Deep-IRSA: A Deep Reinforcement Learning Approach to Irregular Repetition Slotted ALOHA

Ibrahim Ayoub, Iman Hmedoush, Cedric Adjih, Kinda Khawam, Samer Lahoud

Deep-IRSA: A Deep Reinforcement Learning Approach to Irregular Repetition Slotted ALOHA

Ibrahim Ayoub
Inria, Saclay Center, Palaiseau, France
Université Saint-Joseph de Beyrouth, Lebanon
ibrahim.ayoub@inria.fr

Iman Hmedoush
Inria
Saclay Center, Palaiseau, France
iman.hmedoush@inria.fr

Cédric Adjih
Inria
Saclay Center, Palaiseau, France
cedric.adjih@inria.fr

Kinda Khawam
DAVID laboratory
Université de Versailles, UVSQ
Versailles, France
kinda.khawam@uvsq.fr

Samer Lahoud
Faculty of Engineering ESIB
Université Saint-Joseph de Beyrouth, ESIB, CIMTI
Beirut, Lebanon
samer.lahoud@usj.edu.lb

***Abstract*—The Internet of Things (IoT) aims to connect billions of devices, most of which are power and memory-constrained. Such constraints require efficient network access. "Irregular Repetition Slotted Aloha" (IRSA) meets such requirements. In this paper, we optimize IRSA using Deep Reinforcement Learning to obtain Deep-IRSA, and introduce variants that allow re-transmission and user priority classes. We observe the learned degree distribution and throughput, showing that Deep-IRSA performs excellently, is generic, and could well replace known approaches for smaller frame sizes and IRSA variants.**

I. Introduction

A. The Internet of Things (IoT) and Communications

Simply put, enabling IoT means enabling a wide range of devices including, but not limited to, sensors, mobiles, and cars. Connecting these devices to the Internet, and, consequently to each other facilitates data exchange and overall management. The growing number of IoT-enabled devices in addition to their constrained nature, especially in terms of power consumption and memory, gave rise to new research challenges. These challenges should be faced while conserving the reliability of IoT devices and keeping in mind their simple nature.

Most recent wireless standards have to support scenarios with a huge number of connected users and with sporadic transmissions. Therefore, there is a need for prior scheduling before devices are granted access. Cellular networks are a great example of such wireless networks. In cellular networks, the users reserve certain network resources that are exclusively granted to them. Allocating resources allows reliable communications, especially in loaded networks. Cellular network access protocols are embedded with techniques to avoid collisions between different users. "Code Division Multiple Access" (CDMA) and "Orthogonal Frequency-Division Multiple Access" (OFDMA) for example, try to avoid collisions by using orthogonal codes or orthogonal resource blocks for each user, respectively. However, mobile phones and base stations in cellular networks are more powerful than IoT devices and their communication patterns are different. IoT devices become active and transmit for very short duration (maybe only one packet) compared to the total non-activity duration. IoT packets are usually small, so that any scheduling process or collision avoidance technique could consume more resources than the message itself. Therefore, unscheduled random access became a necessity.

B. Irregular Repetition Slotted Aloha

The need for simple, yet effective random access techniques demands looking into traditional random access techniques, mainly ALOHA-like protocols [1]. One of the most prominent protocols of this family is Irregular Repetition Slotted ALOHA (IRSA) [2]. The model is as follows: A large number of wireless nodes is covered by one "Base Station" (BS). Time is divided into frames which are in turn divided into slots. At any given moment, a node is active with a certain activation probability.

At the beginning of a frame, an active node sends several copies of its packet on separate slots in that frame. The number of copies that a node sends is governed by the node repetition degree d. For example, d = 3 means that the node sends three copies of its packet on three slots in the frame. The choice of the degree is according to a degree distribution Λ [2]. $\Lambda(x) = 0.5x^3 + 0.4x + 0.1$, for example, tells a node to use $d = 3$ for 50% of the time, $d = 1$ for 40% of the time, and $d = 0$ for 10% of the time. $d = 0$ means the node does not transmit any packet. The packets sent by one node contain pointers in their headers to locate the rest of the copies of any packet.

