = \int_{\tau^{t-1}}^{\tau^t} p(\tau)d\tau. \tag{4}$$

2) Random Access Schemes: Once activated at frame t, each IoT device synchronizes to the broadcast timing and receives the broadcast from the BS. The received system information are used to determine their RACH frame according to the mechanism of the RACH schemes, which includes the information of the RACH access control factors $\{f_{\text{ACB}}^t, f_{\text{TD}}^t, f_{\text{TB}}^t, f_{\text{BO}}^t\}$. These factors are updated at each frame in order to adaptively alleviate the collisions during RACH. Each IoT device should schedule its RACH transmission intervals according to each RACH scheme based on these factors. According to a prior definition, a BS may solely deploy any single access control scheme or jointly deploy mixtures of these schemes based on its capability and requirements in overload control. Every solely executed RACH scheme as well as their mixtures are detailed in the following:
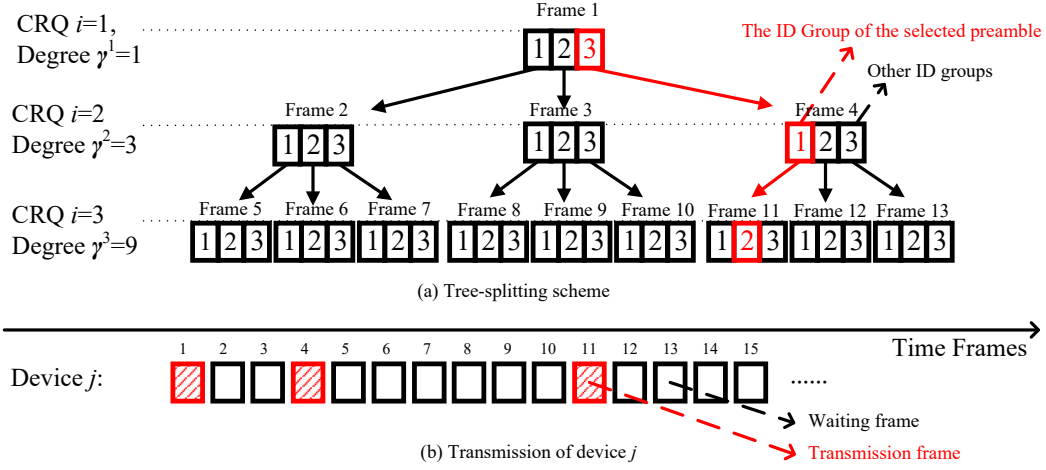
CRQ i=1,
Degree γ^1=1

Frame 1

The ID Group of the selected preamble

Other ID groups

CRQ i=2
Degree γ^2=3

Frame 2   Frame 3   Frame 4

CRQ i=3
Degree γ^3=9

Frame 5 Frame 6 Frame 7 Frame 8 Frame 9 Frame 10 Frame 11 Frame 12 Frame 13

(a) Tree-splitting scheme

Time Frames

Device j:

Waiting frame

Transmission frame

(b) Transmission of device j

Fig. 1: An example of the 3-ary ($f_{\text{TB}}^t = 3$) DQ scheme with depth $f_{\text{TD}}^t = 3$.

- **ACB scheme**: The ACB factor f_{ACB}^t is a probability value representing the possibility of an IoT device to execute RACH in the current frame t. At the beginning of a frame, each activated IoT device randomly generate a number q between 0 and 1, and attempts to RACH only when $q \leq f_{\text{ACB}}^t$, otherwise, the IoT device wait in this frame, and repeats the ACB check in the next frame.

- **DQ scheme**: One IoT device executes DQ scheme when it fails in an initial RACH attempt. After that, it will execute re-transmission according to a f_{TB}^t-ary splitting-tree algorithm with finite f_{TD}^t attempts. The degree f_{TB}^t refers to the number of branches that emanates from a tree node, while the depth f_{TD}^t refers to the number of branches between a tree node and the root node. Thoroughly, each branch of the tree contains the same number of the preambles, where the total preambles are sequentially indexed $\{1, 2, \cdots, F\}$, and equally divided into f_{TB}^t branches. For instance, assuming a 2-ary tree with $F = 54$ preambles, the first preamble group contains preambles with indexes from 1 to 28, and the second preamble group contains the others. The IoT devices uniformly distributed among these preamble groups, due to the preamble random selection process. Based on the preamble group IDentification (ID), the collided IoT devices will be allocated to retransmit preamble in a specific future frame. For IoT device j, its retransmission frame is obtained by calculating a logical Collision Resolution Queue (CRQ) i, which includes two parameters: 1) the total length of the CRQ $(f_{\text{TB}}^t)^i$ (number of nodes in depth i), and 2) the position of this IoT device μ_j^i in this CRQ. The IoT device j can only transmit preamble at frame μ_j^i during the CRQ i. Denoting the prior preamble group IDs of jth IoT device as $\{k_j^1, k_j^2, \cdots, k_j^{i-1}\}$, the position is calculated as

$$\mu_j^i = k_j^{i-1} + \sum_{k=1}^{i-2}(f_{\text{TB}}^t)^{i-1-k}(k_j^k - 1), \quad (5)$$

where k_j^{i-1} denotes the position index within a branch

in depth i, and $\sum_{k=1}^{i-2}(f_{\text{TB}}^t)^{i-1-k}(k_j^k - 1)$ denotes the number of unavailable frames before this branch in depth i. Once the ith CRQ finished, each collided IoT device would calculate their retransmission frame in the next CRQ $i + 1$ based on (5), until reaching the maximum depth f_{TD}^t. For better understanding, an example of 3-ary splitting-tree is given in Fig. 1, where an IoT device first selects the 3rd preamble group ($k_j^1 = 3$) in the 1st CRQ for transmission, and then selects the 1st preamble group ($k_j^2 = 1$) in 3rd position (i.e., $\mu_j^2 = 3$, frame 4) of the 2nd CRQ for transmission. If unsuccessful, it reattempts RACH in the 7th position (i.e., $\mu_j^3 = 7$, frame 11) of the 3rd CRQ, where it transmits a preamble randomly selected from the 2nd preamble group. If the last transmission in a tree still fails, one IoT device initializes a new tree for further retransmissions.

- **BO scheme**: One IoT device switches to a BO mode when it fails in an RACH attempt. To do so, this IoT device postpones its following retransmission attempts for a period, whose length is uniformly selected from $[2^{f_{\text{BO}}-1}, 2^{f_{\text{BO}}}]$ frames[2]. After BO, the IoT device would listen to broadcasting, and re-attempt RACH according to the newly received system information.

- **Hybrid scheme**: For the case of the hybrid scheme, one IoT device organizes its transmission and retransmissions of RACH requests according to a joint principle including ACB, BO, and DQ schemes. In the beginning, the IoT device executes an initialized RACH attempt once it passes an ACB check. Then, if this RACH attempt fails, the IoT device executes retransmissions according to the principle of spitting-tree during the following f_{TD}^t attempts. After these re-attempts, the unsuccessful IoT device would switch to a BO mode. Finally, after BO, the IoT device would listen to broadcasting, and re-

---

[2]3GPP also supports back-off period within $[0, 2^{f_{\text{BO}}}]$ as shown in [16], while its performance would be worse than the proposed one if the RL-based optimization was used.

start RACH from the beginning according to the newly received system information.

For the ACB scheme, the BS is required to broadcast system information to all active devices in each frame. It can lead to relatively higher energy consumption, due to that if an IoT device's access request was denied, the energy it consumed to receive system information in this frame would be wasted. Conversely, the other two schemes can save more receiving energy, due to the direct allocation of re-transmissions frames for each IoT device. However, the ACB scheme will accurately control the traffic volume in each frame, thus is more capable in achieving a lower average access delay, whereas the DQ and BO schemes may allocate re-transmissions into a future frame that is heavily overloaded, and potentially lead to more collisions and higher access delay. Through tackling the problem in (1), the access delay and energy consumption can be balanced without much sacrifice in the throughput.

3) RACH Procedure: Each IoT device determines their RACH frame according to the received system information and the mechanism of the RACH schemes. During their RACH frame, one should execute a four-step RACH procedure immediately. From the perspective of an IoT device, the RACH procedure includes: 1) the IoT device transmits a randomly selected preamble (Message 1, a.k.a., Msg 1) from a pool with F available preambles; 2) the IoT device waits to receive a Random Access Response (RAR) message (Msg 2) within an RAR window; 3) if Msg 2 was successfully received, the IoT device would send the Radio Resouce Control (RRC) connection request (Msg 3) to the associated BS; 4) finally, the IoT device receives RRC connection confirmed message (Msg 4) from the BS.

The RACH procedure may fail due to the collision, which occurs when two or more IoT devices selecting the same preamble in step 1. This collision comes from the fact that the BS cannot decode the Msg 3 of RACH, due to the overlapped transmissions from collided devices using the same channel at the same time [21]. To focus our study on the RACH schemes, we assume that all preambles can be transmitted without errors conducted by the effects of the physical radio channel, and the BS cannot decode any collided signals by using capture effect, as in prior works [2–4, 17]. Based on the RACH model in 3GPP [17], we assume the four-step RACH procedure, either in success or in collision, can finish within one frame (similar to [2–4, 11, 25]). Thus, if a preamble is collided, the related IoT devices can immediately re-attempt RACH in the next frame.

To investigate the performance of RACH, we consider three KPIs, including the average throughput V_s, the average access delay D, and the average energy consumption E. The average access delay D is averaging the total access delay over the number of success devices, where the delay is defined as the number of frames of the IoT device consumed from a newly packet arrival to the RACH success or fail (i.e., exceeds γ_{max} RACH attempts). The average energy consumption E is averaging the total energy consumption over the number of success devices calculated based on the system-level access model

described in [15, Sec. 7]. In details, the energy consumption of the jth IoT device to successfully access to the network is

$$E^j = E^j_{\text{sy}} + E^j_{\text{RACH}}, \tag{6}$$

where E^j_{sy} and E^j_{RACH} are the energy consumption of the jth IoT device in receiving broadcast signal and executing RACH, respectively. According to [15], E^j_{sy} and E^j_{RACH} can be obtained by calculating the product between their related average power consumption and working time. Therefore, Eq. (6) can be converted to

$$\begin{aligned} E^j = {} & n^j_{\text{sy}} T_{\text{sy}} P_{\text{sy}} + n^j_{\text{RACH}} \big(T_{\text{Msg1}} P_{\text{Msg1}} + T_{\text{Msg2}} P_{\text{Msg2}} \\ & + T_{\text{Msg3}} P_{\text{Msg3}} + T_{\text{Msg4}} P_{\text{Msg4}} \big), \end{aligned} \tag{7}$$

where n^j_{sy} and n^j_{RACH} are the numbers of frames that the IoT device j needed to receive broadcast signal and to execute RACH, respectively. In (7), T_{sy}, T_{Msg1}, T_{Msg2}, T_{Msg3}, and T_{Msg4} are constants, which are the consumed time in receiving broadcast signal, and in executing each step of RACH, respectively. Likewise, P_{sy}, P_{Msg1}, P_{Msg2}, P_{Msg3}, and P_{Msg4} are also constants representing the average power consumption per time unit for receiving broadcast signal, and for executing each step of RACH, respectively. The effects of physical radio channels have already been considered in these constant factors [15].

