Toro Betancur, Veronica; Premsankar, Gopika; Liu, Chen-Feng; Slabicki, Mariusz; Bennis, Mehdi; Di Francesco, Mario

Learning How to Configure LoRa Networks with No Regret: a Distributed Approach

Published in:
IEEE Transactions on Industrial Informatics

DOI:
10.1109/TII.2022.3187721

Published: 01/04/2023

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Toro Betancur, V., Premsankar, G., Liu, C.-F., Slabicki, M., Bennis, M., & Di Francesco, M. (2023). Learning How to Configure LoRa Networks with No Regret: a Distributed Approach. *IEEE Transactions on Industrial Informatics*, *19*(4), 5633-5644. https://doi.org/10.1109/TII.2022.3187721

Learning How to Configure LoRa Networks with No Regret: a Distributed Approach

Verónica Toro-Betancur, Gopika Premsankar, Chen-Feng Liu,
Mariusz Słabicki, Mehdi Bennis, Mario Di Francesco

***Abstract*—LoRa is one of the most popular technologies for low-power wide area networks. It offers long-range communication with a low energy consumption, which makes it ideal for many applications in the Internet of things. The performance of LoRa networks depends on the communication parameters used by individual nodes. Several works have proposed different solutions, typically running on a central network server, to select these parameters. However, existing approaches have not addressed the need to (re-)assign parameters when channel conditions suddenly vary due to additional traffic, changes in the weather or the presence of obstacles. Moreover, allocation strategies that require a central entity to decide communication parameters do not scale due to the large number of configuration packets that must be sent to the nodes. To address these issues, this work proposes NoReL, a distributed game-theoretic approach that allows nodes to autonomously update their parameters and maximize their packet delivery ratio. NoReL is based on a stochastic variant of no-regret learning, which is proven to reach an ϵ-coarse correlated equilibrium in LoRa networks. Extensive simulations show that NoReL achieves a higher delivery ratio than the state of the art in both static and dynamic environments, with an improvement up to 12%.**

***Index Terms*—LoRa, no-regret learning, game theory, Internet of Things, low power wide area networks**

I. Introduction

Low-Power Wide Area Networks (LPWANs) offer long-range wireless communication with low energy consumption at low data rates [1]. This makes them ideal in the Internet of Things (IoT), especially for scenarios where sensors *infrequently* send small packets. Long Range (LoRa) is one of the most popular LPWAN technologies. It achieves communication ranges of 3-5 km in urban areas and 10-15 km in rural environments [2]. LoRa offers a scalable network architecture [3] where nodes simply send packets to all gateways in range, which in turn forward them to a central network server that filters out duplicates. For these reasons, LoRa has been used in several application scenarios, smart metering in particular. These applications demand low traffic from each node, however, LoRa networks usually include thousands of devices which makes reliable communication very challenging [4]. Moreover, LoRa operates in the sub-GHz Industrial, Scientific and Medical (ISM) bands; consequently, the bandwidth is limited and sharing it becomes more difficult as the number of devices increases.

The performance of LoRa networks can be improved by optimizing the communication parameters of individual nodes – in particular, the *spreading factor* (SF) and *transmission power* (TP). The SF trades-off communication range and data rate [4]; a high SF results in a long communication range at a low data rate. The TP not only determines the coverage range of the node, but also affects the probability of collision between two nodes [5], as a higher TP achieves a longer communication range. Furthermore, LoRa networks are subject to the capture effect [6]: depending on its receive power, a packet can be successfully decoded by the gateway even in presence of collisions. Therefore, a higher TP implies a higher probability of success for packets that overlap in time at the receiver, at the cost of higher energy consumption.

