DS-IRSA: A Deep Reinforcement Learning and Sensing Based IRSA

Iman Hmedoush, Pengwenlong Gu, Cédric Adjih, Paul Mühlethaler, Ahmed Serhrouchni

DS-IRSA: A Deep Reinforcement Learning and Sensing Based IRSA

Iman Hmedoush*, Pengwenlong Gu*, Cédric Adjih*, Paul Mühlethaler*, and Ahmed Serhrouchni†
* Inria, France
† LTCI, Telecom Paris, Institut Polytechnique de Paris, France

***Abstract*—One of the main difficulties to enable the future scaling of IoT networks is the issue of massive connectivity. Recently, Modern Random Access protocols have emerged as a promising solution to provide massive connections for IoT. One main protocol of this family is Irregular Repetition Slotted Aloha (IRSA), which can asymptotically reach the optimal throughput of 1 packet/slot. Despite this, the problem is not yet solved due to lower throughput in non-asymptotic cases with smaller frame sizes. In this paper, we propose a new variant of IRSA protocol named Deep-Learning and Sensing-based IRSA (DS-IRSA) to optimise the performance of IRSA in short frame IoTs, where a sensing phase is added before the transmission phase and users' actions in both phases are managed by a deep reinforcement learning (DRL) method. Our goal is to learn to interact and ultimately to learn a sensing protocol entirely through Deep Learning. In this way, active users can coordinate well with each other and the throughput of the whole system can be well improved. Simulation results show that our proposed scheme convergence quickly towards the optimal performance of almost 1 packet/slot for small frame sizes and with enough minislots and can achieve higher throughput in almost all cases.**

I. Introduction

The fast development of the Internet of Things (IoTs) in recent years led to the development of MAC layer protocols that seek to support the requirements like high flexibility, low transmission delay and low transmit power. The Irregular Repetition Slotted ALOHA (IRSA) protocol [1] is a promising one since it enhances the contention resolution diversity slotted ALOHA (CRDSA) protocol by allowing each user to transmit more than two replicas in each frame with a certain degree distribution. Then, by exploiting a successive interference cancellation (SIC) algorithm at the receiver side, it can achieve higher throughput compared to other ALOHA protocols.

One of the main roles of MAC protocols is to decide the transmission strategy of the connected nodes. It has been proved that with IRSA, 97% packets can be decoded [2] in cases with a very large frame length. However, in practical IoT scenarios, since both frame length and transmit energy are limited, collisions may still cause a large number of data packets to fail to be decoded, and the resulting retransmission will cause a noticeable peformance decrease. Therefore, how to realize the full potential of the IRSA protocol in IoT networks remains a challenging subject.

In the past few years, many research efforts have been made to optimise the performance of the IRSA protocols, especially in finite frame length scenarios. In [3], the authors derived an optimal transmission probability distribution for IRSA with a reception capability $K = 2$, which can make the IRSA protocol achieve a higher load threshold. In [4], the authors proposed a Bayesian algorithm to detect the active users for IRSA, in which both the sparsity in user activity and the underlying structure are jointly considered. For the IoTs, Chen *et. al.* analysed the maximum reception rate and the transmit power of users [5], which helps to maximize energy efficiency. In [6], the authors proposed to reduce the number of useless replicas based on a tracking degree distribution control (TDDC) algorithm, which can reduce power waste. Besides the analytical scheme, machine learning schemes are also proposed to enhance the performance of IRSA. In [7], the authors proposed an enhanced Q-learning scheme to improve the performance of IRSA in finite-length frame scenarios, in which the learning scheme is used to optimize the degree distribution. And in [8], Li *et. al.* proposed both a centralised and a distributed Q-learning scheme to maximize the number of successful transmissions in each frame.