III. SINGLE SCHEME OPTIMIZATION VIA REINFORCEMENT LEARNING

In order to evaluate the capability of RL algorithms, in this section, we design several RL algorithms to solve problem (1) with each RACH scheme, to be compared with the existing conventional heuristic methods. These RL algorithms are capable in optimizing either a sole or a joint objective, by adapting the weight of each reward component. In the following, we first introduce the preliminary, including the perfect control strategy of the ACB scheme and the state-of-the-arts MLE-based ACB scheme. We then provide four state-of-the-arts RL algorithms, which are PG Reinforce and AC based on on-policy principle, and DQN and DDPG based on off-policy principle.

A. preliminary

Since the long-term hybrid target defined in Eq. (1) is complex, tackling this problem in an exact manner would be intractable. Classically, most prior works [2–9] simply optimizing the number of access success devices, without taking into account the energy consumption and the access delay. Only optimizing the success access, problem (1) transfers to

$$\max_{\pi(A^t|O^t)} \sum_{k=t}^{\infty} \gamma^{k-t} V_s{}^k, \tag{8}$$

or the even simpler one optimizing only the current frame

$$\max_{\pi(A^t|O^t)} V_s{}^t. \tag{9}$$

The former one (8) has been considered in RL-based ACB scheme optimization as [7], while the latter simplified one (9) has been considered in most RACH optimization using conventional heuristic methods as [2–6, 8].

Previous classical works [2–4, 7–9] have devoted majority efforts to study ACB scheme, due to its advantages in high scalability and reliability. Note that there exist no conventional method that can optimize the number of access success devices, the average energy consumption, and the average access delay simultaneously. In the following, we introduce the conventional heuristic methods to optimize ACB scheme.

Generally, the process of optimization is divided into two sub-tasks, which are traffic prediction and ACB factor configuration. The traffic statistic $\hat{N}^t$ is estimated based on the last observation $U^{t-1} = \{V_s^t, V_c^t, V_i^t\}$ obtained using MoM [3, 26, 27], MLE [4], drift analysis [2], or Bayesian method [8]. Then, according to the estimated backlog $\hat{N}^t$, the optimized ACB factor f_{ACB}^t for the next frame t can be calculated by using $f_{\text{ACB}}^t = \min(1, \frac{F}{\hat{N}^t})$ (proof can be found in [3, Sec. IV.A]), where F is the number of available preambles. Next, we first review the current state-of-the-art conventional traffic estimator of [4], which is based on MLE, and then describe the ideal ACB control scheme with known traffic, namely, Genie-aided ACB.

1) Maximum Likelihood Estimator (MLE): In [4], the traffic prediction problem is cast as a Bayesian probability inference problem, which calculates the probability for each possible traffic statistics at every frame t based on the last observation $U^{t-1} = \{V_s^t, V_c^t, V_i^t\}$. Due to the intractability of the given problem, an ideal assumption[3] that the current traffic load at frame $t-1$ (occurred) equals to the traffic load at the frame t (requires to be predicted) is made. However, even with this ideal assumption, the optimal Bayes estimator is still intractable [4]. To solve it, [4] presented the maximum likelihood of the Bayes estimator with respect to each traffic load value n under each possible observation u, which is given as

$$\hat{N}_{\text{ML}}^t = \hat{N}_{\text{ML}}^{t-1} = \underset{n \in \{0,1,\dots,N_{\max}\}}{\arg\max} \; \mathbb{P}\{U^{t-1}|N^{t-1} = n\}, \quad (10)$$

where $N_{\max}$ is an upper bound on the traffic load statistics to enable implementation. Note that each probability value in $\mathbb{P}\{U^{t-1}|N^{t-1} = n\}$ produces the likelihood of a value n with an observation u.

To solve problem (10), it is assumed that, at each frame t, each activated IoT device sequentially and independently chooses their preamble one after another, rather than choosing simultaneously in practice. This assumption does not change the uniformly selection principle of the random access, while the sequential selection process formulates a Markov chain to facilitate the calculation of likelihood $\mathbb{P}\{U^{t-1}|N^{t-1} = n\}$. Under this assumption, the vector of likelihoods $\mathbb{P}\{O^t|N^t = n\}$ for every n can be obtained by calculating the steady-state probability vector of the formulated Markov chain. In the run-time, the traffic statistics can be obtained by selecting the one with the maximal likelihood $\mathbb{P}\{U^{t-1}|N^{t-1} = n\}$ under

the specific observation U^{t-1}. The details on the numerical procedure of MLE traffic prediction can be found in [4].

2) Genie-Aided ACB: : We consider an ideal upper bound that the actual number of RACH requesting IoT devices N^t is available at the BS, namely, Genie-Aided ACB scheme. The BS optimizes the ACB factor f_{ACB}^t according to the real backlog N^t.

B. Reinforcement Learning

DRL is one of the most capable methods to optimally solve complex POMDP problems, due to the reliance on the deep neural networks as one of the most impressive non-linear approximation functions [28]. The related DRL algorithms have been widely used in the dynamic optimization for wireless communication systems, e.g., [13, 14]. However, despite that these algorithms are generally model-free, a direct application of these general DRL approaches can not facilitate the optimal solution in all sorts of dynamic optimization problems wireless systems. For instance, a discrete-action DRL algorithm (e.g., DQN [29]) may not be the best option to optimize the ACB scheme, due to its requirements in non-discrete control of the ACB factor (f_{ACB}^t). Therefore, we compare the state-of-the-arts DRL approaches to evaluate their capability in optimizing the number of access success devices of each RACH scheme.

We now introduce the general framework of DRL-based approaches to tackle problem (1). To optimize the number of success devices for a RACH scheme, we consider a DRL agent deployed at the BS, which learns to choose appropriate actions progressively by exploring the environment. One DRL agent is responsible for an output variable A^t (action), which represents the selected network parameter at the frame t. For instance, considering the ACB scheme, the output action is the ACB factor f_{ACB}^t for the frame t. The selection of variable A^t depends on the value $Q(S^t, A^t)$ or directly the policy π according to the observed state S^t. By using a delayed reward $R^t = V_s^{t+1}$, the DRL agent updates its policy π of action A^t, in an online manner, to progressively find the optimal solution of the RACH scheme for every state S^t.

Unlike the conventional methods presented in Sec. III-A, which only considers the single last observation O^{t-1}, the state variable S^t of one DRL agent at frame t consists of information in previous T_o frames $S^t = [O^{t-T_o}, O^{t-T_o+1}, ..., O^{t-1}]$. Including this historical information is due to that they can be useful to capture time-correlated features of the traffic generation mechanism and the RACH schemes. For instance, assuming the history length T_o is long enough, one BS may recognize the wake-up duration of the periodical activated IoT devices.

To recognize patterns in temporal data, we employ an RNN, specifically a GRU network, to approximate the value function $Q(S^t, A^t; \boldsymbol{\theta})$ or the policy $\pi(S^t|A^t; \boldsymbol{\theta})$ of each DRL algorithm, where $\boldsymbol{\theta}$ represents the weights matrix of the GRU RNN. A many-to-one *stateless*[4] implementation of GRU RNN is

---

[3]Note that this assumption is necessary to enable tractability of traffic prediction in most existing conventional heuristic traffic prediction methods [2–4].

[4]Different from the *stateless* implementation, the *stateful* RNN does not need to re-initialize the memory at each training step, while its training progress is more resource-hungry and less stable [30].

Algorithm 1: On-policy PG Reinforce/AC algorithms

---

input : Action space $|A|$, Operation Iteration I.

1 Algorithm hyperparameters: learning rate $\lambda \in (0,1]$, discount rate $\gamma \in [0,1)$;
2 Initialization of the parameterized policy $\pi(s|a; \boldsymbol{\theta})$ and its state-value function $v(s; \boldsymbol{w})$ if needed;
3 **for** *Iteration* $\leftarrow 1$ **to** I **do**
4   Initialization of S^0 by executing a random action;
5   **for** $t \leftarrow 0$ **to** $T-1$ **do**
6   Select action A^t according to the current policy $\pi(a|S^t; \boldsymbol{\theta})$;
7   The BS broadcasts the selected action A^t , and backlogged IoT devices execute RACH;
8   The BS observes S^{t+1}, and calculates the related $R^t = V_s^t$;
9   Restore the tuple S^t, A^t, R^t;
10   **end**
11   Obtain the trajectory of an episode $S^0, A^0, R^0, S^1, \cdots, S^{T-1}, A^{T-1}, R^T$;
12   Calculate return G^t for every frame t;
13   **PG**: Calculate the loss $\nabla L_{\text{PG}}(\boldsymbol{\theta}_{\text{PG}}^t)$ using Eq. (12);   **AC**: Calculate the critic loss $\nabla L_{\text{AC}}(\boldsymbol{w}_{\text{AC}}^t)$ and actor loss $\nabla L_{\text{AC}}(\boldsymbol{\theta}_{\text{AC}}^t)$ using Eq. (13) and Eq. (14), respectively;
14   Update the policy parameters $\boldsymbol{\theta}$ and state-value function $\boldsymbol{w}$ (AC algorithm).
15 **end**

---

adopted, where the historical observations in O^t is sequentially fed into the network, and only the RNN with the last input O^{t-1} is connected to the output layer for the further RACH parameter generation at frame t (to be detailed in follows). During the implementation, note that the memory of GRU RNN needs to be re-initialized at each frame, and the historical length T_o should generally be chosen according to the expected memory for time-correlation recognition.

The input of each DRL agent is the variables in state S^t; the intermediate layers are the introduced GRU RNN; while the output layer depends on different DRL algorithms. According to the unique training principle of each DRL algorithm, a loss function $L(\boldsymbol{\theta}^t)$ can be calculated to update the value function approximator $Q(S^t, A^t; \boldsymbol{\theta})$ (value-based algorithm) or the policy approximator $\pi(A^t|S^t; \boldsymbol{\theta})$ (policy-based algorithm). As the GRU RNN is used as the intermediate layers, we adopt standard Stochastic Gradient Descent (SGD) implemented via BackPropagation Through Time (BPTT) [31] for updating as

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \lambda \nabla L(\boldsymbol{\theta}^t), \quad (11)$$

where λ is the learning rate, and the loss function $L(\boldsymbol{\theta}^t)$ for each DRL algorithms will be detailed as follows. Here, we consider four DRL algorithms, including PG, AC, DQN, and DDPG. The former two are basic on-policy algorithms, while the latter two are state-of-the-arts off-policy algorithms. The learning principle of these four algorithms are described as follows.