At the BS, decoding begins after the whole frame has been received. Collisions occur when more than one user transmits on the same slot. We suppose that the slots which contain collisions cannot be decoded. Hence, decoding starts by looking for singletons which are slots containing only one packet from one user. When singletons are decoded successfully, their pointers are used to locate and remove their copies in the frame [2]. The main idea is to use "Successive Interference Cancellation"

(SIC): after decoding and reconstructing the physical signal of the packets recovered from singleton slots, IRSA looks for their copies in the frame and physically removes them. This is demonstrated in Figure 1.

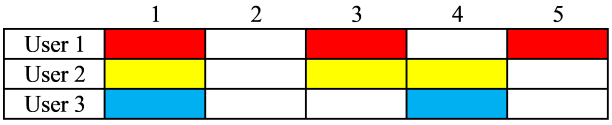


Figure 1: IRSA Frame

In Figure 1, the BS receives the frame and starts the decoding process. User 1 has a singleton on slot 5, so it is decoded and its other copies on slots 1 and 3 are removed. This creates a singleton for user 2 on slot 3. This singleton is decoded and its copies on slots 1 and 4 are removed. Finally, packets of user 3 are decoded as well.

Locating singletons is not always possible. For instance, if user 2 did not send a copy on slot 3, one may end up in a situation as in Figure 2, where no more singletons are found. This is called a stopping set. The packets of users in stopping sets cannot be decoded.

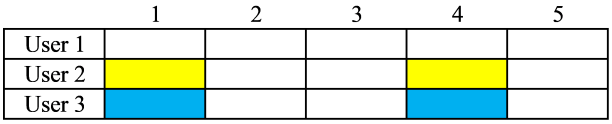


Figure 2: IRSA Stopping Set

In this paper, we use "Reinforcement Learning" (RL), specifically "Deep Reinforcement Learning" (DRL) to identify the best degree distribution such that we maximize the system throughput. In the following, we present a brief overview of RL and DRL. A general reference for RL can be found in [3]. RL is a branch of "Machine Learning" (ML) that focuses on learning to make efficient decisions (actions). Unlike other branches of ML, there is no supervision in RL, but only a *reward* that is used to train the agent to select *actions* in a way that attempts to maximize cumulative future rewards. An agent finds itself in a certain *state* of the environment, and this state is the information used to decide what action to take next.

Formally, the state of the agent has the Markov property: any future state only depends on the current state and not the history. The RL environment can be described as a "Markov Decision Process" (MDP) defined by the tuple $< S, A, P, R, \gamma >$. S is the finite set of possible states, A is the finite set of actions an agent could take, P is the transition matrix from one state to another given a certain action, R is the reward obtained after taking a certain action given a certain state, and finally, $\gamma \in [0, 1]$ is a discount factor which serves to value immediate rewards over delayed ones. If the agent does not have a complete view of the state, then the MDP is said to be a "Partially Observable Markov Decision Process" (POMDP) which is our case as detailed later.

The agent interacts with the environment by taking an action based on its state. After the action is taken, a reward is calculated, which tells the agent how right or wrong the last action was. The goal of the agent is to maximize its reward.

RL's objective is to find an efficient policy. A policy decides the actions an agent selects based on its state and it could be either deterministic or stochastic. Value methods are often associated with deterministic policies: the agent updates a Value Function or a Q-function that estimates how beneficial it is to take an action given a certain state or how good is a certain state. In classical RL, a lookup table can be constructed and the action with the highest Q value is chosen. This is known as Q-Learning and has variants such as SARSA [3]. Stochastic policies are often associated with Policy Gradient Methods. A stochastic policy $\pi(a, s)$ gives the probability of taking a certain action given a certain state [3].