In this paper, we propose a new variant of IRSA protocol named Deep-Learning and Sensing-based IRSA (DS-IRSA) to optimise the performance of IRSA in short frame IoT networks, where a sensing phase is added before the transmission phase and users' actions in both phases are managed by a deep reinforcement learning (DRL) method. Specifically, with an extra sensing phase, active nodes can send and attempt to sense short jamming signals with the intent of interacting with other nodes and potentially performing some (weak) form of coordination. And motivated by a very recent work that exploits learning not just to learn the parameters, but to learn the entire methods of interactions [9] or even entire programs, we propose a DRL method based on Proximal Policy Optimization (PPO) [10] to help users design their transmit strategies in both sensing and transmission phases. Our initial goal is to learn a sensing protocol ultimately and entirely through Deep Reinforcement Learning (DRL). In this way, active users can coordinate well with each other and the throughput of the whole system can be well improved. Simulation results show that compared to classical IRSA [1] and our previously proposed Deep-IRSA [11] and Deep-RC-IRSA [12], the DS-IRSA convergence quickly towards the optimal performance of almost 1 packet/slot for small frame sizes and with enough minislots. And it can achieve higher throughput in almost all cases.

II. IRSA Protocol and System Model

The main underlying principle of modern random access protocols are to make each terminal send multiple copies of the same MAC packet, with identical preamble and payload information bits. The payload contains signalling information concerning the temporal positions of the corresponding replicas of each packet. It enables an iterative decoding process with SIC so that the collision issue in slots can be addressed.

As a successful version of the such protocols, IRSA can optimise the probability mass function to maximize the peak throughput performance, which operates as follows: As illustrated in Fig. 1 each user sends ℓ replicas of its packet within the same MAC frame. The repetition rate of each packet is selected randomly from a probability distribution $(\Lambda_i)_{i=2,\dots,L}$, where Λ_i is the probability that the packet is repeated i times. At the end of the frame, the receiver attempts to decode the packets. Each time that a packet is successfully decoded, IRSA will use SIC to remove all replicas of this packet in other slots and check if other packets can be decoded. This process is repeated until no more packets can be decoded.

Fig. 1: An example of IRSA iterative decoding with SIC: 5 users transmit in a frame of 5 slots. First, the packet of user D can be decoded in slot 4. Then, the replica of this packet in slot 5 can be removed, which makes the packet of user B in slot 5 can be decoded. Following this procedure, the packets of the user A can be decoded in slot 3. However, the packets of users C and E form a stopping set as they sent two copies on the same two slots and cannot be decoded.

In this paper, we consider a mMTC network consisting of M users and one single base station (BS). We assume that the wireless channel is divided into frames, and each frame contains N equal length time slots. A collision occurs when several users transmit their packets in the same time slot, which can only be addressed by iterative decoding with SIC.

III. Deep Sensing IRSA (DS-IRSA)

In this section, we present a new IRSA protocol version based on sensing and DRL. In our proposed method, a sensing phase is added before the transmission phase, which provides users with the presence and activity information of other users. Then, with the help of DRL, the actions of active users are defined by a deep neural network, that will learn transmitting strategies and in effect, the entire protocol.

A. Sensing Phase Design

Our proposed extended IRSA frame is illustrated in Fig. 2, in which a sensing phase containing several minislots is added before the transmission phase. The added sensing phase allows users to interact and collect information about the environment. Specifically, all users can decide whether to transmit a jamming signal in each minislot, which is referred to as jamming bursts [13]. Then with the knowledge of their actions and their sensed energy level in the channel, they can adapt their transmission strategy to avoid collisions.

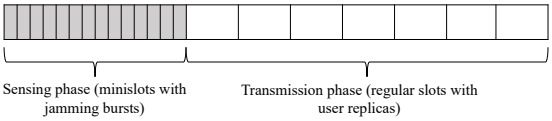


Fig. 2: Extended frame: a sensing phase consists of several minislots added before the transmission phase.

