A lightweight, fully-distributed AI framework for energy-efficient resource allocation in LoRa networks

Antonio Scarvaglieri
University of Catania
Catania, Italy
antonio.scarvaglieri@phd.unict.it

Sergio Palazzo
University of Catania
Catania, Italy
sergio.palazzo@unict.it

Fabio Busacca
University of Catania
Catania, Italy
fabio.busacca@unict.it

ABSTRACT

As the Internet of Things (IoT) continues to grow rapidly, efficient resource utilization is crucial for the sustainability and performance of IoT networks. In this context, LoRa technology, known for its low-power, long-range communication, has become popular for IoT applications. However, the limited energy and spectrum resources in Long Range (LoRa) networks present challenges in achieving optimal network performance in dense deployments. In particular, existing centralized approaches, such as LoRa Adaptive Data Rate, may fail to scale up to the large networks typical of IoT scenarios. To address all of these issues, we propose a smart, **fully-distributed** resource allocation scheme based on multi-agent cooperative Q-learning approach. Simulations results prove that our approach improves the Packet Delivery Ratio (PDR) and reduces the energy consumption of up to **43%** as compared to fixed SF and random non-smart strategies, respectively, while also keeping the decision process at a device-level, with no centralized entities involved.

KEYWORDS

LoRa, IoT, Reinforcement Learning, LPWANs, Q-Learning, Distributed Decisioning

ACM Reference Format:
Antonio Scarvaglieri, Sergio Palazzo, and Fabio Busacca. 2023. A lightweight, fully-distributed AI framework for energy-efficient resource allocation in LoRa networks. In *2023 IEEE/ACM 16th International Conference on Utility and Cloud Computing (UCC '23), December 4–7, 2023, Taormina (Messina), Italy*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3603166.3632564

1 INTRODUCTION

Recent studies suggest that the number of Internet of Things (IoT) devices will exceed 24 billion by the year 2030, resulting in a potential annual revenue of up to 1.5 trillion [1]. With the rapid expansion in the number of IoT devices, both the industrial and academic sectors are dedicating substantial efforts to designing, developing, and assessing the performance of IoT-based technologies. In this regards, there is a specific emphasis on Low-Power Wide Area Network (LPWAN) technologies [2–4].

What sets LPWAN communication protocols apart is their ability to strike low-power and long-range communications in exchange for low transmission rates. For these reasons, LPWAN communications are a key-enabler for energy-constrained IoT scenarios, such as city-wide pollution and air quality monitoring, especially if compared to other wireless protocols like Wi-Fi and LTE, among others. In particular, Long Range (LoRa) has emerged as one of the most prominent LPWAN technologies. LoRa operates in unlicensed spectrum bands and employs a proprietary modulation derived from Chirp Spread Spectrum (CSS); in such a way, it is able to achieve an impressive coverage range of 2-5 kilometers in urban areas and of 15-30 kilometers in open space [5] with a relatively low Transmission Power (TP) (for instance, the TP is as high as 25 mW for the Semtech SX1272 transceiver [6]). In turn, LoRa supports only low to moderate data rates, ranging from about 165 bps to 5.5 kbps in the EU region, for a Spreading Factor (SF) of 12 and 7, respectively [7]. Given the scarce transmission performance, the optimal allocation of the communication resources proves of utmost importance.

In a typical LoRa network setup, one possible way to achieve this result is to exploit the Adaptive Data Rate (ADR) mechanism [8]: a central node, i.e., a LoRa Gateway, selects and sends the optimal transmission parameters for each node based on the node distance/the channel conditions. However, such a mechanism can cause the burdening of the network as the number of connected devices increases. Even worse, the ADR mechanism does not take into account the Intra Spreading Factor interference, leading to a possible increase in the overall packet loss of the network. For these reasons, in this paper we propose an innovative and energy-efficient resource allocation strategy based on Q-Learning [9], a Reinforcement Learning (RL) algorithm. According to our strategy, each LoRa device runs a lightweight Q-Learning agent, and is therefore able to make autonomous, smart decisions over its own SF configuration. In particular, during each learning round, the central server broadcasts a single message containing the reward value of the RL environment. The LoRa nodes can accordingly exploit this value to update their Q-Table, which contains information on how the agent should act based on their current state.