Different works in the literature have addressed the optimal assignment of SFs and TPs to LoRa nodes with the goal of improving the overall network performance [7, 8]. These approaches run at the network server and require the final parameters to be sent by the gateway to individual nodes as downlink packets. As such, they target scenarios where the nodes are configured once with the appropriate SF and TP, i.e., it is assumed that the network conditions will remain constant, therefore, the parameters do not need to be altered. However, recent empirical studies have shown that channel conditions in LoRa vary over time due to the weather, the presence of obstacles, and in-band traffic [9, 10]. Such changes can severely impact the reliability of LoRa networks. To maintain an acceptable network performance under varying channel conditions, centralized optimization problems [7, 8] need to be re-run periodically, resulting in an increased downlink traffic to communicate parameters, which further reduces network performance [11]. A few adaptive and distributed approaches have also been proposed [12–14], wherein nodes can choose their SFs and TPs with some feedback from the gateway or the network server. However, some of these solutions require *all* uplink packets to be acknowledged, which increases the overall traffic in the network, while others target scenarios where network conditions do not frequently vary [14].

To address these issues, this work presents NoReL, a fully distributed approach based on no-regret learning [15] to assign SFs and TPs in LoRa networks. With NoReL, nodes

This work was partially supported by: the Academy of Finland under grants number 319710, 319758, 326346, and 338854; the Polish National Center for Research and Development under grant POIR.04.01.04-00-1414/20.

V. Toro-Betancur and M. Di Francesco are with the Department of Computer Science, Aalto University in Espoo, Finland. E-mail: {veronica.torobetancur, mario.di.francesco}@aalto.fi

G. Premsankar is with the Department of Computer Science, University of Helsinki in Helsinki, Finland. E-mail: gopika.premsankar@helsinki.fi

C.-F. Liu is with the Digital Science Research Center, Technology Innovation Institute, 9639 Masdar City, Abu Dhabi, United Arab Emirates. E-mail: chen-feng.liu@tii.ae

M. Słabicki is with the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences in Gliwice, Poland. E-mail: mslabicki@iitis.pl

M. Bennis is with the Centre of Wireless Communications, University of Oulu, 90014 Oulu, Finland. E-mail: mehdi.bennis@oulu.fi

 DOI: 10.1109/TII.2022.3187721

autonomously select appropriate SFs and TPs with minimal feedback from the gateway, thereby allowing nodes to adapt to dynamic changes in the environment. To this end, the problem is modeled as a stochastic game with incomplete information where nodes act selfishly and aim at maximizing their own packet delivery ratio [16, 17]. Here, nodes choose the action that minimizes the regret of not having played it previously. By following such a procedure, nodes reach an ϵ-Coarse Correlated Equilibrium (ϵ-CCE) [18] – a generalization of the well-known Nash equilibrium – in which no node can improve its payoff – namely, its delivery ratio – by unilaterally changing its own communication parameters.

The main contributions of this work are the following. First, it formulates a distributed game-theoretic approach that incurs limited communication overhead. Second, it proves that no-regret learning leads to an ϵ-CCE in LoRa networks. Finally, extensive simulations show that NoReL achieves a higher delivery ratio than the state of the art in both static and dynamic environments, with an improvement up to 12%.

The rest of the article is organized as follows. Section II reviews the related work. Section III introduces the system model, then presents NoReL and proves its convergence. Section IV evaluates NoReL and compares its performance against the state of the art. Last, Section V provides concluding remarks as well as directions for future work.

II. Related work

This section classifies and discusses the state of the art on configuration of LoRa networks.

Centralized assignment. Many works have proposed centralized solutions to configure communication parameters in LoRa networks. One option is assigning a node the minimum SF that allows it to reach at least one gateway; despite being very simple, such a technique can already achieve a high delivery ratio [19]. More sophisticated approaches have also been proposed to further improve performance through optimization techniques leveraging linear programming [7] and sequential waterfilling [8]. Although these approaches obtain stable solutions for SF and TP allocation, most do not define how to inform nodes about the parameters to be used. Moreover, these solutions provide a static allocation of parameters, thus, they are unsuitable for dynamic environments. In fact, they would have to run periodically and send a large number of downlink packets with new configuration parameters. Instead, NoReL is distributed; nodes can update their parameters at any time with only limited feedback from the gateway.

