Designing Medium Access Control Protocol Sequences Through Deep Reinforcement Learning

Cédric Adjih, Chung Shue Chen, Chetanveer Gobin, Iman Hmedoush

Designing Medium Access Control Protocol Sequences Through Deep Reinforcement Learning

Cedric Adjih*, Chung Shue Chen†, Chetanveer Sharma Gobin‡, and Iman Hmedoush*§
*Inria Saclay, Rue Honore d'Estienne d'Orves, 91120 Palaiseau, France
‡Institut National des Sciences Appliquées (INSA) de Lyon, 69621 Villeurbanne, France
Nokia Bell Labs† & Standards Research§, 12 Rue Jean Bart, 91300 Massy, France
Email: cedric.adjih@inria.fr, chung_shue.chen@nokia-bell-labs.com, chetanveer.gobin@insa-lyon.fr, iman.hmedoush@nokia.com

***Abstract*—This work aims to design protocol sequences through deep reinforcement learning (DRL). Protocol sequences are periodic binary sequences that define multiple access control among users, introduced for systems considering collision channel without feedback (CCw/oFB). In this paper, we leverage the recent advancement of DRL methods to design protocol sequences with desirable new properties, namely Throughput Maximizing User-Irrepressible (TMUI) sequences. TMUI has two specific properties: (i) user-irrepressibility (UI), and (ii) maximizing the minimum individual throughput among the users. We assumed that the transmission channel is divided into time slots and the starting time of each user in joining the system is arbitrary such that there exist random relative time offsets. We use a DRL approach to find TMUI sequences. We report the obtained TMUI protocol sequences and conduct numerical studies comparing TMUI against slotted ALOHA. Simulation results also show that the new medium access control (MAC) protocol does hold the UI property and can achieve substantially higher minimum individual user throughput, under the same system parameters.**

***Index Terms*—MAC, deep reinforcement learning (DRL), collision channel without feedback, protocol sequence, TMUI.**

I. Introduction

The Internet of Things (IoT) aims to connect millions of devices, most of which are power and memory constrained. Such constraints require efficient network access and very low complexity multiple access protocols. One of the main challenges to IoT implementation is to have robust wireless multiple access and connectivity.

Bandwidth and throughput, compatibility, scalability, and security are the major areas one should think about before IoT deployment. For thin or power-constrained devices, it is often favorable to have a simple multiple access protocol which for example does not require frequent monitoring of the channel for feedback or acknowledgments. It is also favorable to have a simple protocol that does not require complicated processing such as a back-off algorithm or random number generation, as in the case of Wi-Fi networks.

Besides, when considering a wireless network with dynamic network topology, due to user mobility or time-varying radio signal propagation delays, transmission synchronization among users could be very complicated. This would lead to unknown time offsets of different user transmissions. As a result, collisions will occur. In some cases, such as when the communication requires low latency (for Ultra Reliable Low Latency Communications (URLLC)), or when the link in the reverse direction would be too limited, or would be unreliable, for instance, if it is subject to random access, one might avoid relying on feedback. In that case, one can consider the collision channel without feedback (CCw/oFB) model invented by J. Massey [1] which is a natural fit.

There is a recent revisit of the above model due to mobile ad hoc networks and wireless IoT and new development of protocol sequences for multiple access control [2]–[4]. The shared time channel is divided into time slots of fixed equal duration. Each user is assigned a deterministic zero-one binary sequence, known as *protocol sequence*, which governs its channel access. The ones in a protocol sequence represent the slot where the user would transmit, while the zeros represent idle slots. A *collision* occurs if two or more users transmit in the same time slot. In this paper, we will follow the above considerations and focus on the discovery of suitable protocol sequences with specific user quality-of-service (QoS) criteria for medium access control.

Traditional protocol sequence constructions require, most of the time, complex mathematical tools including number theory, field theory, and, combinatorics. Besides, sometimes it is difficult to model some user QoS criteria using classical mathematical tools and then derive a corresponding sequence generating function. In addition, sometimes an analytical sequence construction method is only available for some specific system parameters or has some limitations, for example, the number of supported users is a prime number. In the meantime, there are new advancements in machine learning for solving traditionally hard problems. The above reasons have motivated us to consider recent deep learning methods for designing protocol sequences. We decided to adopt Deep Reinforcement Learning (DRL). Loosely speaking, DRL is a family of machine learning where an agent takes decisions with a Deep Neural Network (DNN) model and learns how to behave efficiently in an environment by performing actions and collecting rewards. This method is appropriate in the case of protocol sequence design since it allows unsupervised learning and we do not have a labeled dataset of protocol sequences. Also, we do not want to rely on an exhaustive search. DRL can solve complex tasks and has been widely adopted in recent MAC protocol designs [5].