There are problems with large state/action spaces where, for example, a large memory is needed to store the value of each action if a tabular method is used. In such cases, value function approximation is used. Using a "Neural Network" (NN) as a function approximator gave rise to DRL where the reward received is used to update the weights of the NN. In DRL, the state is fed to a NN which can output either Q values or policies (probabilities of taking actions). The agent then either takes the action with the highest Q value or selects the action according to the output policy. In this paper, we use the DRL algorithm "Proximal Policy Optimization" (PPO) [4] and its implementation by OpenAI and others (stable baselines). PPO is a policy gradient method.

II. Related Work

The literature contains a considerable number of articles on Random Access. We focus on Modern Random Access and specifically on IRSA and its variants. The work in [1] is a recent general reference that introduces the need for Modern Random Access in IoT networks and shows how IRSA family protocols are relevant for 5G and 6G. The pioneering work in [5] presents a predecessor of IRSA called "Contention Resolution Diversity Slotted Aloha" (CRDSA) which operates by having each agent sends two copies of its packet on randomly selected slots and applying SIC at the receiver. IRSA itself was presented in [2] and was shown to reach the throughput of 0.97 packets per slot for large frames: instead of two copies, a random number of copies is selected from a degree distribution. The work in [2] studies and optimizes the degree distribution through coding theory tools (density evolution was initially designed for low density parity check (LDPC) codes). Density evolution provides a tool to evaluate the expected performance (throughput or loss) of IRSA given the degree distribution and the load (as average number of users per slot). Density evolution results are valid when frame size grows infinitely. Optimizing IRSA performance given by density evolution is not straightforward because density evolution itself consists of applying iteratively one function. Hence [2] uses a meta-heuristic method, differential evolution, to (attempt to) find the optimal degree distributions.

Several variants of IRSA exist, for instance, the ones introduced by [6] and [7]. "Priority IRSA" (P-IRSA) in [6]

associates priorities to nodes in the network. The idea is that high priority classes should have a higher packet delivery ratio and throughput than lower priority classes. This is achieved by introducing an access control mechanism that allows or denies the access to the network based on users' priorities. Accordingly, P-IRSA assigns probabilities of network access based on the load conditions. The work in [7] addressed energy consumption in IRSA by decreasing the number of sent replicas if the load conditions allow it, (for instance at low load) while maintaining a similar throughput. The work is based on a tracking degree distribution control (TDDC) algorithm with an adaptive degree distribution scheme.

Because the optimization of IRSA variants can be difficult, there has been a trend to use RL methods to optimize the performance of IRSA and its variants. The work in [8] formulates the problem as a "Multi-Armed Bandit" (MAB) at the BS. The BS learns online an optimal transmission strategy defined by the number of source packets that has been sent and the repetition of each packet by trial and error. This has been done by considering each possible transmission as an action (an arm). An internal state represented by the buffer of each agent was introduced in [9] where Decentralized Reinforcement Learning was used to let nodes decide the number of replicas to be sent based on their own buffer state. In [10], the authors use a variant of Reinforcement Learning called Regret Minimization (RM) to optimize the degree of frameless IRSA.

In the literature, there have not been a generic method to perform optimization of all the different variants of IRSA. Density evolution equations can sometimes be adapted for some simpler variants, but then differential evolution often has to be used with it; it does not necessarily converge well for large solution spaces. In addition, density evolution is valid only for asymptotically infinite frame size, and even if some approximations for finite length frame size exist, they are not necessarily accurate for all loads. When the state of each frame is not independent (e.g. repetitions, frameless, etc.), optimization is even more complex, and this is where reinforcement learning has been proposed: nevertheless, DRL has not been used for finding good IRSA policies, which is problematic for large state spaces and stochastic policies.

The current work addresses these issues; we show how to apply DRL to optimize different variants of IRSA. Our results illustrate that DRL performs well, and provides excellent degree distributions. Thus, we provide a generic method for optimizing IRSA, where it is easy to introduce modifications and variants of IRSA. We also reveal this by studying and combining two variants of IRSA: re-transmissions, and priority classes. To our knowledge, this work is the first to use DRL for IRSA action selection.