These features make our SF allocation strategy **fully-distributed** and **scalable**, as opposed to traditional, centralized LoRa resource allocation schemes, like the aforementioned ADR mechanism. Moreover, the usage of table-based Q-Learning, as opposed to Deep Q-Learning, makes our algorithm suitable even for heavily-constrained, simple IoT devices. As described later in this paper, we also resort to a threshold mechanism to keep the size of the state-action table suitable for the limited memory of the IoT devices.

In the following are the main contributions of our work:

This work was partially supported by MUR, Italy, under the PRIN project "Liquid_Edge". The research was also partially supported by European Union (NextGenerationEU) through the MUR-PNRR projects SAMOTHRACE (ECS00000022) and RESTART (PE0000001).

UCC '23, December 4–7, 2023, Taormina (Messina), Italy

ACM ISBN 979-8-4007-0234-1/23/12.
https://doi.org/10.1145/3603166.3632564

- We designed a lightweight RL framework suited for a **fully-distributed**, **highly scalable** LoRa SF allocation;
- We ran a numerical evaluation campaign based on the LoRaSim simulator. We compared the proposed approach with non-smart allocation strategies, namely fixed-SF approaches, ADR algorithm, and a random allocation approach. The results show how our smart allocation scheme is able to improve the average Packet Delivery Ratio (PDR) as compared to fixed SF strategies and to decrease the energy consumption of about **43%** as compared to the random allocations.

The remainder of this paper is organized as follows. Section 2 explores the related works about SF allocation in LoRa networks. Section 3 introduces the System Model, while Section 4 describes the simulation setup; Section 5 focuses on the analysis of the simulation results. Finally, Section 6 draws some conclusions about the work.

2 RELATED WORK

Several studies have been conducted on Spreading Factor allocation in LoRa networks, in both centralized and distributed assignment.

In the first group, the baseline is set on the ADR [8] algorithm, proposed by SemTech and already included in LoRaWAN specifications. The ADR selects a SF based on the distance between the gateway and the node. However, this approach appears to be poor performing, especially as the number of the end-devices grows [10].

The authors in [4] presented a Equal-Interval-Based (EIB) allocation scheme in which the network area is subdivided in concentric circles of the same width. Nodes belonging to the same circle will use the same SF. However, collision problems could be more frequent if the nodes are very close to each other. All of these allocation schemes are however suited for static environments, where the number of nodes and their location is fixed. In case of changes in the network topology and/or the number of nodes, new parameters should be computed and sent back to each connected devices, overburdening the network. Among the centralized schemes, [11] and [12] proposed centralized dynamic allocation schemes based on Deep Reinforcement Learning (DRL), in which the AI algorithm is run respectively on the the network server and on the gateway. The latter provides support for the mobility of end-devices, however the gateway is supposed to have unlimited power and computational resources. On the other hand, distributed allocation schemes for SF are mostly based on artificial intelligence algorithms. Among the distributed allocation schemes is [13], where the author propose LoRa-MAB, an allocation method based on Multi Armed Bandit (MAB), in which each device is considered as an intelligent agent capable of choosing channel, SF and TP values in order to minimize the collision rate. However, after an agent is initialized the action space is reduced based on its current distance from the gateway, and this could lead to sub-optimal solutions in case of non-fixed devices. Hamdi et al. [14] proposed an allocation method based on DRL which takes care of frequency and SF selection. However, applying the proposed schemes on resource-constrained LoRa nodes is quite challenging as complete network information, packet acknowledgments and high computing power are needed. The approach proposed by Yu et al. [15], instead, uses Multi-Agent Q-Learning algorithm to assign SF and TP to each node in the network. Each agent can select between 6 SFs and 5 TP values and this results in a growth of the action space, and on the Q-Table size consequently. The state space size, instead, is equal to 2^N, and so increases together with the number of LoRa devices. Moreover, the reward is different for each node and thus there is a need for end-to-end communications with the server in order to update the values on the Q-Table. This leads to more spectrum and energy consumption, reducing the overall efficiency of the network. Hong et al. [16] propose three algorithms for SF allocation in LoRa Networks. One of them, LR-RL, is based on temporal-difference based RL algorithms, however it does require that the gateway sends packet acknowledgments in order to work.

The approach proposed in this paper is part of the distributed allocation schemes group. Unlike [14], it does not involve the usage of neural networks, which do not fit to resource-constrained IoT devices. Instead, like [15] it relies on Multi Agent Q-Learning in order to assign the SF only to all agents: the adopted strategy is strictly cooperative, so the reward value, which is common to all agents, is broadcast to all nodes in the coverage area of the gateway.