Distributed assignment. Distributed approaches have primarily leveraged game theory [14, 20] for nodes to autonomously choose parameters, with appropriate feedback from the gateways or the network server. For instance, Tolio et al. [14] present a two-player game to allocate SFs based on complete information. However, that work requires each node to have complete information about the SFs used by other nodes and assumes that all nodes transmit with the highest TP. Instead, NoReL is modeled as an incomplete information game, wherein nodes do not require information about the actions taken by other nodes. The allocation problem has also been modeled as a leader and follower game [20]. Specifically, the authors propose to allocate SFs to the nodes for a specific time duration. Their approach requires frequent downlink transmissions, as the parameters need to be updated every time the allocation time expires, which can occur every 20 s. Moreover, the solutions above assume perfect orthogonality between SFs and have not been evaluated in dynamic environments.

Adaptive assignment. Adaptive parameter assignment has also been considered in the literature. The LoRaWAN specifications include a built-in Adaptive Data Rate (ADR) mechanism to assign SFs and TPs on a link basis and improve scalability [21]. Słabicki et al. [22] have studied the performance of ADR; they have also proposed alternative approaches to configure nodes and increase the delivery ratio. Moreover, DyLoRa [12] has been designed as an adaptive mechanism that assigns communication parameters to maximize the energy efficiency of the network. All these solutions allow both the nodes and the network server to run independent algorithms that converge to a stable configuration; in both cases, nodes can only increase their SF and TP when uplink connectivity is lost. Instead, NoReL offers nodes more flexibility in choosing their SFs and TPs, which allows them to quickly adapt to highly varying channel conditions.

Machine learning-based. Machine learning techniques, such as deep reinforcement learning [23] and multi-armed bandits [13, 24], have also been employed in the literature to allocate SF and TP in LoRa networks. These approaches require downlink transmissions after each packet to either acknowledge a received packet or notify nodes about updated communication parameters. As a consequence, they lower the overall network performance since the probability of collisions increases with higher downlink traffic [11]. In contrast, NoReL requires only sporadic feedback from the gateways.

No-regret learning. No-regret learning has been widely studied [15, 25] and applied to wireless networks [26]. However, to the best of the authors' knowledge, such a technique has not been applied to LoRa networks. For instance, no-regret learning has been shown to converge to an ϵ-CCE in cellular networks [18]. However, LoRa is inherently different from the cellular networks studied so far, as it uses an unslotted ALOHA channel access model – transmissions are not aligned, which results in asynchronous operations.

III. Distributed Allocation of SFs and TPs

This section describes the process of allocating transmission parameters in LoRa with NoReL. It first introduces the system model and formulates the problem of optimizing the delivery ratio in distributed settings. It then presents a non-cooperative stochastic game and the proposed no-regret learning strategy.

A. System model

The considered LoRa network comprises multiple gateways and nodes denoted by $\mathcal{M}$ and $\mathcal{N}$, respectively. In particular, the reference architecture is the one in the LoRaWAN specifications [27]: nodes communicate to all gateways in range over the LoRa physical layer; the gateways forward received

packets to a central network server over the backhaul network (e.g., through wired or cellular links), where received packets are filtered and finally sent to the relevant application server. Nodes are stationary and in the range of one or more gateways, without any restrictions on their actual location. Each node ($n \in \mathcal{N}$) uses a spreading factor (SF) s_n and transmission power (TP) p_n from the sets $\mathcal{S}_n$ and $\mathcal{P}_n$, respectively. $\mathcal{S}_n$ and $\mathcal{P}_n$ depend on the geographical region where the LoRa nodes operate [4], and on the node's distance to the closest gateway. Nodes are Class A devices and employ the ALOHA-based medium access control protocol in LoRaWAN [27]. After sending a packet, a node also remains active for two short time windows to receive downlink packets. Such a communication protocol has a low implementation complexity and supports energy-efficient operations.