Note that our proposed sensing phase is only among users, which requires all active users to operate in full-duplex mode. In this way, they can sense the channel while simultaneously transmitting a jamming burst. Besides, in an idealised model, we also assume that all users are able to precisely measure the amount of energy of the simultaneously transmitted jamming bursts. Thus, they can detect the number of transmitting users on the minislot, In this way, all nodes in the network share a common knowledge: the outcome of the sensing phase that can be summarized by an integer sequence, with one value for each minislot corresponding to the number of users simultaneously transmitting a jamming burst on this minislot.

Then, in the transmission phase, the IRSA protocol is applied: each node selects some slots and then transmits replicas of its data packet in these slots. Note that no sensing is performed by any user during the transmission phase. As with IRSA, the BS listens to the signals of the transmission phase and decodes the replicas of each user with SIC.

B. Exploiting Sensing Information

In our considered mMTC networks, the initial active users on a frame are undifferentiated. This is because in a mMTC network, active nodes are essentially only a small random subset of a very large set of devices and as a design choice, we assume that they have an identical behaviour (e.g. no slots or "codes" are pre-defined per node). Our proposed sensing phase allows active users to interact, sense the channel, and collect some information. The collected information can be exploited to differentiate users to optimise their transmission strategies in the transmission phase. The system can be modelled as a Markov Decision Process (MDP).

In this paper, we propose a DRL method on the user side, which helps users implicitly synchronise and optimise their transmitting strategy in both the sensing phase and transmission phase to differentiate the activities of active users. As illustrated in Fig. 3, an identical neural network implemented on the user side: sequentially, minislot by minislot, it uses the energy level sensed in the channel in previous minislots and its own actions to decide quickly whether to send a jamming burst or not. Then, in the IRSA transmission phase, another neural network sharing the same intermediate layers takes the same channel usage information collected in the sensing phase as input and returns the user a binary

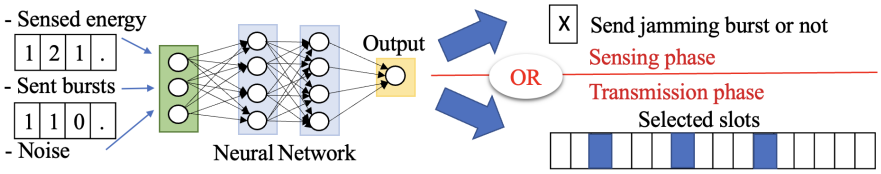


Fig. 3: Overview of the DRL's role in DS-IRSA protocol.

codeword sequence that indicates the selected slots. In this way, the transmission strategies of different active users can be differentiated effectively.[1]

More details about the design and training process will be presented in the next subsection.

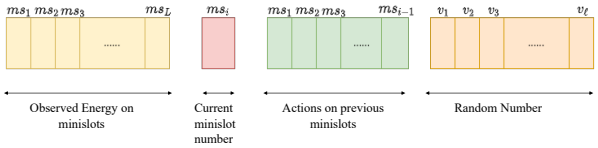


Fig. 4: State computation of DS-IRSA.

C. DRL Method Design

In this part, we present our DRL method design from four major aspects: action, states of MDP, policy update method and model updates.

1) Action: In our case, the same two neural networks that share the same layers are used by users to select their actions in the sensing phase and the transmission phase respectively. In this way, the actions of a user $m \in M$ for one frame can be denoted as A_m corresponding to its transmission strategy in the two phases respectively.

In the sensing phase, the previous output actions by the first neural network for user m can be represented by a vector $A_m^{\text{sens.}} \in \{0,1\}^L$, where L is the total number of minislots. Thus, the action $A_{m,t}^{\text{sens.}} = 1$ means the user m transmitted a jamming burst at minislot t ($t \leq L$), otherwise, it remains silence. Similarly, in the transmission phase, the vector $A_m^{\text{trans.}} \in \{0,1\}^N$ denotes the slot selection decision made by the second neural network for user m to transmit its replicas, where N is the total number of slots.