3 SYSTEM MODEL

This section focuses on the mathematical model of the considered LoRa network and of the SF allocation problem. The system architecture reflects a typical LoRaWAN network, and consists of M gateways and N nodes, with the latter randomly located inside urban/densely populated areas within a certain radius from the first. The network nodes send uplink packets at random time intervals following unslotted ALOHA Medium Access Control (MAC) protocol. Accordingly, inter-SF and intra-SF interference is possible. We assume that the nodes are powered with batteries, like Class A LoRaWAN devices, and that the gateways do not interfere with end-devices uplink transmissions.

Generally speaking, the energy consumption of a LoRa node depends on several transmission parameters, namely: Spreading Factor, Transmission Power and Code Rate. In order to devise a resource allocation framework that is effective and scalable, yet suitable for low-power and resource-constrained devices, we choose to focus on the choice of the optimal SF for each node. In fact, among the aforementioned transmission parameters, the SF is the one with the biggest influence on the LoRa communication performance [16]. First, an optimal choice of this parameter allows the devices to exploit the quasi-orthogonality of the different SFs, enabling multiple simultaneous transmission on the same physical channel. Moreover, the choice of the SF directly influences the receiver sensitivity and the Time-on-Air (ToA) duration, as shown in Table 1. As the SF increases, the coverage range is accordingly extended, thus lowering the node outage probability. Unfortunately, higher SFs are associated to a larger ToA, which implies: i) an increase in the energy consumption; ii) a longer occupation of the logical channel that corresponds to the selected SF, and, therefore, a bigger probability of colliding transmissions. For all these reasons, our study focuses on the SF selection in a fully distributed fashion.

In such a view, the LoRa nodes equipped with the proposed solution attempt to strike a certain trade-off between their energy consumption and the communication reliability, measured in terms of the Packet Delivery Ratio (PDR), i.e., the ratio between the number of packets delivered and of packets sent. In principle, picking the lowest SF possible is the ideal choice, as this always yields the

Table 1: ToA and sensitivity values at different SFs for a packet of 64 bytes, bandwidth fixed at 125 kHz, CR = 4/5

SF	Sensitivity	ToA	SF	Sensitivity	ToA
7	-123.0 dBm	41 ms	10	-132.0 dBm	288 ms
8	-126.0 dBm	72 ms	11	-134.5 dBm	577 ms
9	-129.0 dBm	144 ms	12	-137.0 dBm	991 ms

highest data-rate and the lowest ToA, i.e., the least energy consumption. However, the bigger the number of nodes selecting the lowest SFs, the bigger is the intra-SF interference suffered by those nodes. Hence, a node may either: i) select the lowest SFs to benefit from a higher data rate and a lower energy consumption, but accordingly suffer from a more crowded channel; ii) pick a higher SF, where the intra-SF interference is less severe, at the expense of a higher energy consumption and lower data-rate.

Let us now discuss the learning model. First, each node is modeled as an independent agent in a Multi-Agent Reinforcement Learning (MARL) problem. The scenario can be viewed as a non-episodic one since there is no terminal state that marks the end of communication within the network. The set of actions that each agent can take, or *action space*, involves the selection of one among the six available SFs {7, 8, 9, 10, 11, 12}. Accordingly, the action space size is $A = 6$. The *observation space* is instead defined by a six-element array, where each value denotes the utilization rate of a specific SF.

As the computational and storage resources of a LoRa node are limited, we applied a lightweight RL algorithm, Q-Learning, which does not involve the use of neural networks. It estimates the Q-values of state-action pairs and learns an optimal policy by iteratively updating the Q-values based on observed rewards and state transitions. Those values are stored in a 2D matrix, called Q-table, in which each cell corresponds to a state-action value pair. Its dimension should be limited, as typical LoRa end-devices are equipped with constrained storage memory. Nonetheless, if the whole number of nodes that are actually employing a SF were used to represent this status, that would cause an exponential expansion of the state space and, subsequently, of the size of the Q-table. Indeed, in such a case, tracking the occupancy state of each SF requires the usage of a 6-element array, each in the range between 0, i.e., no nodes uses that SF. and N, i.e., all nodes picked that SF. The state-action table would accordingly be $A \cdot (N+1)^A$. Given the limited availability of storage on IoT devices, this makes such a solution infeasible, especially for a large number of devices N, as commonly happens in typical IoT settings. Therefore, in order to reduce the state space, we resort to a threshold system, where the number of thresholds, and, therefore, of states, can be adjusted by modifying a parameter T. For instance, we can select a number of thresholds equal to $T = 3$, where the level of occupation of a specific SF is accordingly defined as "Low", "Moderate", and "High", depending on the fraction of nodes that picked that SF. In such a case, the table has instead a size of $A \cdot (T)^A$. Therefore, depending on the choice of T, it can be individually managed and stored inside each single node. The choice of the parameter T obviously strikes a trade-off between the modeling accuracy and the table size. A bigger T improves the granularity of the state representation, yet yields a bigger state-action table, and vice versa.