1) PG Reinforce: In this algorithm, the DRL agent learns a parameterized policy π to select actions without consulting a value function. The output layer consists of a Softmax non-linearity with $|\mathcal{A}|$ number of probability factors, where $|\mathcal{A}|$ represents the size of action space. The probability $\mathbb{P}\{a^t = A^t|S^t, \boldsymbol{\theta}_{\text{PG}}^t\}$ of each possible action a is selected under the state S^t, which is parameterized by the weights $\boldsymbol{\theta}_{\text{PG}}^t$ at frame t. Recall that $\boldsymbol{\theta}_{\text{PG}}^t$ consists of both the GRU RNN parameters and the weights of the softmax layer. To train the policy $\pi(A^t|S^t; \boldsymbol{\theta}_{\text{PG}})$ (following (7)), the gradient of the loss function

$L(\boldsymbol{\theta}_{\text{PG}}^t)$ is given as

$$\nabla L(\boldsymbol{\theta}_{\text{PG}}^t) = \mathbb{E}_{S^i, A^i, G^i}\left[\gamma^i G^i \nabla_{\boldsymbol{\theta}} \ln \pi(A^i|S^i; \boldsymbol{\theta}_{\text{PG}})\right], \quad (12)$$

where the expectation is taken with respect to a minibatch for $i \in \{t-M_r, ..., t+1\}$ with size M_r, $G^t = \sum_{k=0}^{\infty} \gamma^k R^{t+k+1}$ is the return at frame t, and γ is the discount rate. The implementation of PG Reinforce algorithm for RACH optimization is shown in **Algorithm 1**.

2) AC: In this algorithm, a parameterized policy $\pi(A^t|S^t; \boldsymbol{\theta}_{\text{AC}})$ is learned to select actions called *actor*, and a state-value function $v(S^t; \boldsymbol{w}_{\text{AC}})$ is learned to evaluate the action called *critic*. The *actor* follows the setting of PG Reinforce, where the output layer includes $|A|$ number of Softmax units to generate policy probabilities. The *critic* is a state-value function with the output of one linear unit. The gradient of the loss function for training the *critic* $L(\boldsymbol{w}_{\text{AC}}^t)$ is given as

$$\nabla L(\boldsymbol{w}_{\text{AC}}^t) = \mathbb{E}_{S^i, G^i}\left[\gamma^t(G^i - v(S^i; \boldsymbol{w}_{\text{AC}}))\nabla_{\boldsymbol{w}_{\text{AC}}} v(S^i; \boldsymbol{w}_{\text{AC}})\right], \quad (13)$$

and the gradient of the loss function for training the *actor* $L(\boldsymbol{\theta}^t)$ is given as

$$\nabla L(\boldsymbol{\theta}_{\text{AC}}^t) = \mathbb{E}_{S^i, A^i, G^i}\Big[\gamma^i(G^i - v(S^i; \boldsymbol{w}_{\text{AC}}))\nabla_{\boldsymbol{\theta}_{\text{AC}}} \ln \pi(A^i|S^i; \boldsymbol{\theta}_{\text{AC}})\Big]. \quad (14)$$

The implementation of AC algorithm for RACH optimization is shown in **Algorithm 1**.

3) DQN: In this algorithm, the DQN agent learns a state-action value function approximator $Q(S^t, A^t; \boldsymbol{\theta}_{\text{DQN}})$ to select the action, where the output layer is composed of linear units. The weights matrix $\boldsymbol{\theta}_{\text{DQN}}$ is updated in a fully online manner, which occurs along each frame, so as to avoid the complexities of eligibility traces. Applying a double DQN training principle

---

Algorithm 2: Off-policy DQN/DDPG algorithms

input : Action space $\mathcal{A}$, Operation Iteration I.

1   Algorithm hyperparameters: learning rate $\lambda \in (0, 1]$, discount rate $\gamma \in [0, 1)$;

2   Initialization of the action-state value function $Q(s, a; \boldsymbol{\theta}_{\text{DQN}})$ for DQN, or the parameterized actor $\pi(s; \boldsymbol{\theta}_{\text{DDPG}})$ and critic $v(s, a; \boldsymbol{w}_{\text{DDPG}})$ for DDPG;

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

4   Initialization of S^0 by executing a random action;

5   **for** $t \leftarrow 0$ **to** $T - 1$ **do**

6   **DQN: if** $p_\epsilon^t < \epsilon$ **then** select a random action A^t from $\mathcal{A}$;   **DDPG**: Select $A^t = \pi(S^t; \boldsymbol{\theta}_{\text{DDPG}}) + \mathcal{N}^t$;

               else Select $A^t = \underset{a \in \mathcal{A}}{\arg\max}\, Q(S^t, a; \boldsymbol{\theta}_{\text{DQN}})$;

7   BS broadcasts action A^t, and backlogged IoT devices execute RACH;

8   BS observes S^{t+1} and calculates $R^t = V_s^{t+1}$;

9   Storing transition (S^t, A^t, R^t, S^{t+1}) in replay memory M;

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

11   **DQN**: Calculate the loss of Q-function $\nabla L(\boldsymbol{\theta}_{\text{DQN}}^t)$ **DDPG**: Calculate the loss of critic $\nabla L_{\text{DDPG}}(\boldsymbol{w}_{\text{DDPG}}^t)$ and actor

               using Eq. (15); $\nabla L_{\text{DDPG}}(\boldsymbol{\theta}_{\text{DDPG}}^t)$ using Eq. (16) and Eq. (17), respectively;

12   Perform a gradient descent for each primary network;

13   Update the target networks using: $\bar{\boldsymbol{w}}^t \leftarrow \sigma \boldsymbol{w}^t + (1 - \sigma) \bar{\boldsymbol{w}}^t$, and $\bar{\boldsymbol{\theta}}^t \leftarrow \sigma \boldsymbol{\theta}^t + (1 - \sigma) \bar{\boldsymbol{\theta}}^t$.

14   **end**

15   **end**

---

[32], the gradient of loss function $L(\theta_{\text{DQN}}^t)$ used to train the value function approximator $Q(S^t, A^t; \boldsymbol{\theta}_{\text{DQN}})$ is given as

$$
\begin{aligned}
\nabla L(\boldsymbol{\theta}_{\text{DQN}}^t) =& \mathbb{E}_{S^i, A^i, R^{i+1}, S^{i+1}} \Big[\big(R^{i+1} + \\
& \gamma \max_{a \in \mathcal{A}} Q(S^{i+1}, a; \bar{\boldsymbol{\theta}}_{\text{DQN}}^t) - \\
& Q(S^i, A^i; \boldsymbol{\theta}_{\text{DQN}}^t) \big) \nabla_{\boldsymbol{\theta}_{\text{DQN}}} Q(S^i, A^i; \boldsymbol{\theta}_{\text{DQN}}^t) \Big],
\end{aligned} \tag{15}
$$

where $\bar{\boldsymbol{\theta}}_{\text{DQN}}^t$ is a so-called target value function. Note that $\bar{\boldsymbol{\theta}}_{\text{DQN}}^t$ is initialized using the same network structure as the primary DQN $\boldsymbol{\theta}_{\text{DQN}}$, which is only used to estimate the future value of the Q-function. The parameter of the target network $\bar{\boldsymbol{\theta}}_{\text{DQN}}^t$ is partially copied from the primary value $\boldsymbol{\theta}_{\text{DQN}}$ at each frame (implementation details are shown in **Algorithm 2**). The hybrid use of the primary DQN and its target network forms the so-called double DQN. Note that, different from the PG and the AC algorithms given in Sec. III-B1 and Sec. III-B2, the expectation here is taken with respect to a random minibatch. This minibatch is uniformly picked in random from a finite replay memory with size M. The implementation of DQN algorithm for RACH optimization will be detailed in **Algorithm 2**.

4) DDPG: This algorithm adapts the ideas of DQN to the continuous action selection. Rather than discrete action control in the BO and the DQ schemes, the DDPG can be applied for continuous action control in ACB access scheme. Similar to AC algorithm, DDPG leverage an *actor* $\pi(S^t; \boldsymbol{\theta}_{\text{DDPG}})$ to learn action selection, while the *critic* uses a state-action value function $v(S^t, A^t; \boldsymbol{w}_{\text{DDPG}})$ to evaluate the action. The input of *critic* includes both the current state S^t as well as the current action A^t (obtained by the *actor* $\pi(S^t; \boldsymbol{\theta}_{\text{DDPG}})$), and the output is a linear unit. In terms of the action selection, the *actor* deterministically maps a state to a specific action according to the current policy $\pi(S^t; \boldsymbol{\theta}_{\text{DDPG}})$. To do so, the output of *actor* uses a Sigmoid non-linearity unit, which generates continuous numbers within $(0, 1)$. The Sigmoid output perfectly matches the range of the ACB factor $(0, 1]$, thus it can be directly used

as the ACB factor at each frame t. During the training, *critic* $L(\boldsymbol{w}_{\text{DDPG}}^t)$ is updated by minimizing the loss using

$$
\begin{aligned}
\nabla L(\boldsymbol{w}_{\text{DDPG}}^t) =& \mathbb{E}_{S^i, A^i, R^{i+1}, S^{i+1}} \Big[\big(R^{i+1} + \\
& \gamma v(S^{i+1}, \bar{\pi}(S^i; \bar{\boldsymbol{\theta}}_{\text{DDPG}}); \bar{\boldsymbol{w}}_{\text{DDPG}}^t) - \\
& v(S^i, A^i; \boldsymbol{w}_{\text{DDPG}}^t) \big) \nabla_{\boldsymbol{w}_{\text{DDPG}}} v(S^i, A^i; \boldsymbol{w}_{\text{DDPG}}^t) \Big],
\end{aligned} \tag{16}
$$

where $\bar{\boldsymbol{w}}_{\text{DDPG}}$ and $\bar{\boldsymbol{\theta}}_{\text{DDPG}}$ are the weights of the target critic and actor, respectively. Similar to DQN, these two target networks are updated by partially copying from the primary two. To train the *actor* $L(\boldsymbol{\theta}^t)$, the gradient of the loss function is given as

$$
\begin{aligned}
\nabla L(\boldsymbol{\theta}_{\text{DDPG}}^t) =& \mathbb{E}_{S^i, A^i, R^{i+1}, S^{i+1}} \Big[\nabla_a v(S^i, a; \boldsymbol{w}_{\text{DDPG}}^t)|_{a = \pi(S^i)} \\
& \nabla_{\boldsymbol{\theta}_{\text{DDPG}}} \pi(S^i; \boldsymbol{\theta}_{\text{DDPG}}) \Big].
\end{aligned} \tag{17}
$$

In (16) and (17), the expectation is also taken using a random minibatch. The implementation of DDPG algorithm for RACH optimization is shown in **Algorithm 2**.