2) The States: As illustrated in Fig. 4, the state of an agent $m \in M$ is a combination of some observed values and with additional random input values acting as entropy sources:

- The first component of the state is the observed energy on the minislots by the agent, which is represented by a vector of length L of integer values between 0 and M. Recall that L is the maximum number of minislots and M is the total number of agents.
- The second component of the state is the number (index) of the current minislot.
- The third component of the state is the actions that were taken by the user for previous minislots.
- The fourth component of the state is a random vector of a fixed length ℓ. In our simulations, we choose a vector of length $\ell = 10$ that has random values between $[0, 15]$. Classically, it acts as an entropy source to help the model output different codewords in the same state[2].

3) Policy Gradient Methods: We use Proximal Policy Optimization (PPO) [10] to train our proposed DS-IRSA, i.e. the two neural network models shared and used by all users. Policy gradient methods [14] are a type of RL technique that relies upon optimizing parametrized policies concerning the expected return (long-term cumulative reward) by gradient ascent, which is not an action-value method. They learn a parameterized policy that can choose actions without a value function and can be trained typically with a variant of the Policy Gradient Theorem [14, Sect. 13.2].

In typical DRL policy gradient methods, the policy denoted π_θ, is a stochastic policy that takes the observed state s_t from the environment and suggests an action a_t to take as an output. It is given by a neural network model parametrized by a set of coefficients (weights) θ, that determines the probability $\pi_\theta(a_t \mid s_t)$ to select action a_t. During the training, the stochastic gradient ascent is used to iteratively improve the policy, which requires an estimator of the policy gradient from the sampling obtained through the episodes. The improved gradient estimator of the policy gradient by PPO can be given by [10, Eq. 1]:

$$\hat{g} = \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, \hat{A}(t)\right] \tag{1}$$

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, R(t)\right] \tag{2}$$

where $\hat{A}_t$ is an estimation value of the advantage function at time step t, which is the difference between the discounted long-term cumulative reward $G(t)$ and the baseline estimate $b_s(t)$:

$$\hat{A}(t) = G(t) - b_s(t) \tag{3}$$

which reflects how much better the impact of the taken action based on the expectation of what normally should happen ($b_s(t)$). In our case, the reward $G(t)$ is the number of packets that can be decoded after the transmission phase and the discount factor γ is set to 1 for all minislots and the transmission phase.

PPO [10] is an optimisation of the online policy gradient algorithm which, instead, optimises a clipped surrogate objective. Indeed the central optimization objective of PPO can be defined as the expectation of the minimum of two functions:

[1] See [12, Sect. 7.1 and 7.2] for a proposal of a "human-designed" protocol.

[2] observe that otherwise, with 0 minislots for instance, the model would output the same policy for all users, which cannot be optimal. Using $\ell = 10$ and values in $\{0, 1, \ldots 15\}$ is more than a sufficient number of entropy sources so that this is no longer limiting the performance.

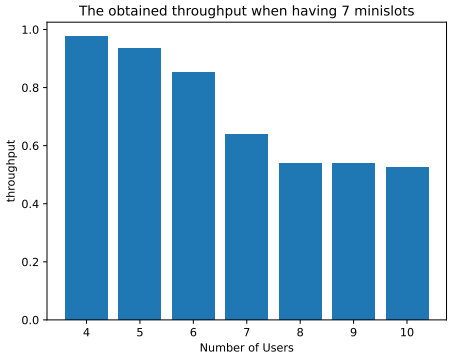


Fig. 5: Throughput for different numbers of users, in case of a sensing phase of 7 minislots.