Authors are listed in alphabetical order. The work and experimental implementation was conducted by C. Gobin during his internship at INRIA/Nokia.

In this paper, we present a new DRL based protocol sequence construction scheme. We leverage the recent advancement of DRL methods to design protocol sequences with the following two specific properties: (i) user-irrepressibility (UI), and (ii) maximizing the minimum individual throughput among the users, namely Throughput Maximizing User-Irrepressible (TMUI) sequences. In this paper, we present our proposed scheme, report the obtained TMUI sequences and conduct a numerical study to compare their performance against slotted ALOHA under CCw/oFB model. Simulation results show that the obtained TMUI sequences do hold the UI property and achieve significantly higher minimum individual user throughput among the users than that of slotted ALOHA. The main contributions of the paper are summarized below.

- We propose a new method to design MAC protocol sequences using DRL.
- We show that it is possible to determine protocol sequences with interesting properties such as TMUI and arbitrary length. The latter is a significant relaxation from sequence construction methods that rely on rigid mathematical structures.
- We construct protocol sequences for MAC which can achieve a significantly higher minimum individual throughput and outperform traditional ALOHA under CCw/oFB setting.

The rest of the paper is organized as follows. In Section II, we present the state of the art. In Section III, we describe the system model. In Section IV, we present our deployed DRL scheme for finding TMUI sequences. In Section V, we present the numerical results and comparisons. Finally, we conclude the paper in Section VI.

II. Related Work

In the current literature, there are two approaches for constructing protocol sequences. The first approach is to use traditional methods of combinatorics, coding theory, and number theory, such as in [6], [7]. The second approach, which is more recent, is to use machine learning or deep learning methods, which have been applied in many areas including human genetics research, robotics, telecommunications, and many other areas. Since we focus on protocol sequences, in the following, we present machine learning-based sequence construction techniques and some related work.

In [8], the authors proposed a new model called AlphaSeq for sequence discovery. The method is inspired by AlphaGo developed by DeepMind and is used to construct desired sequences using DRL techniques. AlphaSeq treats the sequence discovery problem as an episodic symbol-filling game, in which a player fills symbols in the vacant positions of a sequence set sequentially during each episode of the game. The episode ends with a completely filled sequence set, upon which a reward is returned based on the desirability of the existing sequence set. AlphaSeq models the above game as a Markov Decision Process (MDP) and adapts the DRL framework of AlphaGo to solve the MDP. Compared with traditional sequence construction through mathematical tools, AlphaSeq is particularly suitable for problems with complex objectives intractable to mathematical analysis, For example, AlphaSeq is shown effective in finding the zero-forcing complementary codes for code-division multiple access (CDMA) systems and discovering new sequences that outperform existing Legendre sequences for mismatched filter (MMF) estimator.

Another work is the design of deterministic grant-free medium access control (MAC) using DRL [9], which aims to design interference-canceling (IC) codes for multiple users under physical layer successive interference cancellation (SIC). These codes find their applications in Ultra-Reliable Low-Latency Communications (URLLC) for 5G. Traditional search algorithms cannot be used to derive IC codes since the search space is too big. Also, the codewords rely on special mathematical properties among them, which make their construction difficult. They use a DRL-based algorithm to search IC codes, with carefully designed metrics and reward functions as per the underlying mathematical constraints. Simulation results show that the algorithm can discover IC codes that yield significantly lower failure probability than a random access protocol under the same latency requirements, and is very suitable for URLLC.

In [10], a reinforcement learning (RL) based scheme is used to design error correction codes (ECC). A constructor-evaluator framework is proposed in which the code constructor can be realized by using various machine learning algorithms, while the code evaluator provides code performance metric measurements for each evolution. The authors used the Advantage Actor Critic (A2C) for the constructor. The code constructor keeps improving the code construction to maximize the code performance, which is evaluated by the code evaluator until the performance metric converges. Simulation results show that comparable code performance can be achieved for the existing codes and the method can provide superior performance in comparison to classical constructions in certain cases, for example, when being applied for decoding polar codes.