The goal of this article is to maximize the delivery ratio of each node by choosing appropriate SFs and TPs. The delivery ratio of each node is described by using the model in [5], which is accurate enough to include the most important factors affecting real-world networks – including the capture effect, channel variations, and duty-cycle restrictions. The capture effect states that one out of multiple packets simultaneously received at a destination can be successfully decoded – specifically, the packet whose receive power is higher than the others by the SF-dependent thresholds given in [6, Table II]. These thresholds show that different SFs are not fully orthogonal, namely, that packets transmitted with different SFs have a small probability of interfering with each other; for this reason, SFs are called *quasi-orthogonal*. The packet generation of nodes follows a Poisson distribution as it represents the discrete and random nature of transmissions in LPWANs [3, 5]. Radio propagation follows the log-distance path loss model. Accordingly, the delivery ratio of node n is given by [5]:

$$D_n = (1 - \mathbb{P}_c)\left(1 - \prod_{m \in \mathcal{M}} \mathbb{P}_o^m\right), \tag{1}$$

where $\mathbb{P}_c$ is the probability of interference between the packets sent by node n and the other nodes in the network, and $\mathbb{P}_o^m$ is the outage probability of node n, i.e., the probability that node n is unable to reach the gateway m. Following [5], $\mathbb{P}_c$ is defined as:

$$\mathbb{P}_c = \prod_{i=1}^{|\mathcal{C}|}\left(1 - \left(\prod_{R_n^k \in \mathcal{C}_i}\left(1 - \prod_{j \in R_n^k}\left(1 - \mathbb{P}(E_j')\mathbb{P}(C_j^m)\right)\right)\right)\right). \tag{2}$$

Here, $\mathbb{P}(E_j')$ is the probability of an interfering node $j \in \mathcal{N}$ transmitting simultaneously with node n, and $\mathbb{P}(C_j^m)$ is the probability that packets from node j collide with those from node n. To estimate the probability of node n's transmission not being received by any gateway in range, the set of nodes ($\mathcal{N}$) is partitioned into sets R_n^k, called regions, with respect to each node n and set of gateways k. Each region R_n^k contains the interferers of n that can reach the gateways in k. $\mathcal{C}$ is the set of set covers of all regions, that is, if $\mathcal{C}_i \in \mathcal{C}$ and $\mathcal{C}_i = \{R_n^{k_1}, R_n^{k_2}, \ldots, R_n^{k_j}\}$, then $k_1 \cup k_2 \cup \cdots \cup k_j = \mathcal{M}_n$, where $\mathcal{M}_n$ is the set of gateways that node n can reach. Finally, the outage probability $\mathbb{P}_o^m$ is given by:

$$\mathbb{P}_o^m = \frac{1}{2}\left[\text{erfc}\left(\frac{P_{\text{r},s} - p_n + \overline{PL}(d_0) + 10\gamma \log\left(\frac{d_{n,m}}{d_0}\right)}{\sigma_n\sqrt{2}}\right)\right]. \tag{3}$$

This probability expresses the chance that the receive power of packets from node n at gateway m is below the sensitivity $P_{\text{r},s}$ of the receiver for SF s. Here, p_n, $\overline{PL}(d_0)$, γ, σ_n, and $d_{n,m}$ are, respectively, the TP of node n, path loss at distance d_0, path loss exponent, standard deviation of the normal distribution for channel variations, and the distance between node n and gateway m (see [5] for additional details).