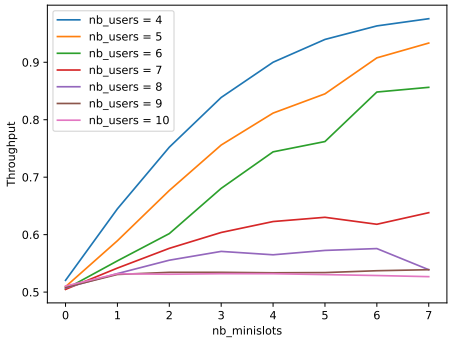


Fig. 6: Throughput for different numbers of users, in case of a sensing phase of an increasing number of minislots.

the normal policy gradient objective $r_t(\theta)\hat{A}(t)$ and a truncated version of $r_t(\theta)$ between $[1-\epsilon, 1+\epsilon]$:

$$L^C(\theta) = \hat{E}_t\left[\min\left(r_t(\theta)\hat{A}(t), \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}(t)\right)\right] \quad (4)$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ and the term $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}(t)$ modifies the surrogate objective by clipping the probability ratio. It removes the incentive for moving $r_t(\theta)$ outside the interval $(1-\epsilon, 1+\epsilon)$. The minimum of the clipped and unclipped objective is taken, so the final objective is a lower bound (i.e., a pessimistic bound) of the unclipped objective.

In this way, the final loss function that is used to train the neural network in PPO is as follows:

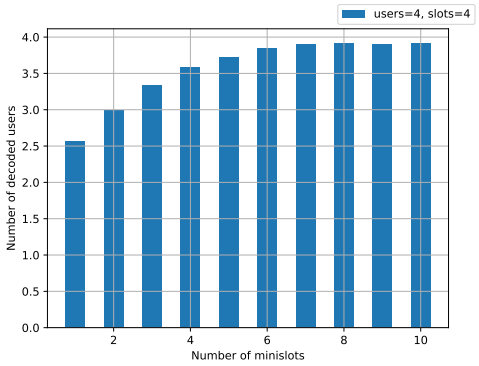
$$L_t(\theta) = \hat{E}_t\left[L_t^C(\theta) - c_1 L_t^V(\theta) + c_2 S\left[\pi_\theta\right] s(t)\right] \quad (5)$$

where $L_t^C(\theta)$ is the clipped PPO objective, $c_1 L_t^{VF}(\theta)$ is used to update the baseline $b_s(t)$. and the last term $c_2 S\left[\pi_\theta\right] s(t)$ is to guarantee that the agent does enough exploration during the training process. c_1 and c_2 are coefficients, S denotes an entropy bonus and $L_t^V(\theta)$ is s a squared-error loss of the value function.

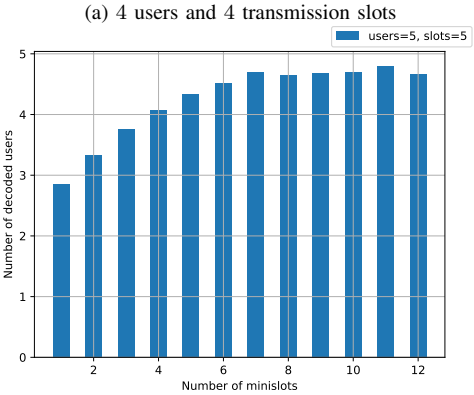
4) Model updates: For the training, the number of users is fixed to M per frame. Training is done through the simulation of episodes, and each episode corresponds to one frame. Each episode is divided into steps. Each step accounts for an action taken by one user. A full episode is completed when all the agents take their final actions, which is the slot selection for the transmission phase. At the end of each episode, the reward is calculated. Every time a number of episodes have been done, the weights of the models are updated according to PPO.

IV. Simulation Results

In this section, we present the simulation results of our proposed learning method. We developed our own IRSA simulator with Python. Our simulator allows for constructing frames and generating user transmissions. IRSA iterative decoding is performed using a collision model after which the decoding information is obtained, e.g. the number of decoded users. The (normalized) throughput is the average packet delivery ratio. For the DRL method, we use the implementation "stable baselines 2" derived from OpenAI Baselines. and the neural network models implementing policy and the baseline value function estimate are with the default parameters, which contain 2 layers of 64 neurons and approximately 4000 weights.



(a) 4 users and 4 transmission slots



(b) 5 users and 5 transmission slots

Fig. 7: Impact of the number of minislots in the sensing phase on the number of decodable users in the transmission phase.