Another interesting approach is in [11], which formulates the sequence discovery problem as a maze-traversing game. The idea is to use RL techniques to construct polar codes for Successive Cancellation List (SCL) decoder. The tabular RL algorithm SARSA(λ) [12] is used to solve efficiently the game. Simulation results show that the game-based constructions can match today's polar code constructions for SCL decoding and even outperform the standard constructions [11]. Moreover, that method has very efficient training in terms of the number of required training samples.

III. System Model

In this section, we describe the system model, and we introduce the targeted TMUI protocol sequences with their related characteristics.

Consider a system of M users, sharing the same communication channel, transmitting to a single receiver. Each user i will be attributed one unique and periodic protocol sequence of length L. Each sequence is a deterministic zero-one pattern that indicates when the user should transmit or not. We denote the sequence of user i at time t, by $s_i(t)$.

A. Collision Channel Without Feedback

Under the model of CCw/oFB [1], due to the lack of feedback channel for coordination of transmissions, it is considered that users start transmitting at arbitrary times, so that there are unknown starting time delay offsets among the users. It is worth noting that in such a system, one can consider either a time slotted channel or a completely asynchronous channel, among the users. In the former, users are assumed to know the slot boundaries in the time slotted channel. However, in the latter case, users can transmit at a completely arbitrary time instants. In this paper, for the sake of simplicity, we will adopt the time slotted channel, which is implementable by broadcasting a simple beacon signal for the users to identify the slot boundaries [7], whereas the time is divided into time slots of equal duration.

B. Protocol Sequence

Let $\mathcal{S}$ denote a set of binary $\{0,1\}$ protocol sequences, which consists of M sequences and each is of length equal to L, i.e., $\mathcal{S} \triangleq \{s_0, s_1, \ldots, s_{M-1}\}$, where $s_i \triangleq (s_i(0), s_i(1), \ldots, s_i(L-1))$, for $i = 0, 1, 2, \ldots, M-1$. A user i will use its sequence s_i to transmit at time slot t if and only if $s_i(t) = 1$, for $t \in [0, L-1]$. When $s_i(t) = 0$, user i keeps silent.

Definition 1: The *duty factor* of a sequence is the number of ones (i.e., the Hamming weight) divided by its period L, which measures the fraction of time a user is transmitting.

For example, a duty factor of $\frac{1}{2}$ means that the number of transmissions of a user equal to $\frac{L}{2}$ during the sequence period L. In this work, we aim to leverage the recent advancement of DRL methods to discover protocol sequences with the following two properties under CCw/oFB model, called TMUI:

1) The sequence set is user-irrepressible (UI), (see [13]).
2) The minimum individual throughput among the users in the system under arbitrary time delay offsets is maximized.

We explain the above two properties below.

1) User Irrepressible (UI): We present the definition of UI in the following for completeness.

A sequence has the period equal to $L \in \mathbb{Z}^+$. We denote the residues of the integer modulo L by the set $\mathbb{Z}_L \triangleq \{0, 1, 2, ..., L-1\}$. Given a binary sequence $s(t)$, the *characteristic set* of $s(t)$ is defined as:

$$I_s = \{t \in \mathbb{Z}_L \mid s(t) = 1\}. \tag{1}$$

As an example, consider the following sequence:

$$s = (0, 1, 1, 1, 0, 0, 0, 1),$$

its characteristic set is then given by $I_s = \{1, 2, 3, 7\}$. Each value of I_s represents the indexes of the ones in the sequence. Notice that we start the indexing of the positions of the bits in the sequence from 0. The characteristic set of a sequence can also be seen as an alternative representation of the sequence.

A cyclic shift of sequence $s(t)$ by τ produces a sequence $s^{(\tau)}(t)$ which is a shifted sequence of $s(t)$ by τ bits, where $\tau \in \{0, 1, \ldots, L-1\}$. Note that we consider right cyclic shift. A sequence can also be represented in terms of its characteristic set, $I_{s^{(\tau)}} = \{x \bigoplus \tau \mid x \in I_s\}$, where $\bigoplus$ is the addition modulo L. Due to CCw/oFB with arbitrary time delay offsets, a sequence $s_i(t)$ can be *equivalent* to another sequence $s_j(t)$ if the characteristic set of the sequence $s_i(t)$, I_{s_i}, is also found in the set of all possible shifted sequences of the characteristic set of the sequence $s_j(t)$, I_{s_j}.