C. Numerical Results and Evaluation

In this subsection, we evaluate the number of access success devices of the ACB, BO, and DQ schemes using our proposed four DRL algorithms above via numerical experiments. We adopt the standard network parameters following 3GPP technical report for MTC systems [15, 17], where the number of preambles $F = 54$, retransmission constraint $\gamma_{\max} = 10$, and each frame contains 640 milliseconds (ms). Unless otherwise stated, we mostly assume the presence of devices $N = 400$ generating a packet at random according to the time limited Beta profile with parameters $(\alpha, \beta) = (3, 4)$ with period $T = 20$. An example of the resulting average number of activated devices of this Beta profile is shown as orange dashed line in Fig. 2. For different DRL algorithms, we use the same hyperparameters for the training as well as the testing for fair comparison. Each DRL agent is with three layers, where

TABLE I: RL Hyperparameters

Hyperparameters	Value	Hyperparameters	Value
Memory T	20	Learning rate λ	0.0001
Minibatch size	32	Historical samples size M	10000
Minimal exploration rate ϵ	0.01	ACB scheme discount rate γ_{ACB}	0.1
BO and DQ schemes discount rate γ	0.9	Target network update rate σ	0.2

the first two layers are each with 128 GRU units, and the last layer is with 128 Rectifier Linear Units (ReLUs). The other hyperparameters can be found in Table I. The action of ACB f_{ACB}^t is a factor selected from $(0, 1]$, where the discrete model selects factor with the minimum pace of 0.05, and the continuous one can select any value without any limitation. The BO and DQ schemes can only be controlled in a discrete manner, where the action of BO f_{BO}^t is an integer selected from $[0, 8]$, and the DQ scheme can use any tree smaller than $\{f_{\text{TD}}^t, f_{\text{TB}}^t\} = \{3, 6\}$.
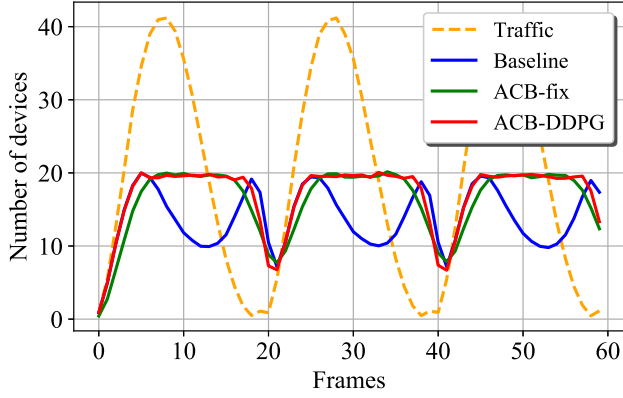
(a) $T = 20$ frames
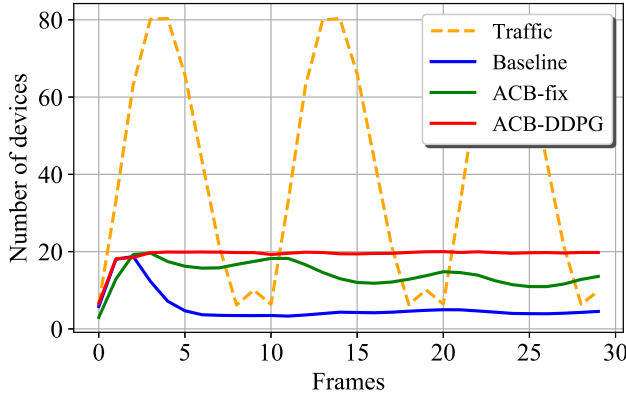
(b) $T = 10$ frames

Fig. 2: The average number of successfully accessed IoT devices V_s per frame over 1000 epochs, where (a) uses the time limited Beta traffic profile with period $T = 20$, and (b) uses the similar traffic profile with period $T = 10$. The dashed line represents the average number of IoT devices with new arrival packets per frame.

We first illustrate the operation of the proposed traffic and ACB scheme under the time limited Beta traffic profile with period $T = 20$ in Fig. 2(a) and period $T = 10$ in Fig. 2(b). Fig. 2 plots the average number of IoT devices under new arrival

packets in each frame, and the average number of the access success devices V_s per frame for the baseline scheme, the fixed-factor ACB scheme, as well as the DDPG-based ACB scheme, respectively. The baseline refers to transmissions without any access control, the fixed ACB scheme uses the fixed factor $f_{\text{ACB}}^t = 0.5$, and the DDPG-based ACB scheme relies on the DDPG algorithm to adaptively assign the ACB factor in each frame as given in Sec. III-B2. We first observe that, with the smaller traffic arrival period, the new arrival packets in Fig. 2(b) is much more intense than that in Fig. 2(a). In Fig. 2(a), it is observed that the access performance of the baseline scheme (i.e., without ACB control) reduces dramatically after the new arrival traffic hits the peak. This is due to that the number of access requests during this period has been accumulated to the maximum, which increases collisions. In contrary, the access performance of fixed-factor ACB schemes remain at around 20 during that period, but is worse than that of the baseline scheme when the traffic is small. This is because the access baring mechanism reduces collisions during the peak traffic period, but also decreases the preamble utilization during the non-peak traffic period. Different from Fig. 2(a), the access performance in Fig. 2(b) always follow ACB-DDPG>ACB-fix>Baseline, due to its much heavier incoming traffic. It can be seen that, in both figures, the DDPG-based ACB scheme performs the best at both peak and non-peak periods, which is due to the capability of ACB-DDPG in assigning ACB factor in an online manner to adaptively control the number of access devices in an acceptable level.
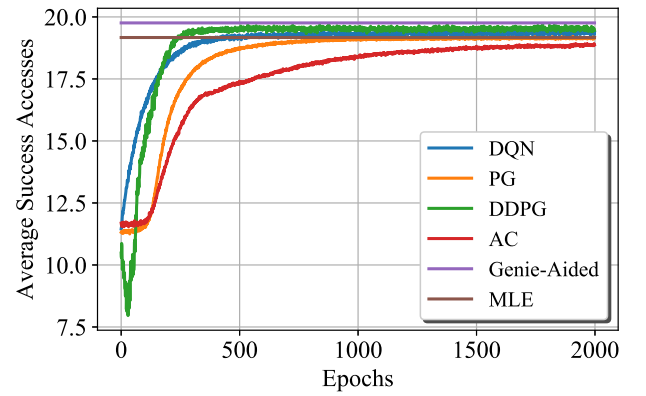

Fig. 3: Average number of successfully accessed IoT devices per frame as a function of the frames in the online training phase.

Fig. 3 compares the evolution of the average number of the access success devices per episodes for each RL algorithm in the training phase under the ACB scheme, where each result is averaged over 100 training trails. The training curves are also compared with lower and upper bounds using the

MLE-based ACB scheme and the ideal genie-aided ACB scheme, respectively. It can be seen that, after convergence, the proposed DDPG and DQN methods are quite close to the ideal upper bound, and slightly outperform the conventional MLE-based ACB scheme. This is due to the fact that DDPG and DQN methods are more capable in accurately capturing the time correlation among historical transmission receptions and the configuration than MLE. MLE is not capable of capturing historical trends in the traffic, since it simplifies the prediction problem by assuming that a forthcoming traffic statistic is equal to a present one [27]. The improvement of DDPG and DQN methods (around 2.5%) is relatively minor, due to that the optimization of the ACB scheme is mathematically tractable using MLE. In fact, the ACB factor would be optimally selected, if the actual backlog was given (according to Sec. II-B). Despite that the improvement of ACB scheme is relatively minor, we argue that the use of RL algorithms in RACH optimization is still worthwhile. The reason is that the optimization of the other RACH schemes are mathematically intractable, such as the presented BO and DQ schemes.
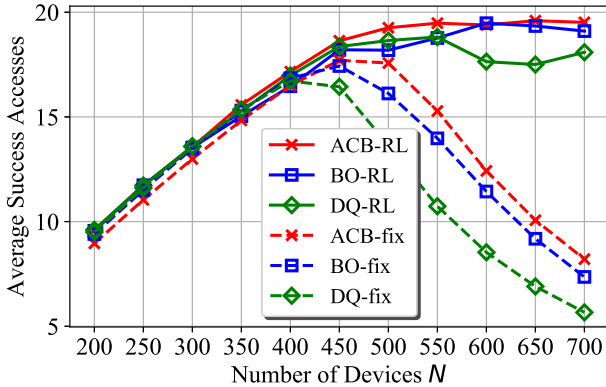
Fig. 4: The average number of successfully accessed IoT devices V_s during 1000 testing episodes.

In Fig. 3, it is seen that the off-policy algorithms (DDPG and DQN) outperform the other on-policy algorithms (PG and AC) in terms of both the adaptation speed and the performance after convergence. This is mainly due to the experience replay mechanism, which efficiently utilizes the training samples, and also due to the randomly sampling, which smooths the training distribution over the previous behaviors. We further observes that, despite the similar learning principle and neural network architecture used, the DDPG slightly outperforms the DQN. This is due to the continuous control mechanism of DDPG, which provides the infinite action space of the ACB configuration, whereas the DQN is only limited to the configured 20 action space. Motivated by this, we use DDPG for ACB configuration and DQN for any other discrete factors configurations, including BO and DQ in the rest of figures.

Furthermore, in practice, one important reason to use off-policy RL algorithms in RACH optimization is that their training and updating processes can be deployed in a cloud or the edge by gathering samples from multiple BSs, which would greatly save computational resource. However, it is

emphasized that, despite that off-policy RL algorithms were used, the feedfrowarding process of these algorithms is still computation-hungry. In detail, by running simulations with Python on a personal computer with an Intel Core i5-9600K processor, the processing time of executing ACB for one frame with the MLE predictor is about 0.043 ms, while that with the DDPG algorithm is about 0.899 ms. This extra execution delay of employing learning-based algorithms in the network needs to be considered during implementation, despite that this delay can be reduced when employed in a BS with higher computational capability. Fortunately, even with the limited computational resources, the execution time of learning agents is much fewer than 5 ms, which is the minimal period of RACH opportunities in most cellular networks, including LTE, NB-IoT, eMTC, 5G NR.

Fig. 4 plots the average number of the access success devices per frame per episode using the fixed-factor RACH schemes and RL-based RACH schemes versus various numbers of devices N. The "ACB-fix", "BO-fix", and "DQ-fix" are the ACB, BO, and DQ schemes based on the fixed factors of $f_{ACB}^t = 0.5$, $f_{BO}^t = 2$ (back-off from $[2,4]$ frames), and $\{f_{TD}^t, f_{TB}^t\} = \{2,2\}$, respectively. It is observed that all schemes achieve similar performance when the number of devices N is smaller than 400, but the RL-based schemes substantially outperform the conventional fixed-factor (non-dynamic) access control schemes in heavy traffic region, where the number of devices N is bigger than 400. This showcases the capability of RL algorithms to better optimize each RACH scheme, as they can well manage access load in the presence of heavy traffic. It is also interesting to note that "ACB-RL" generally outperforms the "BO-RL" and "DQ-RL" in any number of devices. This is because the ACB schemes can control the access request probability of each device in each frame, whereas the "BO-RL" and "DQ-RL" may allocate re-transmissions into a future frame that is heavily overloaded, which may lead to more collisions.