III. System Model

We consider a group of nodes communicating with a single BS. Time is divided into frames and each frame is further divided into M slots. At the beginning of each frame, N new users enter the network. Each of these users has a packet to send in the frame. The network load $G = \frac{N}{M}$ represents the average number of unique packets (users) transmitted per slot. The throughput is the average number of correctly decoded users per frame and the normalized throughput T represents the average number of correctly decoded users per slot. In some other scenarios, users may be divided into classes of service based on the returned reward after decoding. The network includes newcomers whose arrival rate is fixed at N new users per frame, but may, in some variants, also includes re-transmitters. Re-transmitters are users that failed to be decoded in previous frames and will re-transmit for a maximum number of times before giving up. The definition of G does not change. Our proposed system model considers two types of users: (a). The newcomers who are the users connected to the network for the first time. The arrival rate of the newcomers is fixed at N new users per frame. (b). In some other scenarios, we consider the re-transmitters. Re-transmitters are users that failed to be decoded in previous frames and will re-transmit for a maximum number of times before giving up. The definition of G does not change in both cases. The application of DRL is done according to the following principles:

- IRSA is recast in the DRL framework, it becomes a multi-agent system, where each node is an agent.
- The actions of the nodes are essentially the degree selections.
- The state is a feedback about the previous frame containing the number of undecoded users combined with an internal state from each user containing its class and whether it is a newcomer to the network or a re-transmitter.

Due to the lack of communication between users in the network, each user only has a partial view of the state. This transforms the problem into a POMDP [3, Sect. 17.3]. Formally, the POMDP elements are: (a). the state S: a feedback from the BS and the internal user state, (b). the action A: a set of actions where each action consists of choosing a degree, (c). the conditional transition probabilities P: is not explicitly computed, (d). the reward R: is the average probability of users being correctly decoded and (e). the discount factor $\gamma = 1$ since the reward is calculated at the end of each frame.

The training is done in a centralized offline manner: an optimized NN model is obtained through training by simulations. The intent is that in a real application, this NN model will be distributed to the devices beforehand, and would be used as is by the nodes to perform IRSA random access in a real network. In this work, the NN was trained for a single load, mainly $G = 1$.

The learning process is done as follows:

- Users receive feedback from the BS to inform them about the number of collisions in the past frame.
- The user adds an internal state to the received feedback: its class of service and whether or not it is a re-transmitter.
- The feedback, together with the internal state is fed to the neural network model as the state (as input).

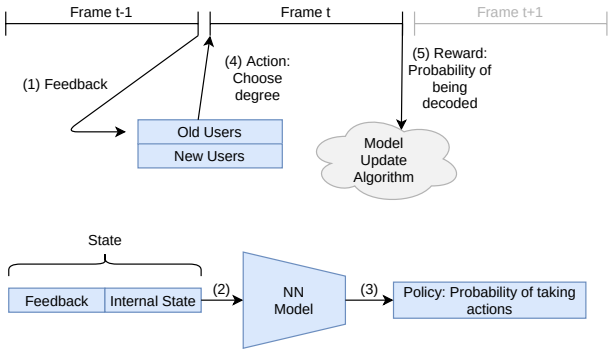


Figure 3: The Learning Process

- The neural network model outputs the probabilities of taking each possible action.
- The actions are the repetition degrees of each user. Each degree is between 0 and a maximum degree d_{max}, which we set at the beginning of the learning process.
- After all the active users select and perform their actions, the frame is decoded at the BS. The reward is computed as the average probability of correctly decoding each user at the end of the frame which is given by the ratio reward $= \frac{\text{number of decoded users}}{\text{total number of users}}$.

The learning process is illustrated in Figure 3.