B. Problem description

A distributed approach is introduced next for choosing SFs and TPs depending on external conditions. Specifically, every node aims to individually maximize its own delivery ratio by choosing an appropriate SF and TP pair based on observed channel and traffic conditions. In this regard, a high standard deviation of the node's channel (σ_n) implies a higher outage probability, according to Eq. (3). However, a higher SF lowers the gateway sensitivity, resulting in a decreased probability of outage ($\mathbb{P}_o^m$). A node also has an incentive to use a high TP such that the probability of interference with other nodes, i.e, $\mathbb{P}(C_j^m)$, is minimized [5, Eq. (16)]. Moreover, a higher average sending rate (λ_n) entails a higher $\mathbb{P}(E_j')$ in Eq. (2) which, in turn, increases the probability of interference $\mathbb{P}_c$. Therefore, the delivery ratio D_n is maximized at all channel and traffic conditions by using the highest SF and TP available to the node – for instance, SF12 and TP14 in the European region [4]. By this reasoning, all nodes will choose the same configuration. However, the increased time-on-air due to the higher SF [4, 8] increases the probability of interference with other transmissions [5]. In other words, a node's delivery ratio is decreased if other nodes in the network simultaneously use high SFs and TPs [8]. Since the nodes operate individually, each node is unaware of the SFs and TPs chosen by other nodes. Taking into account this lack of knowledge, the problem to maximize the delivery ratio is formulated as follows.

Problem 1 (Distributed optimization of the delivery ratio)**.** *The distributed maximization of delivery ratio is given by:*

$$\max_{\Pr(s_n,p_n|\sigma_n,\lambda_n)} \sum_{s_n,p_n}\sum_{\sigma_n,\lambda_n} D_n \Pr(s_n,p_n|\sigma_n,\lambda_n)\Pr(\sigma_n,\lambda_n),$$

$$\text{subject to} \sum_{s_n \in \mathcal{S}_n}\sum_{p_n \in \mathcal{P}_n} \Pr(s_n,p_n|\sigma_n,\lambda_n) = 1,$$

$$0 \le \Pr(s_n,p_n|\sigma_n,\lambda_n) \le 1, \forall s_n \in \mathcal{S}_n, p_n \in \mathcal{P}_n,$$

for each node $n \in \mathcal{N}$. Here, $\Pr(s_n,p_n|\sigma_n,\lambda_n)$ is the probability that node n chooses SF s_n and TP p_n given a standard deviation σ_n of the channel and an average sending rate λ_n.

Problem 1 incurs a configuration race among the nodes in the network, as previously mentioned. To tackle the competition between nodes and the lack of information about other nodes, a non-cooperative game [28] is leveraged to assign SFs and TPs in a distributed manner, as described next.

C. Non-cooperative stochastic game

The competition among nodes to selfishly maximize their individual delivery ratio under diverse external conditions can be modeled as a non-cooperative stochastic game played among the set $\mathcal{N}$ of LoRa nodes, with $|\mathcal{N}| = N$. The game is defined as $\mathcal{G} \equiv \big(\mathcal{N}, \mathcal{A}, \mathbf{\Omega}, \{u_n\}_{n\in\mathcal{N}}\big)$, where $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N$ is the network-wide action space and $\mathcal{A}_n = \mathcal{S}_n \times \mathcal{P}_n = \{\alpha_n^1, \alpha_n^2, \ldots, \alpha_n^{K_n}\}$ denotes the action space of player n with $K_n = |\mathcal{S}_n||\mathcal{P}_n|$ total number of actions. Action $\alpha_n = (s_n, p_n) \in \mathcal{A}_n$ consists of a tuple of SF and TP of player n with values s_n and p_n, respectively. That is, a node's action is its choice of SF and TP. Moreover, $\mathbf{\Omega} = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_N$ is the space of random events that characterizes the wireless channel and the network traffic. In particular, Ω_n is the set of all tuples $\omega_n = (\sigma_n, \lambda_n)$ that identify the random state of node n, where σ_n is the standard deviation of the channel between n and the closest gateway; and, λ_n is the average sending rate of n. Nodes aim to maximize their individual delivery ratio, thus, each player's utility u_n is defined as in Eq. (1), i.e., $u_n = D_n$. Note that u_n depends on *both* its own action and random state, as well as the actions and random states of other nodes in the network. Thus, node n's utility $u_n : \mathcal{A} \times \mathbf{\Omega} \to \mathbb{R}_{\geq 0}$ can be expressed as $u_n(\alpha_n, \boldsymbol{\alpha}_{-n}, \omega_n, \boldsymbol{\omega}_{-n})$, in which $\boldsymbol{\alpha}_{-n}$ and $\boldsymbol{\omega}_{-n}$ are the vectors of all other nodes' actions and random states[1].