Another choice regards whether to keep a local table for each agent or, on the other hand, to have a single one common to all agents. However, the latter is not an option in this scenario because there are network constraints on LoRa devices that cannot satisfy the requirement of constantly downloading, updating and sending a centralized Q-Table. Moreover, as the number of nodes that shares the same table increases, concurrency access problems may arise. Therefore, the best choice is keep an individual Q-table that will be stored and updated autonomously by each agent.

We propose two ways to design the reward function: the first is based on an individual reward function that leads to a competitive behavior in the interaction between the agents, where each node aims at maximizing its own cumulative reward; the second, instead, uses a shared reward function and relies on maximizing not only the reward of a single agent, but of all the agents instead. In the competitive case, the reward function $r_n(t)$ of each device $n \in N$ can be defined as follows:

$$r_n(t) \ = PDR_n(t) - \beta * \frac{ToA_n(t) - ToA_{SF7}}{ToA_{SF12} - ToA_{SF7}} \tag{1}$$

where the first element is the PDR experienced by the node, while the second element is the normalized ToA of the transmitted packets and directly influences the energy consumption of the node. Finally, β is a weighing parameter. In particular, adjusting β allows for a flexible trade-off between energy consumption and network reliability. Higher values of β prioritize energy saving, potentially at the cost of decreased reliability, while lower values of β emphasize network reliability at the expense of an increased energy consumption. The optimal value of β depends on the specific requirements and constraints of the network and should be carefully chosen based on the desired balance between energy efficiency and reliability. Instead, the reward in the cooperative scenario is common to all agents and can be defined as follows:

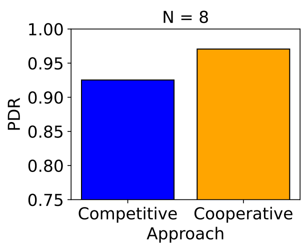
$$r(t) \ = PDR_{avg} - \beta * \frac{ToA_{avg} - ToA_{SF7}}{ToA_{SF12} - ToA_{SF7}} \tag{2}$$

In order to properly balance the exploration-exploitation tradeoff, we resort to the "ϵ-greedy" policy [17], where the agent's probability of experimenting at any time step is determined by a constant value ϵ, whose value ranges from 0 to 1. For all timesteps, each agent selects a random action, i.e., explores, with probability ϵ. Otherwise, it selects the best-known action, i.e., exploits the information he gained throughout the learning process, with a probability of $1 - \epsilon$.

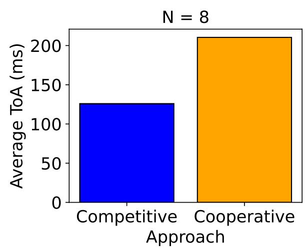
In this study, we choose to adopt an exponential decay. In such a way, the ϵ value decays exponentially as the agent gains more knowledge about the state space, promoting the exploitation of acquired knowledge. A discount factor γ is used to update the values inside the Q-Table. Finally, the Q-Table is initialized by filling all cells with a fixed value. Note how this reward refers to the average PDR and ToA in the system.

4 SIMULATION SETUP

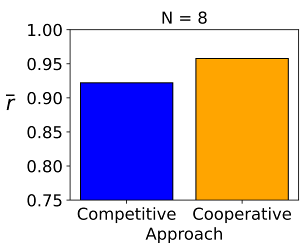
This section describes the main simulation setup for our scenario. In particular, we first introduce all the simulation parameters, to then focus on the comparison between different allocation strategies and our framework based on Q-Learning. Among the various LoRa network simulators available, we selected LoRaSim, a discrete-event simulator, which was originally developed to analyze the scalability of LoRa networks by the authors in [18]. LoRaSim allows the user to deploy LoRa end devices and gateways within a two-dimensional grid. LoRaSim implements the log-distance path loss model, which perfectly fits the assumptions made in Section 3. The source code of the simulations has been made available on GitHub. [1]