Definition 2: A sequence set $\mathcal{S}$ is called *user-irrepressible* (UI) if for all sequences $s_i \in \mathcal{S}$, the individual throughput of each user is strictly positive for all possible time delay offsets, i.e., at least one collision-free transmission is guaranteed for every user in each sequence period.

Note that one application of user-irrepressibility is to offer bounded delay such that each user does not have to wait for longer than the given period L for having a packet to be sent without collision.

2) Minimum Individual Throughput Maximization: We denote the throughput of user i by $T(s_i)$, where $s_i \in \mathcal{S}$. We choose to maximize the minimum individual throughput of users, i.e., $\min_i T(s_i)$, under arbitrary time delay offsets among all users. Without loss of generality, we use τ_i to denote the time delay offset imposed on user i, where $\tau_i \in \{0, 1, \ldots, L-1\}$, relative to a common periodical starting reference time. Therefore, the goal is to:

$$\underset{\mathcal{S}}{\text{maximize}} \quad \left(\min_i T\left(s_i^{(\tau_i)}\right)\right) \tag{2}$$

for arbitrary τ_i, $\forall i$, subject to the selected sequence set $\{s_i\}$ for each user $i \in M$.

IV. Applying DRL to Find TMUI Protocol Sequences

We use the framework of Reinforcement Learning (RL), where one agent interacts with its environment by taking actions. Upon taking an action, the agent is presented with a new state and consequently receives a reward signal as feedback. We use DRL, and specifically, policy-based methods, where the choice of the action, the *policy*, is made by a Deep Neural Network. We treat the sequence discovering process as an episodic bit-flipping game. Therefore, the environment is the sequence set, and the actions consist in flipping bits of the sequences. We made the choice of keeping the duty factor and hence the number of transmissions for each user fixed throughout the episode. For this reason, and in this work, only two bits have to be flipped per step. In the following, we describe the main elements of the DRL approach which aims to discover TMUI protocol sequences under CCw/oFB model for the medium access control problem.

A. The State Representation

The state $s \in S$ in our system is represented by the sequence set and a side indicator, where the side indicator is to specify on which sequence the bits are going to be flipped. The sequences are indexed starting from 0. The side indicator shows the index of the sequence on which the action will be applied.

B. Action

The action space is denoted by $\mathcal{A}$. Each action in the action space is denoted by $a \in \mathcal{A}$. In our work, the action corresponds to the two position(s) of the bit(s) that would be flipped. Fig. 1 illustrates how the action is applied to the state, in which there are two sequences and each has length $L = 6$. Note that the side indicator in the original state (on the left hand side) is 0, which means that the first sequence will be flipped. This state is fed to the neural network and as an example, it selects the action $(1, 4)$. Therefore, the 1st and 4th bits of the first sequence are flipped. The new state (on the right hand side) is outputted: the first sequence has been flipped and the second sequence remains the same, whereas the side indicator is now 1 to express the fact that the second sequence will be flipped during the next step.

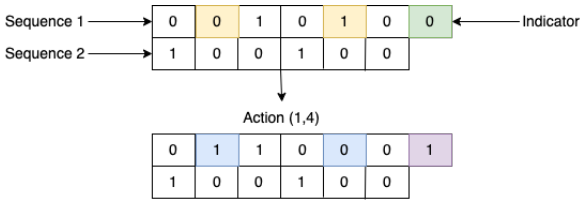

Fig. 1. An example of actions taken on the sequences

C. Reward

The reward is a feedback given in RL about the desirability of the action that it took at each learning step. In our case, it is a measure of how far a sequence set is from the optimal possible set that matches the two required performance metrics of TMUI, i.e., being user-irrepressible and maximizing the minimum individual throughput among the users per sequence period regardless of any starting time offsets. The reward is used by the DRL algorithm PPO, to eventually update the weights of the DNN model. The episode either ends after a predefined number of steps (actions) or when a desired sequence set is found. To avoid the problem of sparse rewards in RL [14], a reward is computed at each step of the episode, instead of computing a reward only at the end of the episode based on the final quality of the sequences. At each step, the returned reward is stored in an episodic buffer. When the episode ends either after a set of sequences has been found with the desired throughput or after a predefined maximum number of steps per episode, the maximum reward in the buffer is noted. The reward buffer is used at the end of the episode, to select the maximum intermediate reward as a final reward, indicative of the best sequence sets that were achieved.