The formulated non-cooperative stochastic game targets a mixed strategy that describes a conditional probability distribution over all possible actions, as follows:

$$\pi(\boldsymbol{\alpha}^k, \boldsymbol{\omega}^j) = \Pr(\mathcal{A} = \boldsymbol{\alpha}^k | \mathbf{\Omega} = \boldsymbol{\omega}^j), \quad \forall\, \boldsymbol{\alpha}^k \in \mathcal{A},\ \boldsymbol{\omega}^j \in \mathbf{\Omega},$$

where $\boldsymbol{\alpha}^k$ and $\boldsymbol{\omega}^j$ represent a specific realization of the network-wide action and random state. One desirable solution in the game is a game-theoretic equilibrium where all players are satisfied with the strategy. To quantify this, the Coarse Correlated Equilibrium (CCE) and its more general form [18] (ϵ-CCE) are considered next.

Definition 1 (ϵ-Coarse Correlated Equilibrium)**.** *Given a non-negative $\epsilon \geq 0$, the mixed strategy π is an ϵ-CCE for some $\boldsymbol{\omega}^j$ if it satisfies:*

$$\sum_{\boldsymbol{\alpha}_{-n}\in\mathcal{A}_{-n}} u_n(\alpha_n^k, \boldsymbol{\alpha}_{-n}, \omega_n^j, \boldsymbol{\omega}_{-n}^j)\pi(\alpha_n^k, \boldsymbol{\alpha}_{-n}, \omega_n^j, \boldsymbol{\omega}_{-n}^j)$$
$$-\sum_{\boldsymbol{\alpha}\in\mathcal{A}} u_n(\boldsymbol{\alpha}, \omega_n^j, \boldsymbol{\omega}_{-n}^j)\pi(\boldsymbol{\alpha}, \omega_n^j, \boldsymbol{\omega}_{-n}^j) \leq \epsilon,\ \forall n \in \mathcal{N}, \alpha_n^k \in \mathcal{A}_n$$

In other words, the utility enhancement of a player (over the utility achieved by following the ϵ-CCE strategy) who unilaterally deviates from the ϵ-CCE strategy, while other players follow the ϵ-CCE strategy, is bounded by ϵ. Note that a CCE is achieved if $\epsilon = 0$. Moreover, the equilibrium is achieved for each $\boldsymbol{\omega}^j$ independently, i.e., there is an ϵ-CCE for each $\boldsymbol{\omega}^j$. A straightforward way to implement the ϵ-CCE strategy is with a central entity that instructs all nodes to jointly play the action $\boldsymbol{\alpha}$ according to the ϵ-CCE distribution $\pi(\boldsymbol{\alpha}, \boldsymbol{\omega})$. However, the centralized approach is not feasible in the considered scenario, as it requires to send downlink packets to a large number of nodes in the network whenever a configuration change is required, e.g., every time the channel conditions get worse. This would significantly reduce the capacity of the LoRa network [11]. Thus, a regret-based method [18] is introduced for each node to learn an SF and TP allocation strategy that achieves an ϵ-CCE for each ω_n^j in a distributed fashion.

[1]The $-n$ subscript denotes the set of all players excluding the n-th one.