(a) PDR

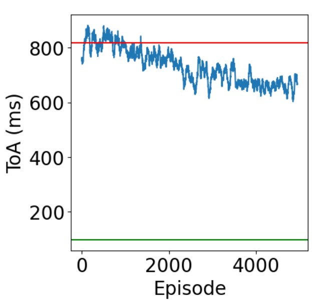


(b) ToA

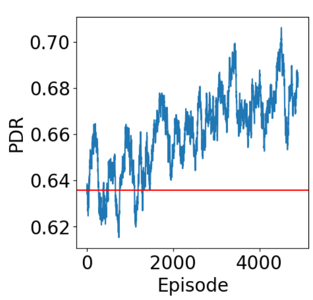


(c) Reward

Figure 1: Network performance comparison between competitive and cooperative approach



(a) Average ToA



(b) Average PDR

Figure 2: Network performance improvement as the learning process goes on. The red line indicates values for fixed SF12, the green line instead stays for fixed SF7

Let us now discuss the simulation parameters. Specifically, the experiments involve a variable number N of class A end devices and $M = 1$ gateway which are located inside a grid of size `maxX` X `maxY`, with `maxX` = 500 meters and `maxY` = 500 meters as well; the first are positioned randomly within a maximum distance `loc_range` = 64 meters radius from the center. For what concerns the log-distance model, we set $\gamma = 2.08$, $d_0 = 40$ meters, $L_{PL}(d_0) = 127.41$ dB. As regards the communication phase, at each timestep each individual device within the network can transmit a packet consisting of 50 bytes, with a probability `tx_prob`. The center frequency is 868.3 MHz, while the TP value is fixed at 14 dBm. Moreover, the communication bandwidth is 125 kHz, and the CR is 4/5. The minimum time interval between subsequent transmissions of the same node is set to $\lambda_n = 100$ seconds. Once λ_n seconds have passed since the last transmission, the device may transmit in a random time interval in the range $[0, \delta_n]$, with $\delta_n = 12$ seconds. The learning phase consists of episodes, in which the number of discrete timesteps for the simulation is defined as an integer value `max_steps` = 4000. It is important to mention that throughout the course of the experiments, the locations of both the gateways and the devices remain fixed. Consequently, the consideration of device mobility has been omitted from the study. Additionally, all the cells of the Q-table were initialized with a value of 1. Epsilon is scheduled to decay exponentially from $\epsilon = \epsilon_{max} = 1$ to $\epsilon = \epsilon_{min} = 0.2$. Instead, the learning rate α will decay following a step-decay scheduler, which halves its value every 1000 episodes. A summary of all the used parameters is presented in Table 2.

T varied among different tests. Inside the reward function, we set $\beta = 0.25$. This choice has a precise effect on the PDR-ToA tradeoff: the network nodes will prefer a bigger PDR over smaller ToAs. The Q-Table optimization algorithm runs for 4000 episodes.

5 NUMERICAL RESULTS

This section thoroughly describes the numerical results obtained throughout the simulation campaign. First, preliminary comparative tests between the cooperative and competitive approach have been run with $T = 4$ and $N = 8$. The results are shown in Figure 1, and confirm that the cooperative approach yields the best overall performance in the considered LoRa network scenario.

Note how the cooperative approach achieves a higher PDR, yet a bigger ToA (Figures 1a,1b) with respect to the competitive one. This behavior yields a bigger value of the reward for the cooperative case, coherently with the choice of $\beta = 0.25$ (Figure 1c).

This outcome clearly shows how the cooperative approach is more effective in reaching the desired PDR-ToA trade-off. Most notably, this is true even for a network with as few as 8 nodes, that is, with a reduced competition among the nodes, let alone for networks with a bigger number of nodes.

For this reason, we choose to leave out the competitive approach, and instead focus on the cooperative one for further analysis.