IV. HYBRID SCHEME OPTIMIZATION AND DECOUPLED LEARNING STRATEGY

In this section, we aims at solving problem (1) and jointly optimizes three RACH schemes defined by factors $A^t = \{f_{ACB}^t, f_{BO}^t, f_{TD}^t, f_{TB}^t\}$. This hybrid scheme integrates the features of the ACB, the BO, and the DQ schemes, where they may be jointly executed according to control of the DRL agents. Different from the single scheme scenario in Sec. III, this joint execution increases adjustable system factors, which results in an exponential increment of the action space. As evaluated in our previous work [14], optimizing a system with such numerous action space by a single RL agent is not feasible, due to the convergence difficulty. To solve problem (1), in this section, we propose a decoupled learning strategy to efficiently train multiple parallel RL agents that each handles one access control factor. In the following, we first introduce the basic multi-agent cooperative RL for multi-factors optimization. After that, we propose the decoupled learning strategy in detail.
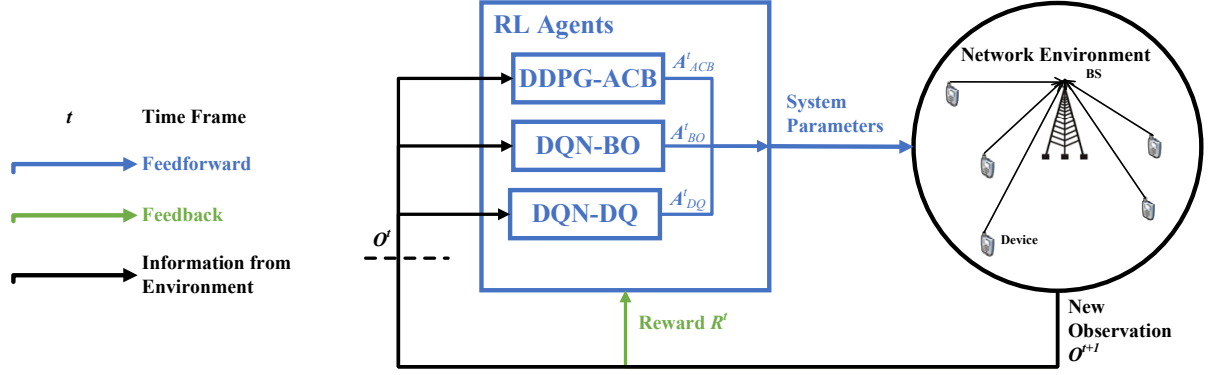
(a) Conventional multi-agent cooperative DRL structure.
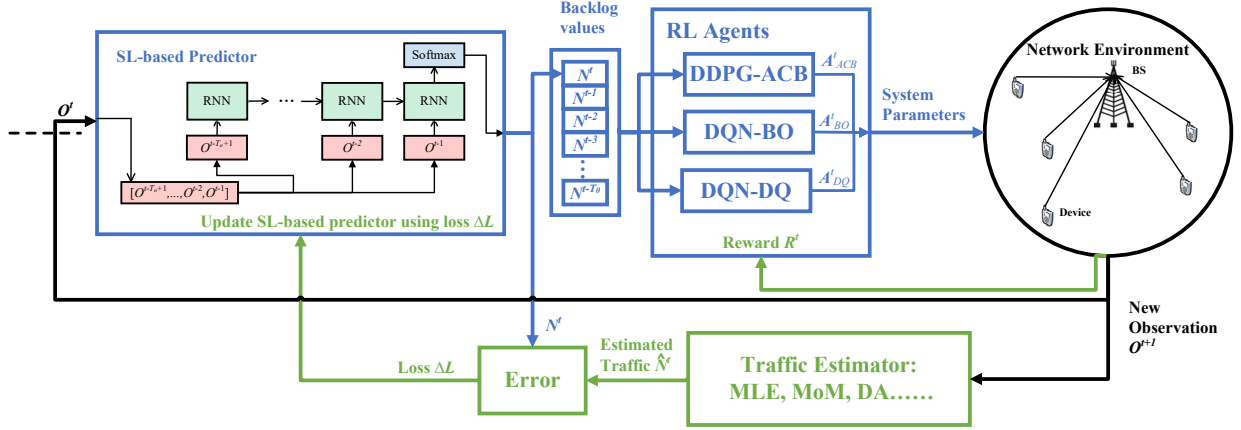
(b) Decoupled learning strategy for multi-agent cooperative DRL.

Fig. 5: Proposed learning structure of (a) conventional multi-agent cooperative DRL, and (b) decoupled multi-agent cooperative DRL.

A. Multi-Agent Cooperative Deep Reinforcement Learning

In this subsection, we introduce a basic multi-agent cooperative DRL method to tackle problem (1). As evaluated, a direct application of DQN or DDPG given in Sec. III is not feasible to solve this problem, due to the enormous size of the action A^t. To solve this problem, the action space $A^t = \{f^t_{\text{ACB}}, f^t_{\text{BO}}, f^t_{\text{TD}}, f^t_{\text{TB}}\}$ can be broken down into three separate sub-actions, each controls one scheme, including A^t_{ACB}, A^t_{BO}, and A^t_{DQ}. As illustrated in Fig. 5 (a), we consider three independent RL agents to handle each of these actions, where the first, A^t_{ACB}, is handled by a DDPG agent, and the other two are handled by two DQN agents. Each agent parameterizes their own value function $Q(S^t, A^t)$ or policy π by using a function θ, which is represented by the multiple layers GRU RNN structure given in III-B.

In each frame, the state variable $S^t = [O^{t-T_o}, O^{t-T_o+1}, ..., O^{t-1}]$ is fed into each agent to generate sub-actions, where each observation O not only includes the historical transmission receptions U, but also the historical action selection of every agent $A = \{A_{\text{ACB}}, A_{\text{BO}}, A_{\text{DQ}}\}$. The share of historical action selection among each agent aims to benefit their cooperation. By doing so, each agent

can understand how the total reward is influenced by each sub-action, so as to predict the future action selection of each other. After each frame, DRL agents are trained in parallel, where their parameters θ and/or w are updated using the approaches given in Eq. (12)-(17) of Sec. III-B. Note that this cooperative training approach has been proposed in our previous works [13, 14], and is akin to theproportional approach proposed in [23], which has been evaluated as a close replacement of the overall function by a factorization.

The training of each DRL agent shares a common reward signal, which guarantees that all of them aims at the same objective as given in Eq. (1). The reward is obtained by using the weighted sum of the success accesses reward R^t_{s}, the access delay reward R^t_{d}, and the energy consumption reward R^t_{e} as given in Eq. (2). The success accesses reward R^t_{s} is derived by directly normalizing the observed average success accesses V^t_{s}. On contrary, the access delay reward R^t_{d} and the energy consumption reward R^t_{e} are inversely to the average access delay V^t_{d} of each succeeded device and the average energy consumption V^t_{e}, which are derived by using a revised hyperbolic tangent activation function (a.k.a., tanh function) as

$$R^t = 1 - \frac{e^{\frac{V^t}{c}} - e^{-\frac{V^t}{c}}}{e^{\frac{V^t}{c}} + e^{-\frac{V^t}{c}}}, \qquad (18)$$

where c is a constant factor, and V^t represents to V_e^t or V_d^t. In (18), the use of the revised tanh function targets to not only generating the sub rewards R_d^t and R_e^t that is inversely proportional to observations V_d^t and V_e^t, but also limiting the sub rewards in the range of $(0, 1]$ to improve the convergence capability of neural networks. Note that the constant factor c is used to flexibly determines the density of reward, where its selection impacts on both the convergence capability and the training efficiency of neural networks.

B. Decoupled Learning Strategy for Hybrid Scheme Optimization

Applying conventional DRL methods to optimize massive access control faces two challenges: i) the DRL agent is expected to be updated in an online manner, but the convergence is really slow due to the complexity of neural network as well as the tradeoff between exploration and exploitation; and ii) the access performance of the converged DRL agents degrades by various hidden information, including, but not limited to, the traffic arrival pattern and random collision occurrence. Aiming at tackling these two challenges, in this subsection, we propose a decoupled learning strategy to train multiple parallel DRL agents as shown in Fig. 5(b). Different from conventional DRL, problem (1) was decoupled into two sub-tasks, including traffic prediction and parameter configuration. These two sub-tasks are successively solved, where we first predict the traffic by using an online supervised learning method [27], and we then input the predicted traffic statistic into DRL agents to configure actions. Details will be discussed in following two parts.

1) Online Supervised Learning for Traffic Prediction: This learning-based predictor adopts a modern RNN model with parameters $\boldsymbol{\theta}_{\mathrm{RNN}}$ to predict the traffic statistic at each frame. The use of RNN is due to its capability in capturing time correlation of traffic statistics over consecutive frames, which can help to learn the time-varying traffic trend for better prediction accuracy. As shown in Fig. 5(b), historical observations $O_{t-T_o}^{t-1}$ are sequentially inputted into the RNN predictor in a *stateless* manner, and only the RNN with the last input O^{t-1} is connected to the output layer. The output layer consists of a Softmax non-linearity with ($N_{\max}+1$) values, which represent the predicted probability $\mathbb{P}\{\hat{N}^t = n | O_{t-T_o}^{t-1}, \boldsymbol{\theta}_{\mathrm{RNN}}\}$ of each possible value n for the forthcoming backlog N^t. Recall that $N_{\max}$ is the upper bound on the backlog statistics to ease implementation. In order to enable online updating, we also implement the label estimation method proposed in [27]. Briefly speaking, with respect to a produced backlog $\hat{N}^t$ at any frame t, the weights of the RNN are adapted with a delay of one frame at frame $t+1$ by using an estimated label $\tilde{N}^t$. This label $\tilde{N}^t$ can be obtained at the next frame $t+1$ by using the MLE estimator described in Sec. II-A, or MoM estimator

described in [3, 27]. At each frame $t+1$, the parameters $\boldsymbol{\theta}_{\mathrm{RNN}}$ of the RNN predictor are updated using SGD via BPTT as

$$\boldsymbol{\theta}_{\mathrm{RNN}}^{t+1} = \boldsymbol{\theta}_{\mathrm{RNN}}^t - \lambda \nabla L(\boldsymbol{\theta}_{\mathrm{RNN}}^t), \qquad (19)$$

where $\nabla L(\boldsymbol{\theta}_{\mathrm{RNN}}^t)$ is the gradient of the loss function $L(\boldsymbol{\theta}_{\mathrm{RNN}}^t)$ to train the RNN predictor. This is obtained by averaging the cross-entropy loss as

$$L^t(\boldsymbol{\theta}_{\mathrm{RNN}}) = - \sum_{t'=t-T_b+1}^{t} \log \left(\mathbb{P}\{\hat{N}^{t'} = \tilde{N}^{t'} | O_{t'-T_o}^{t'}, \boldsymbol{\theta}_{\mathrm{RNN}}\} \right), \qquad (20)$$

where the sum is taken with respect to randomly selected minibatch with size T_b.