A. Problem formulation

In general, the main objective of IRSA optimization is to vary the protocol parameters for optimizing a certain performance metric. In most of IRSA optimization literature, the system metric to be optimized is the throughput as in [8], [9]. Other IRSA variants [10], [6], [7] have explored other metrics as the weighted throughput, the delay, or the power consumption. In this work, our objective is to optimize the throughput of the network. The weighted throughput is also optimized when users are divided into classes. The optimization problem is formulated as follows:

$$\begin{aligned} &\underset{(\Lambda_0,\ldots,\Lambda_{d_{max}})}{\text{maximize}} \quad T \\ &\text{s.t. } 0 \leq \Lambda_0, \ldots, \Lambda_{d_{max}} \leq 1 \\ &\sum_{i=0}^{d_{max}} \Lambda_i = 1 \end{aligned} \tag{1}$$

For weighted throughput and considering we have class 1 and class 2, with throughputs: $T_1(\Lambda_0^{(1)}, \ldots, \Lambda_{d_{max}}^{(1)})$ and $T_2(\Lambda_0^{(2)}, \ldots, \Lambda_{d_{max}}^{(2)})$ (resp.):

$$\begin{aligned} &\underset{(\Lambda_0^{(1)},\ldots,\Lambda_{d_{max}}^{(1)})\ (\Lambda_0^{(2)},\ldots,\Lambda_{d_{max}}^{(2)})}{\text{maximize}} \quad \alpha T_1 + \beta T_2 \\ &\text{s.t. } 0 \leq \Lambda_0^{(1)}, \ldots, \Lambda_{d_{max}}^{(1)} \leq 1, \\ &\quad 0 \leq \Lambda_0^{(2)}, \ldots, \Lambda_{d_{max}}^{(2)} \leq 1, \\ &\sum_{i=0}^{d_{max}} \Lambda_i^{(1)} = 1 \text{ and } \sum_{i=0}^{d_{max}} \Lambda_i^{(2)} = 1 \end{aligned} \tag{2}$$

where: $(\Lambda_i)_{0 \leq i \leq d_{max}}$ are the coefficients of the degree distribution. Λ_i is the probability of using degree i. α and β are the weights that express how important is the throughput of class 1 and class 2 respectively.

B. Implementation Context

We developed our own IRSA simulator written in Python. The simulations allow to construct frames and generate user transmissions. IRSA iterative decoding was also performed using a collision model after the decoding information is obtained (decoded users, throughput, etc.). For DRL, we used OpenAI stable-baselines. The reward is the average probability of users being correctly decoded or the weighted average probability when classes are implemented. The proposed model contains $2(64 + 64)$ layers of neurons amounting to approximately 4000 weights. We used PPO which is a policy gradient method. The implementation of PPO from stable-baselines uses both a policy function and value function with shared layers. The value function is introduced in PPO to improve convergence [3].