The obtained throughputs after applying our proposed DS-IRSA are given in Table. I, where each value is an average

TABLE I: The obtained average number of decoded users of DS-IRSA with 7 minislots using our DRL method.

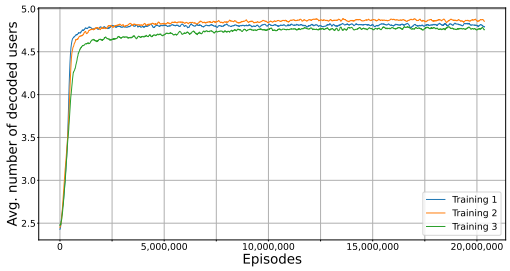
U/S	2	3	4	5	6
2	1.982	1.986	1.986	1.990	1.994
3	1.971	2.996	2.997	2.979	2.969
4	1.967	2.73	3.84	3.73	3.77
5	1.963	2.80	3.463	4.168	4.233
6	1.937	2.76	3.30	3.152	3.873

of 10 simulations. note that the number of minislots in the sensing phase is fixed to 7 and the two phases were trained simultaneously. Compared with our previous work [12, Table.1], it can be observed that with an extra sensing phase, throughputs in all cases are increased, which suggests that the learning in the sensing phase can help the DRL method to differentiate users' action. It can also be observed that with 7 minislots, in cases of 2 and 3 users, even in short frames, almost all packets can be decoded and the protocol is close to its maximum throughput. However, from 4 users to 6 users, the achievable throughput gradually decreased. One reason is that as the number of users increases, 7 minislots are probably not sufficient to synchronize all users during the transmission phase, so more collisions occur and not all packets can be decoded. Another reason is that the complexity of the learning process increases rapidly with the number of competing users, which leads to inaccurate exploitation of the collected information during the sensing phase and can affect the slot selection decisions in the transmission phase.

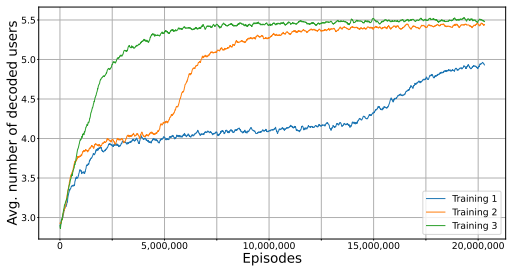
In Fig. 5, we show the obtained throughput of DS-IRSA in case of having 7 minislots in the sensing phase, while we varied the number of competing users from 4 to 10. Note that each bar in this figure (throughput value) is obtained after 20 million of episodes, and each value in the figure represents the mean of the obtained values. These results emphasize that the learning complexity increases with an increasing number of users, which makes the throughput dropdown.

In order to testify to the impact of sensing in short frame scenarios, we show the achievable throughput for different numbers with different numbers of minislots in Fig. 6. For each scenario, we performed almost 20 million of learning episodes. It can be observed that an increase in the number of minislots in the sensing phase results in a notable increase of the throughput in the case of 4, 5 and 6 competing users. This effect has been mathematically proven in [12, section 9.3] for the case of two users. However, as the number of users increases, especially for cases with 9 or 10 users, the increased learning complexity prevents convergence of the learning of a good sensing protocol and limits the sensing gain, which makes the throughput only slightly higher than 0.5.

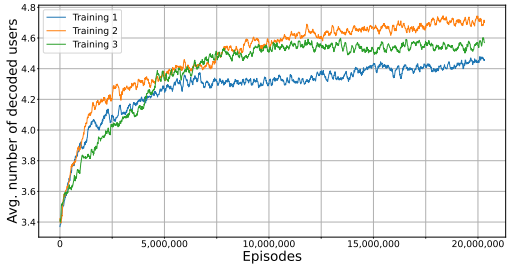
Figure 7 shows more clearly the impact of the minislot number on the throughput, where we consider two scenarios with 4 and 5 competing users respectively. In cases with more than two slots and users, it can be observed that using more minislots in the sensing phase can help increase the achieved throughput. In both cases, in the beginning, only more than half of the packets can be successfully decoded. As the number of minislots increases, especially with more than 6 or 7 minislots, almost all packets can be decoded. In this way, we can conclude that an extra sensing phase with minislots can help increase the throughput in the transmission phase by differentiating users' transmission strategies, especially for mMTC networks. Afterwards, we discuss the convergence of or DRL method. Fig. 8 shows the convergence of the learning process of DS-IRSA in different scenarios. For each scenario, we performed three different trainings with different