D. Distributed no-regret learning

No-regret learning [18] is adopted to achieve an ϵ-CCE in a distributed manner. Specifically, players minimize the regret of their actions: after carrying out a certain action that has led to some utility, they evaluate their regret for not having chosen a different action. Then, the probability of playing the actions with the highest regret is increased accordingly. By doing so, nodes learn and maximize their utilities. This learning occurs in communication rounds as follows. Each node sends c packets by autonomously choosing a particular action. Afterwards, the gateway calculates the *observed utility* of the node as the ratio between the successfully received and sent packets during that round. Additionally, it determines the random state of the node (see Section IV-A) and then sends the related information (i.e., the node's utility and random state) back to the node in a downlink packet. Note that the communication rounds are *asynchronous* – those of individual nodes are independent and have different duration, based on how often they send packets – therefore, the nodes change their actions at different times. Moreover, individual realizations of the random state $\mathbf{\Omega}$ in time – namely, each $\boldsymbol{\omega}(t) \in \mathbf{\Omega}(t)$ – are assumed to be i.i.d.

Given that all nodes in the network follow the strategies they used over the previous T rounds, the regret of a specific action $\alpha_n^k \in \mathcal{A}_n$, given a random state ω_n^j for node n, is defined as

$$r_n(\alpha_n^k, \omega_n^j) = \frac{1}{T}\sum_{t=1}^{T}\Big[u_n(\alpha_n^k, \boldsymbol{\alpha}_{-n}(t), \omega_n^j, \boldsymbol{\omega}_{-n}^j(t)) - \tilde{u}_n(t)\Big] \quad (5)$$

in which $\tilde{u}_n(t)$ is the observed utility of node n at round t. The regret is the normalized difference between the utility that would have been obtained by playing action α_n^k and the actual utility given by the current action. A positive regret $r_n(\alpha_n^k, \omega_n^j) > 0$ signifies that node n can achieve a better utility by consistently playing action α_n^k, while the other nodes follow their original strategies. Then, node n with regret vector $\boldsymbol{r}_n = [r_n(\alpha_n^k, \omega_n^j) : \alpha_n^k \in \mathcal{A}_n, \omega_n^j \in \Omega_n]$ selects its action based on the Boltzmann distribution that results from solving:

$$\beta_n(\boldsymbol{r}_n, \alpha_n, \omega_n^j) = \underset{\pi_n(\alpha_n, \omega_n^j)}{\arg\max} \left\{ \sum_{\alpha_n^k \in \mathcal{A}_n} \Big[\pi_n(\alpha_n^k, \omega_n^j) r_n(\alpha_n^k, \omega_n^j) - \frac{1}{\kappa_n(\omega_n^j)} \pi_n(\alpha_n^k, \omega_n^j) \ln\big(\pi_n(\alpha_n^k, \omega_n^j)\big)\Big] \right\} \quad \forall \omega_n^j \in \Omega_n, \ (6)$$

where $\pi_n(\alpha_n, \omega_n^j) = \Pr(\alpha_n | \omega_n^j)$. Such a distribution allows to explore different actions [18]. In particular, the temperature parameter $\kappa_n(\omega_n^j) > 0$ controls the trade-off between the optimality of the utility (i.e., exploitation) and maximizing the entropy of the strategy (i.e., exploration). Specifically, when

$\kappa_n(\omega_n^j)$ is small, the entropy of the system is high and the node explores actions more broadly. Conversely, the entropy lowers when $\kappa_n(\omega_n^j)$ increases and the node starts choosing the actions with the highest probabilities. In contrast with [18, 26], this work considers random states, therefore, the temperature parameter $\kappa_n(\omega_n^j)$ is chosen to be dependent on ω_n^j to keep the exploration and exploitation phases independent for each random state. The solution of Eq. (6) is given by [18]:

$$\beta_n(\boldsymbol{r}_n, \alpha_n^k, \omega_n^j) = \frac{\exp\left(\kappa_n(\omega_n^j) r_n^+(\alpha_n^k, \omega_n^j)\right)}{\sum\limits_{\alpha_n^i \in \mathcal{A}_n} \exp\left(\kappa_n(\omega_n^j) r_n^+(\alpha_n^i, \omega_n^j)\right)},$$
$$r_n^+(\alpha_n^k, \omega_n^j) = \max\{0, r_n(\alpha_n^k, \omega_n^j)\},$$