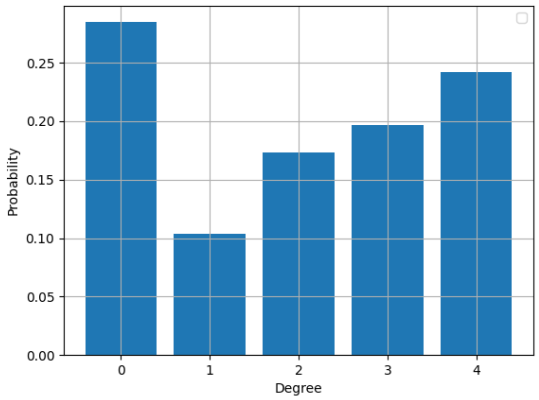


Figure 4: Learned Degree Distribution

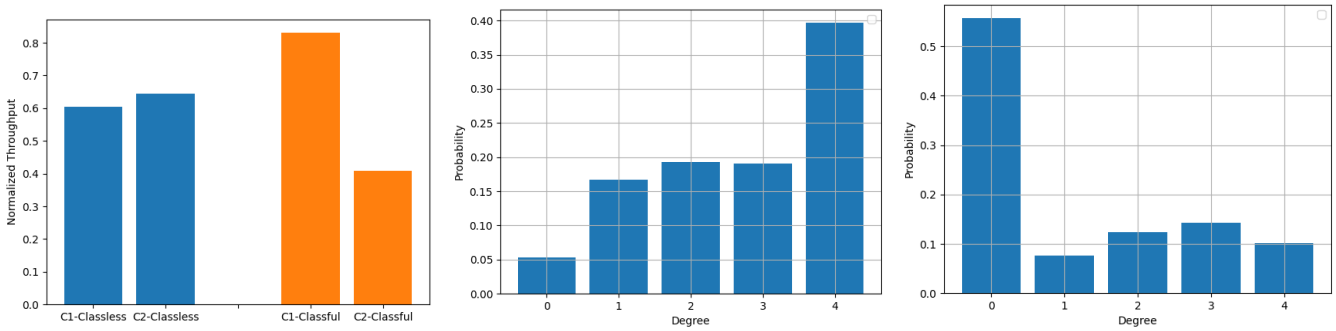
IV. Results and Discussion

In this section, we present the obtained numerical results using our simulator. These results could be recalculated for different network parameters. The parameters we investigate in this work are: frame size, load, classes of users and retransmissions.

The arrival of users in the system is fixed at N users per frame. Rewards are calculated once the decoding of the frame is done (the reward is zero at initialization). The normalized throughput is an additional metric used for studying the performance. Training is done in steps. Each step accounts for an action taken by one agent (degree selection). A full episode is completed when all the active agents take their actions after which the reward is calculated. The obtained graphs are smoothed using a Savitzky-Golay [11] filter of order 3.

A. Learning the Degree

In the following, we aim to plot the learned degree distribution $\Lambda(x)$. We consider $N = 50, M = 50$. All the users belong to the same class hence, they learn the same degree



(a) Throughput with and without Classes (b) Learned Degree Distribution for Class 1 (c) Learned Degree Distribution for Class 2

Figure 5: Impact of Classes and Weights

distribution. Re-transmission is not allowed. The maximum degree $d_{max} = 4$. The learned degree distribution is illustrated in Figure 4. The y-axis represents the probability of using different degrees while the x-axis represents the possible degrees between 0 and d_{max}. We notice that the degree 0 (the agent does not transmit) is the most used in this case since we are dealing with the extreme case where the load is high ($G = 1$.) Any node in the network will use degree 0 with a probability ≈ 0.28, degree 1 with a probability ≈ 0.11 and so on. This will show a benefit when dividing users into classes.

B. Impact of Using Classes

In this section, we show the impact of dividing users into different classes of service. For simplicity, we will use two classes. Class 1 is prioritized over class 2 by assigning a larger reward value when one of its users is decoded (1 for class 1 vs 0.5 for class 2). We consider $N = 50,\ M = 50$. Users are evenly distributed between 2 classes: class 1 and class 2. Re-transmission is not allowed and the maximum degree $d_{max} = 4$. Figure 5a shows the difference in the behavior of the model when dividing users into classes. In the left part of the chart, we find that the throughput is almost identical for both classes when both return the same reward (i.e. classes not implemented). The right part of the chart, however, shows clearly that the model prefers class 1 users over class 2 users, when classes are implemented.

Figure 5b and Figure 5c present the learned degree distributions for class 1 and class 2, respectively, when users are divided into two classes. It is noticed that class 1 users are preferred by the model since they use degree 0 much less than class 2 users. Class 1 users send with degree 4 most of the time, unlike class 2 users which use degree 0 (no transmission) most of the time.