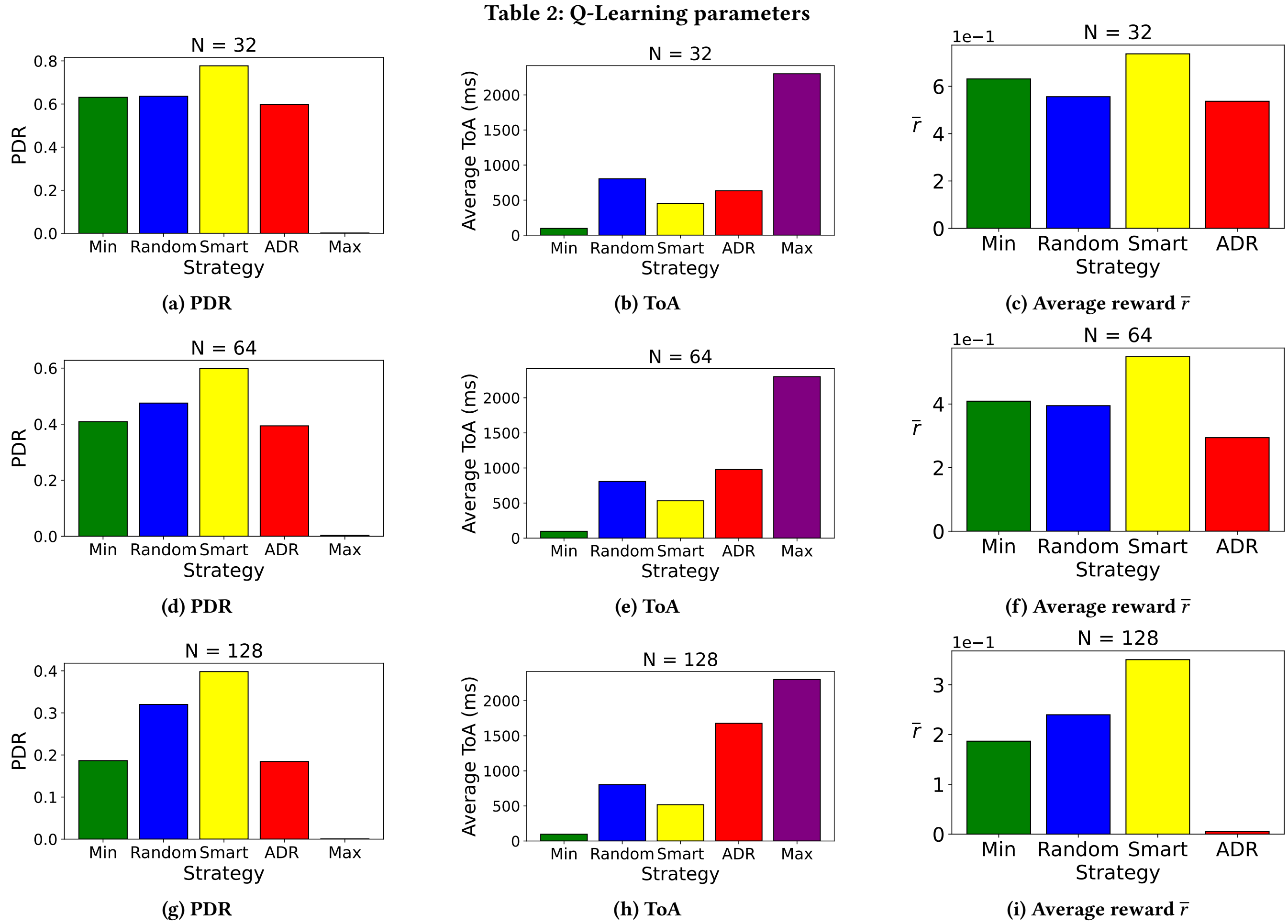
After these preliminary experiments, we run a series of tests with $T = 4$, $N = 32$ in order to gain insights into the learning progress of the agents on a per-episode basis. The learning process leads to higher rewards obtained by the agents as they continue to exploit their knowledge. Accordingly, Figures 2a and 2b respectively reveal a noticeable decline in the average ToA and an increase in the PDR as the episodes progress.

A further test focused on the comparison of our fully-distributed, smart allocation framework with 3 non-smart RA strategies and the ADR algorithm. The first two strategies (Min and Max) involve the usage of a fixed SF, respectively the 7th and the 12th. According to the third strategy (Random), each agent randomly picks its own SF. The ADR, instead, initializes the end updates the SF for each agent after every 100 packets received. This specific test has been

[1] https://github.com/absentio/q-lora

Parameter	Value	Description
n_episodes	4000	Number of episodes for training the agents
T	4, 6, 8	Number of maximum states that specifies the occupancy rate for each SF.
ϵ_{max}	1	Value of epsilon at the start of the episode
ϵ_{min}	0.2	Minimum value for epsilon at the end of the episode
γ	0.99	Discount factor for future rewards
α	0.01	Learning rate starting value
β	0.25	Energy consumption vs PDR trade-off factor