Algorithm 3: Decoupled learning strategy for multi-agent DRL training

input : Action space $\mathcal{A}_{\mathrm{ACB}}$, $\mathcal{A}_{\mathrm{BO}}$, $\mathcal{A}_{\mathrm{DQ}}$, and operation iteration I.

1   Algorithm hyperparameters: learning rate $\lambda_{\mathrm{RNN}} \in (0, 1]$ for traffic predictor, and learning rate $\lambda_{\mathrm{DRL}} \in (0, 1]$, as well as discount rate $\gamma \in [0, 1)$ for DRL agents; Initialization of the RNN parameters $\boldsymbol{\theta}_{\mathrm{RNN}}$ for traffic predictor, the action-state value function $Q(s, a; \boldsymbol{\theta}_{\mathrm{BO}})$ for the BO scheme, the action-state value function $Q(s, a; \boldsymbol{\theta}_{\mathrm{DQ}})$ for the DQ scheme, and the parameterized actor $\pi(s; \boldsymbol{\theta}_{\mathrm{DDPG}})$ as well as critic $v(s, a; \boldsymbol{w}_{\mathrm{ACB}})$ for the ACB scheme;

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

2   Initialization of O^0 by executing a random action;

3   **for** $t \leftarrow 0$ **to** $T - 1$ **do**

4   Predict backlog value $\hat{N}^t$ using $\boldsymbol{\theta}_{\mathrm{RNN}}^t$ according to the historical observations $[O^{t-T_o+1}, \cdots, O^{t-2}, O^{t-1}]$;

5   Merge the predicted value $\hat{N}^t$ in the belief state vector S_{blf}^t;

6   **ACB**: Select $A_{\mathrm{ACB}}^t = \pi(S^t; \boldsymbol{\theta}_{\mathrm{ACB}}) + \mathcal{N}^t$;

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

8   **else** Select $A_{\mathrm{BO}}^t = \underset{a \in \mathcal{A}_{\mathrm{BO}}}{\arg\max} \, Q(S^t, a; \boldsymbol{\theta}_{\mathrm{BO}})$;

9   **DQ: if** $p_\epsilon^t < \epsilon$ **then** select a random action A_{DQ}^t from $\mathcal{A}_{\mathrm{DQ}}$;

10   **else** Select $A_{\mathrm{DQ}}^t = \underset{a \in \mathcal{A}_{\mathrm{DQ}}}{\arg\max} \, Q(S^t, a; \boldsymbol{\theta}_{\mathrm{DQ}})$;

11   BS broadcasts action $A^t = \{A_{\mathrm{ACB}}^t, A_{\mathrm{BO}}^t, A_{\mathrm{DQ}}^t\}$, and backlogged IoT devices execute RACH;

12   BS observes S^{t+1}, and estimate backlog value $\tilde{N}^t$ as well as calculates $R^t = V_s^{t+1}$;

13   Storing transition $(S^t, \tilde{N}^t)$ in the replay memory of RNN traffic predictor;

14   Storing each transition $(S^t, A_{\mathrm{ACB}}^t, R^t, S^{t+1})$, $(S^t, A_{\mathrm{BO}}^t, R^t, S^{t+1})$, and $(S^t, A_{\mathrm{DQ}}^t, R^t, S^{t+1})$ in each DRL replay memory M_{ACB}, M_{BO}, and M_{DQ}, respectively;

15   Sampling random minibatch of transitions from each replay memory;

16   Calculate the loss of RNN $L^t(\boldsymbol{\theta}_{\mathrm{RNN}})$ using Eq. (20), the loss of Q-function for the BO and the DQ schemes $\nabla L(\boldsymbol{\theta}_{\mathrm{DQN}}^t)$ using Eq. (15), and the loss of critic $\nabla L(\boldsymbol{w}_{\mathrm{ACB}}^t)$ and actor $\nabla L(\boldsymbol{\theta}_{\mathrm{ACB}}^t)$ for the ACB scheme using Eq. (16) and Eq. (17), respectively;

17   Perform a gradient descent for the RNN predictor and each primary DRL network;

18   Update the target DRL networks using: $\bar{\boldsymbol{w}}^t \leftarrow \sigma \boldsymbol{w}^t + (1-\sigma)\bar{\boldsymbol{w}}^t$, and $\bar{\boldsymbol{\theta}}^t \leftarrow \sigma \boldsymbol{\theta}^t + (1-\sigma)\bar{\boldsymbol{\theta}}^t$.

19   **end**

20 **end**

---

2) DRL-based Parameter Configuration: As seen in Fig. 5 (b), the newly predicted traffic value is input into the DRL agents along with the historical traffic values, where

TABLE II: RL Hyperparameters

Parameters	Value	Parameters	Value
Synchronization time T_{sy}	0.65 s	Msg1 transmission time T_{Msg1}	0.084 s
Msg2 receiving time T_{Msg2}	0.345 s	Msg3 transmission time T_{Msg3}	0.08 s
Msg4 receiving time T_{Msg4}	0.345 s	Transmit power $P_{\text{Msg1}}, P_{\text{Msg3}}$	0.545 W
Receive power $P_{\text{sy}}, P_{\text{Msg2}}, P_{\text{Msg4}}$	0.09 W		

(a) The average success accesses

(b) The average energy consumption

(c) The average access delay
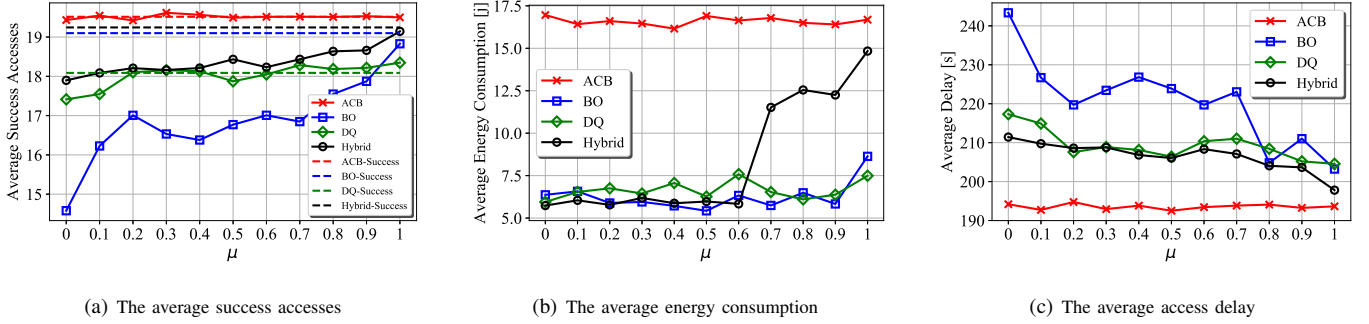
Fig. 6: The (a) average success accesses, (b) average energy consumption, and (c) average access delay per IoT device per episode.

this vector of backlog values is treated as the approximate belief state of the DRL agents. Recall that the historical length T_o is manually selected according to the expected memory for time-correlation recognition. To do so, the belief state variable of each agent can be written as $S_{\text{blf}}^t = [A^{t-T_o}, \hat{N}^{t-T_o+1}, A^{t-T_o+1}, \hat{N}^{t-T_o+2}, ..., A^{t-1}, \hat{N}^t]$. Similar as the training of multi-agent DRL given in Sec. IV-A, every DRL agent shares their historical action selection $A = \{A_{\text{ACB}}, A_{\text{BO}}, A_{\text{DQ}}\}$ to enable cooperation. Different from Sec. IV-A, not only the DRL agents are trained, but also the RNN-based traffic predictor is updated after each frame. In the following, the implementation of decoupled learning strategy for multi-agent DRL training is shown in **Algorithm 3**.

In the training process, the evolution of RNN traffic predictor and the DRL agents are mutual correlated, thus they need to be trained in parallel. This parallel adaptation can be implemented in either simulation or practice, while the former case enables a pre-training to ease practical implementation, and the latter case allows all agents to adapt to the realistic traffics. Specifically, the weights of DRL agents are able to be initialized by first training in offline experiments based on available traffic models, which would considerably reduce the time and computational resource needed for their convergence of training in practice. However, this assumed traffic models may mismatch with the practical traffic statistics, thus the adaptation in practice is still necessary. This pre-training can be treated as a case of meta-learning. An numerical example will be given in the next subsection.

C. Numerical Results and Evaluation

In this subsection, numerical experiments are conducted to evaluate the performance of the hybrid ACB, BO, and DQ schemes in terms of the access success devices number, the access delay, and the energy consumption. Particularly, the following figures represent results of these three KPI taking into account all IoT devices whatever success or fail in access, rather than that in the proposed algorithms, where only the access success IoT devices can be observed in the

BS. The calculation of access delay and energy consumption is based on the functions given in Eq. (6) and Eq. (7), and the related parameters are listed in Table II provided by 3GPP for cellular mIoT systems [15]. We adopt the similar traffic profile, network parameters, and hyperparameters setting of neural networks as Sec. III-C. Importantly, hybrid scheme integrates the features of the ACB, the BO, and the DQ schemes, where they may be jointly executed according to control of the DRL agents.

We start by considering the reward weights are defined by $x_s : x_d : x_e = 1 : \mu : 1 - \mu$, where $\mu \in [0, 1]$. By selecting different value of μ, the DRL agents are able to optimize the system with different priorities between energy and delay. In the following, we plot the average number of access success devices, the average energy consumption, and the average access delay per IoT device per episode of each scheme versus the priority value μ in Fig. 6 (a), Fig. 6 (b), and Fig. 6 (c), respectively. In Fig. 6 (a), "ACB-Success", "BO-Success", "DQ-Success", and "Hybrid-Success" with dashed lines represent these four schemes trained by using the reward only considering the number of access success devices (reward weights $x_s : x_d : x_e = 1 : 0 : 0$). We observe that the average number of access success devices of the ACB scheme is always longer than that of the other schemes, due to that the ACB scheme can directly determine the access request probability of each IoT device in each frame. Except from the ACB scheme, the performance of the other schemes increase with the increase of μ, due to that the access delay is inversely proportional to the number of access success devices. It can be seen that, with the increase of the delay weight (μ), the average energy consumption of the BO, DQ, and hybrid schemes increases in Fig. 6 (b), whereas the average access delay of these schemes decreases in Fig. 6 (c). This shows that the energy-delay trade-off of the network system can be flexibly balanced by selecting different weights in the proposed hybrid reward. When μ is smaller than 0.3, we observe that the proposed hybrid scheme sacrifices the delay property, and consumes the minimum energy. We further observe that, with the

increase of μ, the average access delay of the proposed hybrid scheme gradually closes to that of the optimal ACB scheme, and it always outperforms the BO and DQ schemes. This demonstrates the capability of the proposed hybrid scheme to flexibly adapt to different performance requirements of the network.
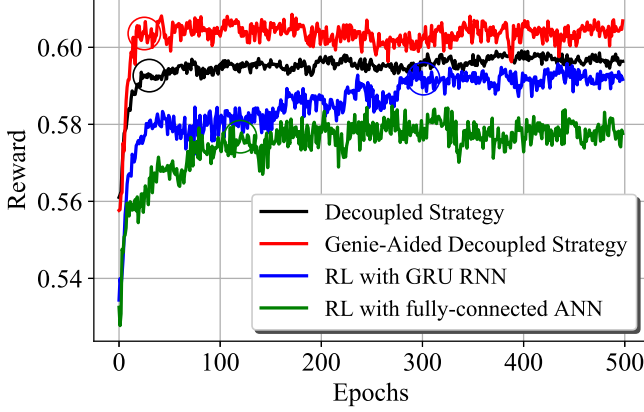