C. Effect of Allowing Re-transmissions

In this section, we show the effect of allowing a maximum number of transmissions. For simplicity, we will use two transmissions. Therefore, the user that was stuck in a stopping set or chose degree 0 in the previous frame, is allowed to re-transmit once in the following frame. We consider $N = 30,\ M = 50$. All the users belong to the same class (one class scenario) and the maximum degree $d_{max} = 4$. The number of users is limited to $N = 30$ since using re-transmissions increases the total number of users in the network. In this case, at the beginning of every frame, we will have a total of N new users and some number of re-transmitting users. Figure 6a shows the difference in terms of system throughput in case of re-transmission and without re-transmission. It is evident that when the load conditions in the network allow it, re-transmission may increase the throughput by allowing more users to be decoded.

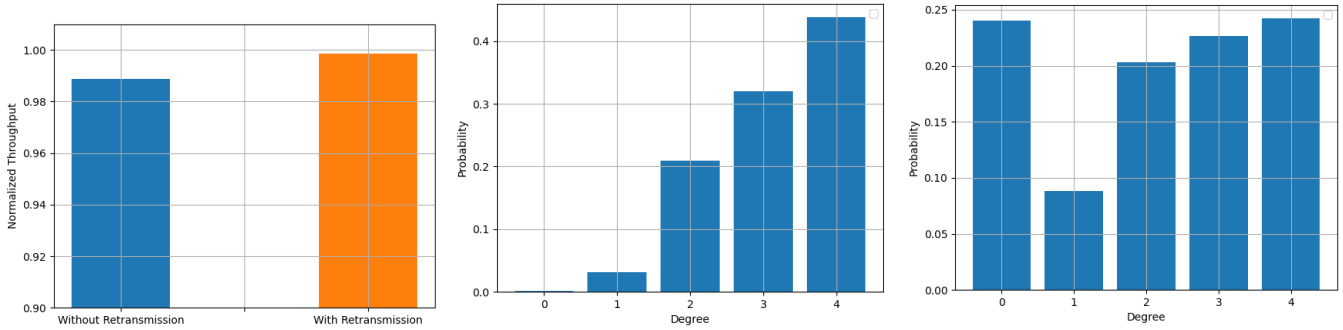
Figure 6b and 6c show the degree distributions learned by new vs re-transmitting users, respectively. The NN model is able to differentiate between new and old users based on the internal state which informs the model if the user in question is a newcomer to the network or a re-transmitter. The new users are prioritized by the network and therefore send with degree 4 mostly as shown in Figure 6b. Users that re-transmit are admitted when load conditions allow it. Thus, they use degree 0 for almost 25% of the time as shown in Figure 6c.

D. Comparison to Near-Optimal Results

In this section, we present a comparison of the throughput we obtained with a highly efficient distribution optimized by differential evolution through simulations. Notice that in the literature, there is a lack of results for optimizing IRSA for smaller frame sizes. In this section, we compute a different distribution for each load in case of small frame size. We consider $N = 50,\ M = 50$. All the users belong to the same class and re-transmission is not allowed. The maximum degree $d_{max} = 4$. We computed optimized distributions through DRL. We also searched for solutions of the described problem in (1) by heuristic optimization, e.g. using differential evolution for an objective function computed by simulations. This approach requires that the state is constant (along with all parameters), hence the optimization is re-run for each different load, and this is less general than DRL.

In Figure 7, we illustrate the results of the obtained throughput when varying the load. There, the throughput we obtained with DRL is compared with the throughput obtained through differential evolution. The plot for DRL is obtained by averaging over the last 25% episodes of the training.

The throughput we obtained for DRL shows excellent performance against the differential evolution results, especially



(a) Throughput with vs without Re-transmission (b) Learned Degree Distribution for New Users (c) Learned Degree Distribution for Old Users

Figure 6: Impact of Re-transmissions

in critical high-load zones between 0.7 and 1.