In the following, we define the *desirability* of an obtained sequence set as a numerical indicator that is correlated with better achieving the maximization goal of Eq. (2). Consider a system of M users. The number of shifts each user can have with respect to other users in the channel is equal to L. The desirability of a sequence set is obtained by summing the number of times when the user has met the predefined target throughput, considering all possible shifts. In the ideal case, the desirability of the sequence set, S, should be at its maximum. We denote the maximum desirability by D_{max}. D_{max} occurs when we obtain an individual throughput equal to or greater than a predefined target for all the users and all the shifts.

Since we have M users, hence L^M possible shifts, and since for each set of shifts, at best, all M users meet the throughput target, one has:

$$D_{\max} = ML^M \tag{3}$$

The metric of desirability of a sequence set, $D(S)$, is used to calculate the reward. We assign a reward of $R = 1$ if $D(S) = D_{\max}$. This value is chosen as a normalized maximum value when an optimal solution has been found.Otherwise, the reward is just the achieved performance $D(S)$ normalized by $D_{\max}$; and empirically we found that it was best to add another penalty equal to -1 when the optimum is not reached. Indeed, this reward structure is half way between a binary 0/1 reward that just indicates whether the optimal is reached (which puts a high emphasis on achieving the optimum, not only being close to it), and a smoother reward $\frac{D(S)}{D_{\max}}$ is proportional to how close a candidate solution is to the optimum (which guides better the RL algorithm towards good solutions).

$$R = \begin{cases} 1, & \text{if } D(S) = D_{\max}, \\ \frac{D(S)}{D_{\max}} - 1 & \text{otherwise.} \end{cases} \tag{4}$$

D. Optimization Problem

The aim of deep reinforcement learning is to maximize the total expected reward for a given episode. This is an optimization problem, where the objective function to be maximized is the expected cumulative return accumulating all the rewards obtained during the episode. One family of methods to achieve this are policy gradient methods, based on the policy gradient theorem (such as the simplest one, the REINFORCE algorithm [15]). Other examples can be found in [16] and references therein. In our work, we use Proximal Policy Optimization (PPO) algorithm [17] for TMUI sequences, as it is one of the state-of-the-art algorithms, and has several available implementations.

E. System Parameters

The system parameters that we used in our framework are listed and defined below for completeness.

1) Duty Factor D_f: The duty factor is the fraction of the time that the user is transmitting in a sequence of period L. We chose identical duty factors for all users. We select a duty factor given by

$$D_f = 1/M, \tag{5}$$

where M is the number of users.

2) Number of Transmissions α: The number of transmissions during the period of L time slots is thus equal to

$$\alpha = D_f \times L. \tag{6}$$

As an example, a duty factor D_f of $\frac{1}{2}$ and a sequence length L of 10 would mean that the number of transmissions α is equal to $\frac{1}{2} \times 10 = 5$. By substituting D_f in (5) to (6), the number of transmissions, denoted by α, can also be written as

$$\alpha = L/M. \tag{7}$$

3) Target Individual Throughput T: The target individual throughput is the throughput we would like to achieve for each user when sharing the channel. The number of transmissions, α, is given by (7). Each user will experience collisions, and hence the individual throughput will certainly not be equal to (7). In addition, in our experiments, the number of transmissions α is rounded down to the nearest integer below its current value. Considering symmetric service and user fairness, we divide the number of transmissions, α, for each user by the total number of users, M, sharing the channel. Therefore, the individual throughput is given by

$$T = \alpha/M \tag{8}$$

substituting $\alpha = \frac{L}{M}$ in the above equation, we obtain,

$$T = \lfloor L/M^2 \rfloor. \tag{9}$$

The floor of the target throughput is taken since it needs to be an integer.

V. Numerical Results

In this section, we present some numerical results to illustrate the convergence of proposed sequence-finding DRL scheme and the resulting user throughput performance when using the obtained TMUI protocol sequences, in comparison to that obtained by slotted ALOHA.

A. Experimental Setting

We used the implementation of PPO from stable-baselines3 [18] (derived from OpenAI baselines) for the DRL algorithm, and implemented an environment corresponding to the description in Section IV. As indicated previously, the sequences are constructed by flipping some bits at each step of the episode, starting from set of randomly initialized binary sequences that still match the duty-factor constraints. The main system parameters are shown in Table I. In our experiment, as a proof of concept, we focus on the construction of TMUI sequences for two users. The duty factor of each user is set to $\frac{1}{2}$ by symmetry.