Fig. 7: Average reward per frame as a function of each epoch in the online adaptation phase

Fig. 7 plots the evolution (averaged over 100 training trails) of the average reward per frame as a function in the online phase for the proposed decoupled strategy, genie-aided decoupled strategy, as well as the conventional RL methods with function approximators using GRU RNN and fully-connected Artificial Neural Network (ANN). The proposed decoupled strategy is conducted according to Algorithm 3, while genie-aided decoupled strategy uses the same algorithm, except that the training is based on the criterion Eq. (20) with the ideal label N^t in lieu of the estimated label $\tilde{N}^t$. The conventional RL methods are based on the method given in Sec. III-B and Sec. IV-A, and specifically, the one without RNN uses fully-connected neural network with two hidden layers, each with 128 ReLU units. For fairly considering both energy consumption and access delay, the reward weights are defined by $x_s : x_d : x_e = 2 : 1 : 1$ ($\mu = 0.5$). The approximated converging point of each scheme is highlighted by circles. We observe that the conventional RL without RNN has the worst performance, due to that its fully-connected neural network cannot capture the time correlation conducted by time-varied traffics and devices' queuing processes. This showcases the necessity of using RNN in RACH procedure optimization.

We then observe that our proposed decoupled strategy can quickly adapt to the network conditions, where its training speed is substantially faster than that of RL with RNN, and its obtained average reward is also higher than that of RL with RNN. Note that the performance of training efficiency may vary for different simulation parameters. This is due to the fact that the proposed decoupled strategy knows that the historical and present traffic statistics are directly correlated with the future performance, while the conventional RL methods need to learn this inherent correlation by experiencing an exploration learning process. Apparently, these extra explorations can

consume increased training time. It is also seen that both these performance of the decoupled strategy is very close to that of the ideal genie-aided one. This demonstrates that the proposed decoupled strategy is capable to optimize the RACH schemes with better performance and faster converging speed.
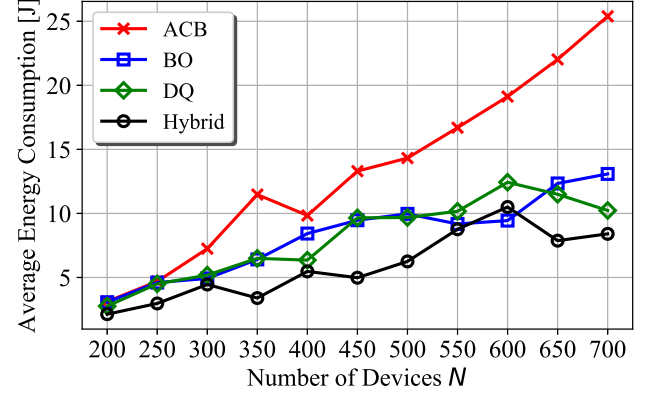
(a) Average energy consumption
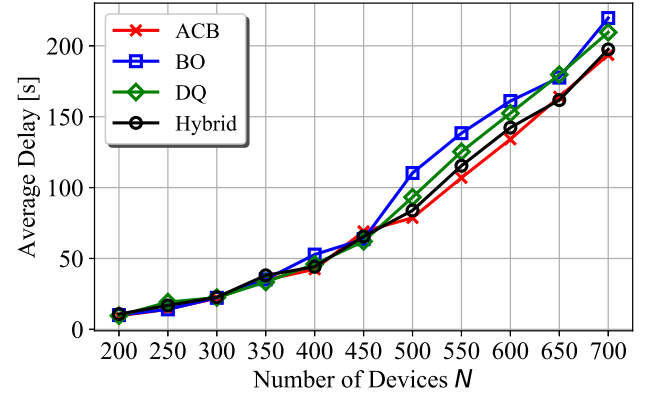
(b) Average access delay

Fig. 8: Performance of decoupled learning strategy in terms of (a) average energy consumption, and (b) Average access delay per IoT device per episode.

We finally illustrates the average energy consumption and the average access delay per IoT device per episode of each scheme versus the presence of devices N in Fig. 8 (a) and Fig. 8 (b), respectively. All schemes are implemented by using the decoupled learning strategy proposed in Sec. IV-B. Note that the results of each point are obtained by evaluating a scheme with its learning agents that are trained after 10^5 episodes. In particular, the DRL agents for each scheme used in Fig. 8 (a) and Fig. 8 (b) are optimized by using the reward weights with $x_s : x_d : x_e = 1 : 0 : 1$ and $x_s : x_d : x_e = 1 : 1 : 0$ in Eq. (2), respectively. More specifically, the learning agents in Fig. 8 (a) aim at saving the energy of IoT device while optimizing the number of access success IoT devices with the reward of energy consumption as $x_e = 1$, while the learning agents in Fig. 8 (b) aim at reducing the access delay while optimizing the number of access success IoT devices with the reward of access delay as $x_d = 1$. In both Fig. 8 (a) and Fig. 8 (b), it can

be seen that increasing the number of IoT devices N increases both the average energy consumption and the average access delay of all schemes, due to that the increase of traffic statistics leads to more collisions.

In Fig. 8 (a), the performance of the ACB scheme is notably worse than the other schemes, due to that IoT devices tends to waste more energy when they repeatedly listen to the ACB factor to ask access permissions. In Fig. 8 (b), the ACB scheme slightly outperforms the BO and the DQ schemes, when the number of devices N is bigger than 450. This is due to that the ACB scheme can accurately set the access request probability of each device at every frame, while the BO and the DQ scheme can only schedule re-transmissions into relatively long future frames. Due to this feature, the ACB scheme is more capable in accurately alleviating the overloaded traffic to reduce access delay, but it also sacrifices the energy consumption as shown in Fig. 8 (a). It is also observed that the hybrid scheme always outperforms the other single schemes in terms of energy saving in Fig. 8 (a), while it is close to, but is no better than, the ACB scheme in terms of delay reduction in Fig. 8 (b). The former phenomenon is due to that the hybrid scheme is capable of adjusting to the complex energy saving requirement by optimally balancing all three schemes, while the latter phenomenon is due to that the hybrid scheme is learned to optimize the performance by mostly relying on the ACB control, which is the best method to reduce access delay.

D. A Case Study of NarrowBand IoT Networks

To show the effectiveness of our proposed decoupled strategy, we now consider a more practical NarrowBand (NB)-IoT scenario according to the system model provided in our prior work [13, 14], in which an NB-IoT network composed of an evolved Node B (eNB) and $N = 30000$ static IoT devices with time-limited Beta traffic profile. The eNB supports three Coverage Enhancement (CE) groups to provide access for IoT devices with different location. Each IoT device determines their CE identity according to the their distance to the associated eNB. After that, each IoT device executes RACH as well as uplink data transmission according to the received system information that relates to their CE identity. More simulation details can be found in our prior work [13, 14], and the 3GPP reports [15, 16].

Different from the results in Sec. IV-C independently focusing on RACH procedure, this model considers an IoT device that is successfully served only when it succeeds in both RACH and uplink data transmission. In each frame, the eNB allocates the radio resources to accommodate the RACH procedure for each CE group with the remaining resources used for uplink data transmission. Considering the target of maximizing the number of served IoT devices, the challenge is to optimally balance the allocations of channel resources between the RACH procedure and data transmission, as well as among each CE group. Similar as [14], we assume the eNB can flexibly select the parameters of the number of RACH periods $n_{\text{Rach},i}^t$, the number of available preambles $f_{\text{Prea},i}^t$, and
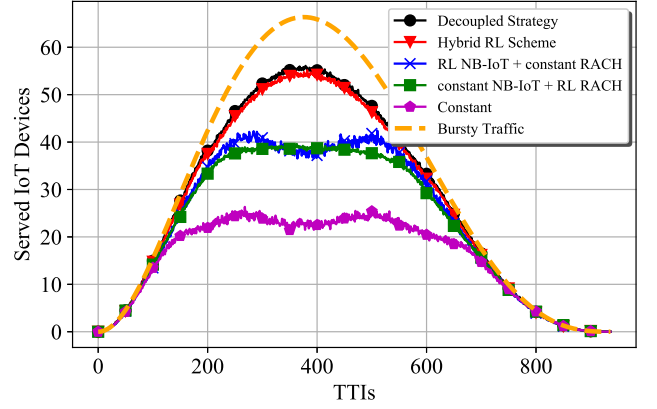

Fig. 9: The average number of successfully served IoT devices per frame during one bursty traffic duration. The dashed line represents the average number of generated packets per frame.

the repetition value $n_{\text{Repe},i}^t$ in each group i at each frame t. Extended from [14], we further consider that the eNB can flexibly select the ABC and the BO factors to alleviate traffic overload. The DQ scheme is not standardized in NB-IoT networks [16], so as it is not used in this case.

Fig. 9 compares the number of successfully served IoT devices per frame during one epoch. The result of each curve is averaged over 1000 testing epochs. The average number of newly generated packets conducted by the time limited Beta profile is shown as dashed line. We compare the performance in terms of the average number of successfully served IoT devices among the following five schemes: 1) "Decoupled Strategy" proposed in Sec. IV-B; 2) "hybrid RL scheme" given in IV-A; 3) "RL NB-IoT + constant RACH", which uses hybrid RL scheme to configure NB-IoT factors $\{n_{\text{Rach},i}^t, f_{\text{Prea},i}^t, n_{\text{Repe},i}^t\}$ as well as sets constant BO and ACB factors $\{f_{\text{ACB}}^t, f_{\text{BO}}^t\} = \{0.5, 2\}$ for each CE group; 4) "Constant NB-IoT + RL RACH", which sets NB-IoT factors $\{[n_{\text{Rach},1}^t, n_{\text{Rach},2}^t, n_{\text{Rach},3}^t], [f_{\text{Prea},1}^t, f_{\text{Prea},2}^t, f_{\text{Prea},3}^t], [n_{\text{Repe},1}^t, n_{\text{Repe},2}^t, n_{\text{Repe},3}^t]\} = \{[4, 2, 1], [48, 36, 24], [2, 16, 32]\}$ as well as configures BO and ACB factors by using hybrid RL scheme; and 5) "Constant" scheme sets constant parameters during whole epoch including NB-IoT factors $\{[n_{\text{Rach},1}^t, n_{\text{Rach},2}^t, n_{\text{Rach},3}^t], [f_{\text{Prea},1}^t, f_{\text{Prea},2}^t, f_{\text{Prea},3}^t], [n_{\text{Repe},1}^t, n_{\text{Repe},2}^t, n_{\text{Repe},3}^t]\} = \{[4, 2, 1], [48, 36, 24], [2, 16, 32]\}$ as well as BO and ACB factors $\{f_{\text{ACB}}^t, f_{\text{BO}}^t\} = \{0.5, 2\}$ for each CE group.