(a) Convergence with 5 users and 5 slots and 10 minislots.



(b) Convergence with 6 users and 6 slots and 10 minislots.



(c) Convergence with 7 users and 7 slots and 8 minislots.

Fig. 8: Training convergence of DS-IRSA for different scenarios.

seeds and trained each with 20 million episodes. Note that the training starts with random initial weights depending on the used seed. Fig. 8a shows a 5-users 5-slots scenario. It can be observed that as the number of minislots is sufficient to find an optimized transmission strategy for the 5 competing nodes, almost all 5 packets can be decoded and the training model converges quickly. On the contrary, in Fig. 8b and 8c, the learning convergence towards the maximum throughput is slower as the number of competing users is larger. We can also observe that in both cases, not all trainings can converge within 20 million episodes and not all packets can be decoded (5.5 in the case with 6 users and only 4.6 in the case with 7 users). Thus, experimentally, with the current implementation and training procedure, adding more minislots to the sensing phase for both scenarios will not guarantee convergence, as the problem of finding the optimal protocol becomes more complicated.

Finally, we represent the decision-making procedure of our learned DS-IRSA protocol as a tree. After the training process, we explore the actions selected by the trained DRL model depending on states. Since there is a random vector as part of

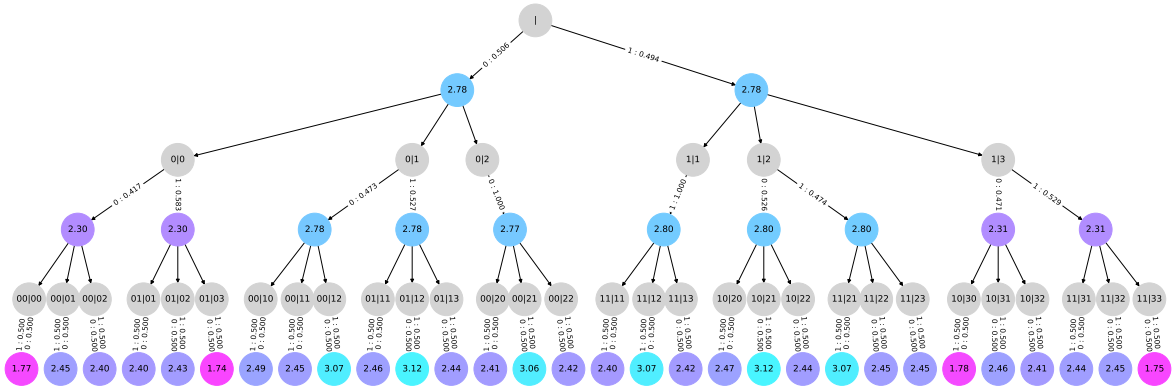


Fig. 9: Decision-making process of the learned DS-IRSA protocol with Deep Reinforcement Learning.

the input, we perform an average of the outputs with different random vectors (each trial corresponds to a simulation). Fig. 9 shows a case that 3 users competing for 3 slots and with 2 minislots in the sensing phase. The grey circles are the observed states, which consist of the previous action (left) and the observed energy in the corresponding minislots (right). The coloured circles represent the estimate of the value function, that is, an estimate of the expected cumulative returns value in the given state. In our case, it corresponds to the number of decoded users. The labels of the arrows between the circles denote the probability to take each action (e.g "$0 : 0.506$" and "$1 : 0.486$" at the top, indicates that sending a jamming burst with a probability of 0.486 and staying silent with a probability of 0.506). Each value is an average of 100 simulations. Due to the intrinsic randomness of random access, several cases may occur in the sensing protocol. In some cases, it can perfectly coordinate users, they select proper slots and all three users' packets are decoded with IRSA (light blue points). But in other cases, coordination is too weak and we are back to the performance of classical IRSA without sensing (pink points). More details and analysis can be found in [12, p36].