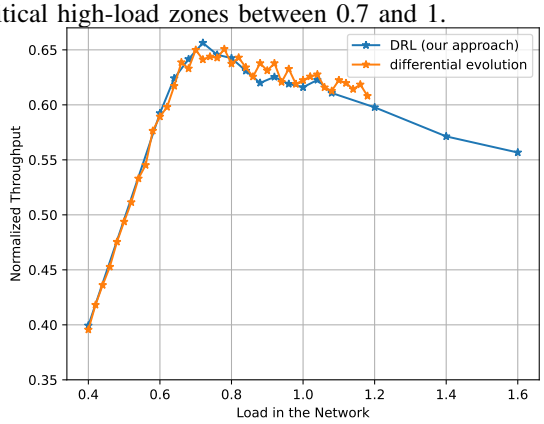


Figure 7: Comparison with Best Known Results obtained by Differential Evolution

E. Performance with Different Frame Sizes

We show in this section the effect on the performance when using different frame sizes and keeping the load equal to one user per slot.

We consider ($N = 10,\ M = 10$), ($N = 25,\ M = 25$) and ($N = 100,\ M = 100$). All the users belong to the same class and re-transmission is not allowed. The maximum degree $d_{max} = 4$. Figure 8 shows that the obtained normalized throughput in the network is not affected by the frame size, considering that we have the same load conditions.

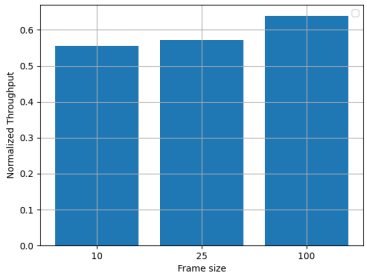


Figure 8: Throughput for Different Frame Sizes

V. Conclusion

In this work, we aim to optimize Modern Random Access for IoT wireless networks. Our work studies IRSA protocol and its variants. We focus on the small frame size scenarios for different network parameters, specifically for two variants: one which allows re-transmissions and another which has user priority classes. We apply Deep Reinforcement Learning techniques to solve this problem. The results show an excellent performance (especially when learning optimal degree distributions). More generally, our work provides a generic method to optimize IRSA and its different variants. Our proposed method proves that it is a promising alternative to known methods in the literature (differential evolution, density evolution, etc.). Future work includes extending the current work with a new variant that incorporates slot (or groups of slots) selection, which enables the agents to intelligently select slots instead of using random selection.

References

[1] F. Clazzer, A. Munari, G. Liva, F. Lazaro, C. Stefanovic, and P. Popovski, "From 5g to 6g: Has the time for modern random access come?" *arXiv preprint arXiv:1903.03063*, 2019.

[2] G. Liva, "Graph-Based Analysis and Optimization of Contention Resolution Diversity Slotted ALOHA," *IEEE Transactions on Communications*, vol. 59, no. 2, pp. 477–487, February 2011.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*, 2nd ed., ser. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.

[5] E. Casini, R. De Gaudenzi, and O. Del Rio Herrero, "Contention resolution diversity slotted aloha (crdsa): An enhanced random access scheme for satellite access packet networks," *IEEE Transactions on Wireless Communications*, vol. 6, no. 4, pp. 1408–1419, 2007.

[6] J. Sun, R. Liu, Y. Wang, and C. W. Chen, "Irregular Repetition Slotted ALOHA with Priority (P-IRSA)," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*. Nanjing, China: IEEE, May 2016, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/document/7504302/

[7] H. Jia, Z. Ni, C. Jiang, L. Kuang, S. Guo, and J. Lu, "Enhanced irregular repetition slotted aloha with degree distribution adjustment in satellite network," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[8] L. Toni and P. Frossard, "Irsa transmission optimization via online learning," *arXiv preprint arXiv:1801.09060*, 2018.

[9] E. Nisioti and N. Thomos, "Decentralized reinforcement learning based mac optimization," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2018, pp. 1–5.

[10] I. Hmedoush, C. Adjih, and P. Mühlethaler, "A regret minimization approach to frameless irregular repetition slotted aloha: Irsa-rm," in *MLN - International Conference on Machine Learning for Networking*, 2020.

[11] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.