In Fig. 9, we observe that the number of served IoT devices of all schemes are close in the light traffic regions at the beginning and end of the epoch, however, in the period of heavy traffic in the middle of the epoch, that number of served IoT devices follows the order "Decoupled Strategy"≈'hybrid RL scheme">"Constant NB-IoT + RL RACH"≈"RL NB-IoT + constant RACH">"Constant". This showcases that the more dynamically configured system factors are, the better performance can be achieved. This is due to that the constant factors cannot fit all the traffic statistics, while RL algorithms can always find a global optimal combination of the system
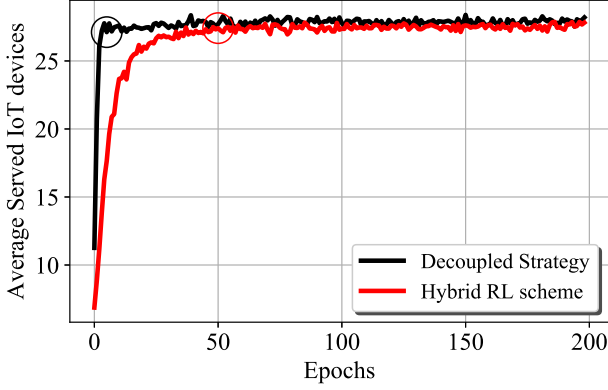
Fig. 10: The average number of successfully served IoT devices per epoch during training.

parameters to adapt to any online traffic statistics. It is also observed that "Decoupled Strategy" slightly outperforms 'hybrid RL scheme", which demonstrates the capability of the proposed decoupled strategy can better manage the channel resources, and achieve better system performance in the presence of heavy traffic. Fig. 10 plots the evolution (averaged over 20 training trails) of the average number of served IoT devices per epoch as a function during the online training for "Decoupled Strategy" and "hybrid RL scheme". We observe that our proposed decoupled strategy can quickly adapt to the network conditions, which is about 10 times faster than the conventional hybrid RL scheme.

V. Conclusion

In this paper, we developed a decoupled learning strategy to maximize a multi-objective function that is composed of the number of access success devices, the average energy consumption, and the average access delay. The proposed algorithm jointly optimized multiple RACH schemes, including the ACB, BO, and DQ schemes. Our proposed strategy decoupled the traffic prediction and the parameter configuration. The former predictor uses a GRU RNN model to predict the real-time traffic values of the network system, which captures temporal correlations due to communication mechanisms and irregular traffic generation. The latter controller configures system parameters of each RACH scheme by using multiple DRL agents, where DQN is used to handle discrete action selection for the BO as well as the DQ schemes, and DDPG is used to handle continuous action selection for the ACB scheme. Numerical results shed light on that the RACH schemes can be effectively optimized in a joint manner by using the cooperative training to adapt to any performance requirement, where it outperforms each single RACH scheme.
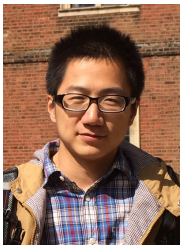
More importantly, by using the proposed decoupled learning strategy, the training speed of the cooperative model considerably outperforms conventional strategies. This result gives clear evidence that integrating predicted traffic into a learning process would benefit both its training convergence and training efficiency. The proposed method can be applied for optimizing any RACH schemes in the 5G NR network,

and can be extended to solve the similar dynamic optimization problems, e.g. quality of service mapping. As most of the current schemes on RACH procedure only relies on the central control in the BS, a promising future direction is to develop the learning algorithm that performs the RACH control in a cooperative manner between IoT devices and BS.
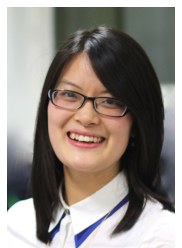
References

[1] N. Jiang, Y. Deng, and A. Nallanathan, "Deep reinforcement learning for discrete and continuous massive access control optimization," *to appear in IEEE Int. Conf. Commun.*, Jun. 2020.

[2] H. Wu, C. Zhu, R. J. La, X. Liu, and Y. Zhang, "Fast adaptive S-ALOHA scheme for event-driven machine-to-machine communications," in *IEEE Veh. Technol. Conf. (VTC Fall)*, Sep. 2012, pp. 1–5.

[3] 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.

[4] H. He, P. Ren, Q. Du, L. Sun, and Y. Wang, "Traffic-aware overload control scheme in 5G ultra-dense M2M networks," *Trans. Emerging Telecommun. Technol.*, vol. 28, no. 9, p. 3146, Sep. 2017.

[5] X. Jian, Y. Jia, X. Zeng, and J. Yang, "A novel class-dependent back-off scheme for machine type communication in lte systems," in *IEEE Wireless Opt. Commun. Conf.*, May 2013, pp. 135–140.

[6] F. Vázquez-Gallego, C. Kalalas, L. Alonso, and J. Alonso-Zarate, "Contention tree-based access for wireless machine-to-machine networks with energy harvesting," *IEEE Trans. Green Commun. Netw.*, vol. 1, no. 2, pp. 223–234, Jun. 2017.

[7] 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.

[8] L. Tello-Oquendo, V. Pla, I. Leyva-Mayorga, J. Martinez-Bauset, V. Casares-Giner, and L. Guijarro, "Efficient random access channel evaluation and load estimation in LTE-A with massive MTC," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1998–2002, Dec. 2018.

[9] J.-R. Vidal, L. Tello-Oquendo, V. Pla, and L. Guijarro, "Performance study and enhancement of access barring for massive machine-type communications," *IEEE Access*, vol. 7, pp. 63 745–63 759, May 2019.

[10] H. Bello, X. Jian, Y. Wei, and M. Chen, "Energy-delay evaluation and optimization for NB-IoT PSM with periodic uplink reporting," *IEEE Access*, vol. 7, pp. 3074–3081, Dec. 2018.

[11] A. Azari, Č. Stefanović, P. Popovski, and C. Cavdar, "On the latency-energy performance of nb-iot systems in providing wide-area iot connectivity," *IEEE Trans. Green Commun. Netw.*, 2019.

[12] P. K. Wali and D. Das, "Optimization of barring factor enabled extended access barring for energy efficiency in LTE-Advanced base station," *IEEE Trans. Green Commun. Netw.*, vol. 2, no. 3, pp. 830–843, Sep. 2018.

[13] 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*, May 2019.

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

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

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

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

[18] N. Jiang, Y. Deng, X. Kang, and A. Nallanathan, "Random access analysis for massive IoT networks under a new spatio-temporal model: A stochastic geometry approach," *IEEE Trans. Commun.*, vol. 66, no. 11, pp. 5788–5803, 2018.

[19] 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.

[20] 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.

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

[22] Z. Gábor, Z. Kalmár, and C. Szepesvári, "Multi-criteria reinforcement learning," in *Int. Conf. Machine Learning*, vol. 98, 1998, pp. 197–205.

[23] 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.

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

[25] G.-Y. Lin, S.-R. Chang, and H.-Y. Wei, "Estimation and adaptation for bursty LTE random access," *IEEE Trans. Veh. Technol.*, vol. 65, no. 4, pp. 2560–2577, Apr. 2016.

[26] F. Schoute, "Dynamic frame length ALOHA," *IEEE Trans. commun.*, vol. 31, no. 4, pp. 565–568, Apr. 1983.

[27] N. Jiang, Y. Deng, O. Simeone, and A. Nallanathan, "Online supervised learning for traffic load prediction in framed-ALOHA networks," *IEEE Commun. Lett.*, 2019.

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

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

[30] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[31] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[32] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Int. Conf. Learning Representations*, 2016.

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 Artificial Intelligence for Wireless Systems, Beyond 5G Wireless Networks, Internet of Things (IoT) and Molecular Communications. He published nearly 500 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 and an AI 2000 Internet of Things Most Influential Scholar in 2020.

He is an Editor-at-Large for IEEE Transactions on Communications and Senior Editor for IEEE Wireless Communications Letters. He was an Editor for IEEE Transactions on Wireless Communications (2006-2011), IEEE Transactions on Vehicular Technology (2006-2017) 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.

Nan Jiang (S'16-M'20) is currently a research engineer with the Telecommunications Research Laboratory, Toshiba Research Europe Ltd., Bristol, U.K. He received the Ph.D. degree in electronic engineering from the Queen Mary University of London, U.K., in 2020. He was a visiting researcher with the King's College London, U.K., in 2016 and 2018. His research interests include internet of things, machine learning, time-sensitive networks, and 5G wireless networks. He received the IEEE ComSoc student travel grant in IEEE ICC'18 and ICC'20. He has also served as a TPC Member for IEEE VTC'19 AND VTC'20.

Jinhong Yuan (M'02-SM'11-F'16) received the B.E. and Ph.D. degrees in electronics engineering from the Beijing Institute of Technology, Beijing, China, in 1991 and 1997, respectively. From 1997 to 1999, he was a Research Fellow with the School of Electrical Engineering, University of Sydney, Sydney, Australia. In 2000, he joined the School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, Australia, where he is currently a Professor and Head of Telecommunication Group with the School. He has published two books, five book chapters, over 300 papers in telecommunications journals and conference proceedings, and 50 industrial reports. He is a co-inventor of one patent on MIMO systems and two patents on low-density-parity-check codes. He has co-authored four Best Paper Awards and one Best Poster Award, including the Best Paper Award from the IEEE International Conference on Communications, Kansas City, USA, in 2018, the Best Paper Award from IEEE Wireless Communications and Networking Conference, Cancun, Mexico, in 2011, and the Best Paper Award from the IEEE International Symposium on Wireless Communications Systems, Trondheim, Norway, in 2007. He is an IEEE Fellow and currently serving as an Associate Editor for the IEEE Transactions on Wireless Communications. He served as the IEEE NSW Chapter Chair of Joint Communications/Signal Processions/Ocean Engineering Chapter during 2011-2014 and served as an Associate Editor for the IEEE Transactions on Communications during 2012-2017. His current research interests include error control coding and information theory, communication theory, and wireless communications.

Yansha Deng (S'13-M'16) received the Ph.D. degree in electrical engineering from the Queen Mary University of London, U.K., in 2015. From 2015 to 2017, she was a Post-Doctoral Research Fellow with Kings College London, U.K., where she is currently a Lecturer (Assistant Professor) with the Department of Engineering. Her research interests include molecular communication, machine learning, 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 Associate Editor of the IEEE Transactions on Communications, IEEE Transactions on Molecular, Biological and Multi-scale Communications, and the Senior Editor of the IEEE Communication Letters. She also received the Exemplary Reviewers of the IEEE Transactions on Communications in 2016 and 2017, and IEEE Transactions on Wireless Communications in 2018. She has also served as a TPC Member for many IEEE conferences, such as IEEE GLOBECOM and ICC.