V. Conclusion

In this paper, we proposed DS-IRSA, Deep Learning Sensing-based IRSA protocol which is composed of two phases: a sensing phase, where the nodes can sense the channel and send short jamming signals, followed by a classical IRSA transmission phase. In both phases, active users' actions are managed by two neural networks that share the same layers. In this way, active users can coordinate well with each other and the throughput of the whole system can be well improved. Simulation results show that our proposed protocol can achieve higher throughput than classical IRSA protocol or other optimized IRSA variants through deep learning. And in cases of small frame sizes and with enough minislots, our proposed deep learning-based scheme convergence quickly towards the optimal performance of almost 1 packet/slot.

VI. Acknowledgments

This work was supported by the 5G-mMTC project, which was funded by the French government as part of the "Plan de Relance et du Programme d'investissements d'avenir"

References

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

[2] G. Interdonato, S. Pfletschinger, F. Vázquez-Gallego, J. Alonso-Zarate, and G. Araniti, "Intra-slot interference cancellation for collision resolution in irregular repetition slotted aloha," in *2015 IEEE International Conference on Communication Workshop (ICCW)*, 2015, pp. 2069–2074.

[3] Z. Chen, Y. Feng, C. Feng, L. Liang, Y. Jia, and T. Q. S. Quek, "Analytic distribution design for irregular repetition slotted aloha with multi-packet reception," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 1, pp. 1360–1365, 2023.

[4] C. R. Srivatsa and C. R. Murthy, "User activity detection for irregular repetition slotted aloha based mmtc," *IEEE Transactions on Signal Processing*, vol. 70, pp. 3616–3631, 2022.

[5] Z. Chen, Y. Feng, Z. Tian, Y. Jia, M. Wang, and T. Q. S. Quek, "Energy efficiency optimization for irregular repetition slotted aloha-based massive access," *IEEE Wireless Communications Letters*, vol. 11, no. 5, pp. 982–986, 2022.

[6] 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)*, 2019, pp. 1–6.

[7] E. Nisioti and N. Thomos, "Fast q-learning for improved finite length performance of irregular repetition slotted aloha," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 2, pp. 844–857, 2020.

[8] Y. Li and K.-W. Chin, "Energy-aware irregular slotted aloha methods for wireless-powered iot networks," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 11 784–11 795, 2022.

[9] E. Pesce and G. Montana, "Improving Coordination in Small-Scale Multi-Agent Deep Reinforcement Learning Through Memory-Driven Communication," *Machine Learning*, pp. 1–21, 2020.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347*, 2017.

[11] I. Ayoub, I. Hmedoush, C. Adjih, K. Khawam, and S. Lahoud, "Deep-IRSA: A Deep Reinforcement Learning Approach to Irregular Repetition Slotted ALOHA," in *10th IFIP International Conference on Performance Evaluation and Modeling in Wireless and Wired Networks (PEMWN)*, 2021, pp. 1–6.

[12] I. Hmedoush, C. Adjih, and P. Mühlethaler, "Deep Learning, Sensing-based IRSA (DS-IRSA): Learning a Sensing Protocol with Deep Reinforcement Learning," INRIA-SACLAY, Research Report RR-9479, Jun. 2022.

[13] H. Al-Mefleh and O. Al-Kofahi, "Taking Advantage of Jamming in Wireless Networks: A Survey," *Computer Networks*, vol. 99, pp. 99–124, 2016.

[14] 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.