TABLE I
Main System Parameters in Our DRL Framework

Learning rate	$1.5 \times e^{-4}$
Policy network	(64, ReLu, 64, ReLu)
Value network	(64, ReLu, 64, ReLu)
Batch size	4096
Number of epochs	20
Clip range	0.1
Discount factor γ	1

We experiment with various cases of sequence length L. The number of transmissions per user and the target individual throughput are set according to (6) and (9), respectively. Table II shows their experimental settings.

B. Initial Conditions

The experiments were launched for each L separately. For each L, we run 100 experiments with different random seeds. This means that there were 100 experiments and each with different random sequences used as the initial sequences as the input to the proposed DRL scheme.

TABLE II
Numerical Values with Rounding Off for Each Experiment

Length L	Number of transmissions α	Target individual throughput T
11	5	2
15	7	3
19	9	4
23	11	5

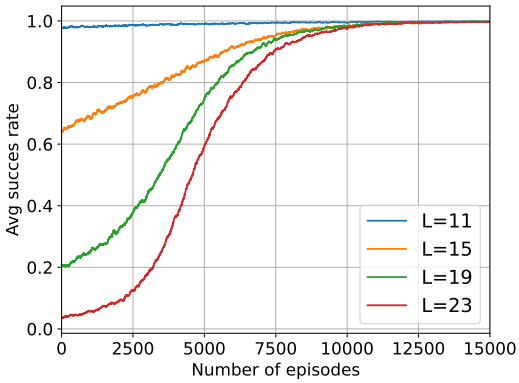

Fig. 2. Average success rate vs number of episodes

C. Convergence

We evaluate the success probability and time taken for each experiment for each user to reach the intended target throughput T. Under our method, we assess user throughput over a sliding window equal to 100 episodes. Table III shows the convergence time corresponding to a given success probability that in how many experiments out of the total number of experiments, the users have already obtained their target throughput.

TABLE III
Convergence Time at Success Probability = 0.99

Length L	Convergence time in steps
11	3468-3568
15	10864-10964
19	10709-10809
23	11012-111112

Fig. 2 shows the average success rate as a function of the number of episodes. Results show that for the various L, our framework performs very well by reaching a success probability of very close to 100%. We can also see that the larger the L value, the larger the number of episodes (time) needed for having the same success rate.

D. Individual Throughput Comparison

We compare the user throughput performance of the obtained TMUI sequences against slotted ALOHA. We consider the sequence lengths, L, from $L = 10$ to $L = 23$. As an illustration we show only $L = 11, 15, 19, 23$ in Table IV.

1) Slotted ALOHA: The probability of transmission on a particular time slot by a user is denoted by p. We set p equal to the actual duty factor of TMUI scheme such that they have the same transmission rate. It is worth noting that we have not compared with other UI sequences in the literature since

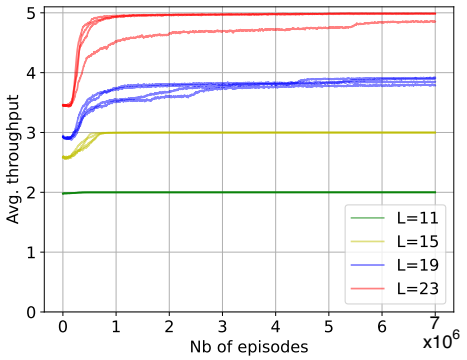

Fig. 3. Throughput convergence curves with the number of episodes

they are not designed to maximize the minimum individual throughput among the users.

2) Comparison between Slotted ALOHA and TMUI: Table IV compares the individual throughput of users under slotted ALOHA and TMUI scheme. We can see that TMUI always has non-zero minimum individual throughput since it is user-irrepressible (UI) while users under slotted ALOHA can have zero throughput. Besides, the minimum individual throughput does meet our predefined target T, given by Table II. We also observe that the variance of the individual throughput of TMUI is zero since we maximize the minimum individual throughput.