Table 2: Q-Learning parameters



(a) PDR **(b) ToA** **(c) Average reward $\overline{r}$**

(d) PDR **(e) ToA** **(f) Average reward $\overline{r}$**

(g) PDR **(h) ToA** **(i) Average reward $\overline{r}$**

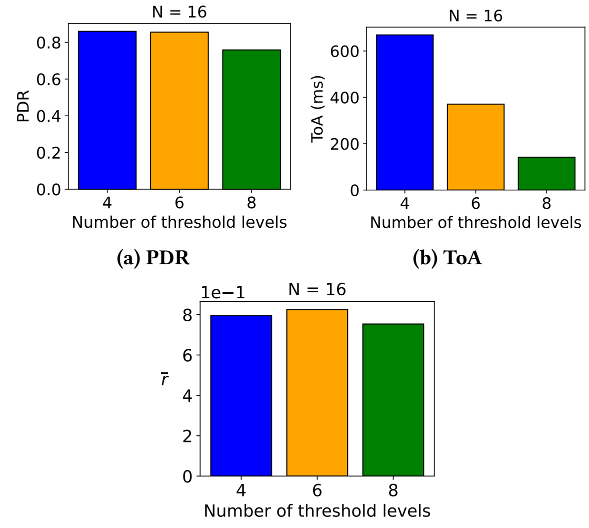
Figure 3: SF allocation strategies comparison when N = 32, 64, 128

run with $N = 32, 64, 128$ and $T = 4$. In all cases, our smart allocation framework achieves the best performance in terms of PDR, as depicted in Figures 3a, 3d, and 3g. Moreover, our smart strategy also achieves the second lowest ToA, as shown by Figures 3b, 3e, and 3h. The lowest ToA is instead obtained by the Min strategy, that, however, represents the lower-bound case (as discussed in Section 3, SF7 yields the smallest ToA). Accordingly, our framework also maximizes the system reward, as visible in Figures 3c, 3f, 3i. Also, note how, as the number of network devices increases and the other parameters remain fixed, the achieved PDR worsens in all cases. This is especially evident for the "Max" strategy, i.e., the case with the longest ToA. In fact, a bigger number of devices increases the probability of intra-SF collisions, with a direct effect on the average PDR.

Figure 4 depicts the energy consumption of our smart allocation framework and of the random and ADR strategies. Note how our framework achieves a decrease in the energy consumption of up to about 43% with respect to the random strategy. Let us remark how this result is obtained in a decentralized way, without introducing a significant overhead in the network. In fact, the reward value and SF occupancy levels, i.e., the input data needed for the algorithm to run, can be broadcast in a single packet to all the nodes in the coverage area of the gateway. This packet is sent every λ_n, and depends on the application running above the LoRa physical level.

Furthermore, we analyzed the effect of different values of the threshold T over the network performance. In these regards, we run several experiments with $N = 16$, and T equal to 4, 6, or 8. The corresponding Q-Tables respectively have a size of 192 KB, 2.1375 MB and 12 MB. On one hand, the need for higher memory resources to store a bigger table can be a key factor in a resource-constrained scenario like the one analyzed. However, an increase in the number of threshold can lead to an increased granularity in the description of the system state and, therefore, in better performance results. The simulation results are shown in Figure 5. Interestingly, the reward function is maximized for $T = 6$, and not for $T = 8$, i.e, the case with the biggest granularity. This result suggests that $T = 6$ achieves the best trade-off between the granularity and the exploration space. In fact, 4000 episodes are not sufficient to accurately explore the larger state-action space produced by the $T = 8$ threshold.

Finally, we analyzed how the node distribution across the different SFs. The results depicted in Figure 6 show how the devices mostly select the lower SFs for transmitting their packets. In fact, SF7 has been used by 41.41% of the overall packets in a simulation with N = 8 end devices and T = 6 threshold levels. While the nodes that select the lower SFs can benefit of lower energy consumption and high data rate, the other end devices which select higher SFs instead can take advantage of the lighter network load, less intra-SF interference and higher range of communication.



(a) PDR **(b) ToA**

(c) Average reward $\bar{r}$

Figure 5: Network performance for different T values

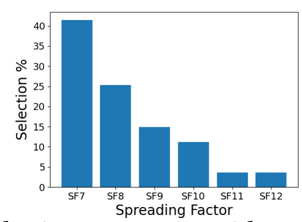


Figure 6: SF selection percentages with N = 8 nodes and T = 6 threshold levels following the proposed allocation method

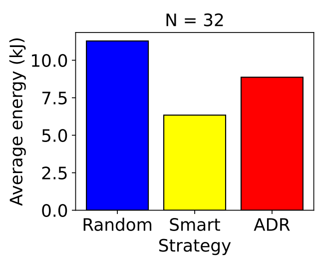


Figure 4: Energy consumption comparison for N =32

6 CONCLUSIONS

In this paper, we proposed a fully-distributed, smart Spreading Factor allocation scheme for LoRa networks based on Q-Learning. Experimental results confirmed that our proposed solution is able to significantly outperform not only non-smart baseline strategies, where the Spreading Factor is either fixed (SF7 or SF12) or randomly selected by the end devices, but also ADR algorithm. More specifically, our allocation scheme improves the PDR as compared to the other solutions. At the same time, it is able to improve the average energy consumption of up to about 43% with respect to the random allocation case, all while keeping the decision process fully-distributed.

REFERENCES

[1] Transforma Insight, "Global IoT market to grow to 24.1 billion devices in 2030, generating $1.5 trillion annual revenue," https://transformainsights.com/news/iot-market-24-billion-usd15-trillion-revenue-2030, May 2020.
[2] J. P. S. Sundaram *et al.*, "A Survey on LoRa Networking: Research Problems, Current Solutions, and Open Issues," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 371–388, 2019.
[3] A. Ikpehai *et al.*, "Low-power Wide Area Network Technologies for Internet-of-Things: A Comparative Review," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2225–2240, 2018.
[4] U. Raza *et al.*, "Low Power Wide Area Networks: An Overview," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 855–873, 2017.
[5] J. Petajajarvi *et al.*, "On the coverage of LPWANs: range evaluation and channel attenuation model for LoRa technology," in *14th International Conference on ITS Telecommunications (ITST)*, 2015, pp. 55–59.
[6] Semtech, "SX1272/3/6/7/8: LoRa Modem Designer's Guide AN1200. 13," 2013.
[7] C. Milarokostas *et al.*, "A comprehensive study on LPWANs with a focus on the potential of LoRa/LoRaWAN systems," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 825–867, 2023.
[8] Semtech, "Understanding the LoRa Adaptive Data Rate," *https://loradevelopers.semtech. com/documentation/tech-papers-and-guides/understanding-adr/*, 2019.
[9] C. Watkins *et al.*, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.
[10] L. Charles *et al.*, "Empirical Analysis of LoRaWAN-based Adaptive Data Rate Algorithms," in *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, 2021, pp. 1–7.
[11] H. Zhong, L. Ning *et al.*, "Optimization of LoRa SF Allocation Based on Deep Reinforcement Learning," *Wireless Communications and Mobile Computing*, vol. 2022, Jan 2022.
[12] I. Ilahi, M. Usama *et al.*, "LoRaDRL: Deep Reinforcement Learning Based Adaptive PHY Layer Transmission Parameters Selection for LoRaWAN," in *IEEE 45th Conference on Local Computer Networks (LCN)*, 2020, pp. 457–460.
[13] D.-T. Ta *et al.*, "LoRa-MAB: A flexible simulator for decentralized learning resource allocation in IoT networks," in *12th IFIP wireless and mobile networking conference (WMNC)*. IEEE, 2019, pp. 55–62.
[14] R. Hamdi *et al.*, "LoRa-RL: Deep reinforcement learning for resource management in hybrid energy LoRa wireless networks," *IEEE Internet of Things Journal*, vol. 9, no. 9, pp. 6458–6476, 2021.
[15] Y. Yu, L. Mroueh *et al.*, "Multi-Agent Q-Learning Algorithm for Dynamic Power and Rate Allocation in LoRa Networks," in *IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, 2020, pp. 1–5.
[16] S. Hong *et al.*, "Reinforcement Learning Approach for SF Allocation in LoRa Network," *IEEE Internet of Things Journal*, vol. 10, no. 20, pp. 18 259–18 272, 2023.
[17] R. S. Sutton *et al.*, *Reinforcement learning: An introduction.* MIT press, 2018.
[18] M. C. Bor *et al.*, "Do LoRa low-power wide-area networks scale?" in *19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2016, pp. 59–67.