TABLE IV
COMPARISON OF INDIVIDUAL THROUGHPUT BETWEEN TMUI AND SLOTTED ALOHA

	Sequence length L	11	15	19	23
Slotted ALOHA	Min throughput	0	0	0	0
	Max throughput	8	9	11	12
	Average throughput	2.74	3.61	4.78	5.76
	Variance	2.17	2.74	3.71	4.44
TMUI	Min throughput	2	3	4	5
	Max throughput	2	3	4	5
	Average throughput	2	3	4	5
	Variance	0	0	0	0

E. Initialization and Random Seeds

In this subsection, we describe another aspect of our experimental study. In this setup, the initial set of bits was changed in the beginning of the episodes. Fig. 3 shows the average throughput of the users during each experiment with various random seeds. We can see that for small L such as 11 and 15, almost all the experiments have the same evolution and towards their respective target throughput. For $L = 19$ and $L = 23$, most seeds have similar convergence and nevertheless all of them are heading towards the target throughput. This demonstrates that the DRL framework is capable of taking any randomly initialized sequences and turning them to TMUI sequences with a reasonable number of bit flips.

VI. CONCLUSION

In this work, we develop a new DRL based protocol sequence construction framework for medium access control design. We consider a system that requires low latency communication such as URLLC, with collision channel without feedback. The obtained TMUI is outputted by our DRL scheme under the two specific properties of user-irrepressibility (UI) and minimum individual throughput maximization. We present the proposed method and its implementation. Numerical results show the robust performance of TMUI sequences compared with conventional slotted ALOHA for simple random access channel. A future work is to generalize the above scheme for any number of users with arbitrary duty factors and for any suitable user service requirement. It is also important to improve the efficiency of the learning process through new algorithmic development and advancement.

ACKNOWLEDGMENT

We would like to thank Dr. Y. Zhang and L. Salaun for some helpful discussions and comments during this work. Besides, a part of the work was carried out at LINCS (www.lincs.fr), and a part was also in the context of the 5G-mMTC project.

REFERENCES

[1] J. Massey and P. Mathys, "The collision channel without feedback," *IEEE Transactions on Information Theory*, vol. 31, no. 2, pp. 192–204, 1985.
[2] C. S. Chen, W. S. Wong, and Y.-Q. Song, "Constructions of robust protocol sequences for wireless sensor and ad hoc networks," *IEEE Trans. on Vehicular Technology*, vol. 57, no. 5, pp. 3053–3063, 2008.
[3] W. S. Wong, "Transmission sequence design and allocation for wide-area ad hoc networks," *IEEE Trans. on Veh. Technol.*, vol. 63, no. 2, 2014.
[4] L. Salaun, C. S. Chen, Y. Chen, and W. S. Wong, "Constant delivery delay protocol sequences for the collision channel without feedback," in *19th Intl. Symposium on Wireless Personal Multimedia Communications (WPMC)*, 2016, pp. 429–434.
[5] I. Hmedoush, "Connectionless transmission in wireless networks (IoT)," Ph.D. dissertation, INRIA Paris-Saclay, 2022.
[6] K. W. Shum, C. S. Chen, C. W. Sung, and W. S. Wong, "Shift-invariant protocol sequences for the collision channel without feedback," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3312–3322, 2009.
[7] Y. Zhang, Y.-H. Lo, K. W. Shum, and W. S. Wong, "New CRT sequence sets for a collision channel without feedback," *Wireless Networks*, 2019.
[8] Y. Shao, S. C. Liew, and T. Wang, "AlphaSeq: Sequence discovery with deep reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3319–3333, 2020.
[9] H. Yu, Y. Kang, Z. Shi, Y. Shao, Y. Lin, and Y. Zhang, "Design of deterministic grant-free access with deep reinforcement learning," in *IEEE 20th Intl. Conf. on Commun. Technology*, 2020, pp. 944–948.
[10] L. Huang, H. Zhang, R. Li, Y. Ge, and J. Wang, "AI coding: Learning to construct error correction codes," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 26–39, 2020.
[11] Y. Liao, S. A. Hashemi, J. M. Cioffi, and A. Goldsmith, "Construction of polar codes with reinforcement learning," *IEEE Transactions on Communications*, vol. 70, no. 1, pp. 185–198, 2022.
[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press Cambridge, 1998.
[13] K. W. Shum, W. S. Wong, C. W. Sung, and C. S. Chen, "Design and construction of protocol sequences: Shift invariance and user irrepressibility," in *IEEE Intl. Symposium on Infor. Theory*, 2009, pp. 1368–1372.
[14] J. Hare, "Dealing with sparse rewards in reinforcement learning," *arXiv preprint arXiv:1910.09281*, 2019.
[15] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, vol. 12, 1999.
[16] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347
[18] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable Baselines3," 2019.