$\forall\, \alpha_n^k \in \mathcal{A}_n, \omega_n^j \in \Omega_n$. However, calculating the utility function $u_n(\alpha_n^k, \boldsymbol{\alpha}_{-n}(t), \omega_n^j, \boldsymbol{\omega}_{-n}^j(t))$ in Eq. (5) requires node n to know all the previous actions and random states of all the other nodes in the network. That is, a perfect information game is assumed. This is clearly impractical, as the information about the other nodes would need to be transmitted by frequent downlink packets. To tackle this issue, NoReL is introduced next as an imperfect information game where nodes iteratively estimate their utility and regret of each action $\alpha_n^k \in \mathcal{A}_n$ to update their strategy at the beginning of each round t, given their current random state ω_n^j. The detailed steps are the following:

$$\hat{u}_n(\alpha_n^k, \omega_n^j, t) = \hat{u}_n(\alpha_n^k, \omega_n^j, t-1)$$
$$+\nu_n(\omega_n^j, t) \mathbb{1}_{\{\alpha_n = \alpha_n^k\}} \mathbb{1}_{\{\omega_n = \omega_n^j\}} \big(\tilde{u}_n(t) - \hat{u}_n(\alpha_n^k, \omega_n^j, t-1)\big), \quad (7)$$
$$\hat{r}_n(\alpha_n^k, \omega_n^j, t) = \hat{r}_n(\alpha_n^k, \omega_n^j, t-1)$$
$$+\gamma_n(\omega_n^j, t) \left(\hat{u}_n(\alpha_n^k, \omega_n^j, t) - \tilde{u}_n(t) - \hat{r}_n(\alpha_n^k, \omega_n^j, t-1)\right), \quad (8)$$
$$\pi_n(\alpha_n^k, \omega_n^j, t) = \pi_n(\alpha_n^k, \omega_n^j, t-1)$$
$$+\mu_n(\omega_n^j, t) \left(\beta(\hat{\boldsymbol{r}}_n(t), \alpha_n^k, \omega_n^j) - \pi_n(\alpha_n^k, \omega_n^j, t-1)\right), \quad (9)$$

$\forall\, \alpha_n^k \in \mathcal{A}_n, \omega_n^j \in \Omega_n, t \in \mathbb{Z}^+$. Here, $\mathbb{1}_{\{x=y\}}$ is the indicator function that is equal to 1 if $x = y$ or to 0 otherwise. The rationale behind these equations is the following. Let us consider the moving-average estimator $\hat{a}(t) = \frac{1}{t}\sum_{\tau=1}^{t} a(\tau)$ of the expectation $\mathbb{E}[a]$, which can be recursively written as $\hat{a}(t) = \frac{1}{t}a(t) + \frac{t-1}{t}\hat{a}(t-1) = \hat{a}(t-1) + \frac{1}{t}[a(t) - \hat{a}(t-1)]$. Here, the difference between the latest realization and the previous estimation – namely, $a(t) - \hat{a}(t-1)$ – is used to calibrate the estimation, and its impact decays in terms of $1/t$. In contrast, the weights of the calibration terms in Eqs. (7)–(9) are in their general form $\nu_n(\omega_n^j, t)$, $\gamma_n(\omega_n^j, t)$, and $\mu_n(\omega_n^j, t)$, which are round-dependent learning rates and satisfy:

$$\lim_{T\to\infty} \sum_{t=1}^{T} \left[\mu_n(\omega_n^j, t) + \gamma_n(\omega_n^j, t)\right] = \infty, \quad (10a)$$

$$\lim_{T\to\infty} \sum_{t=1}^{T} \left[\mu_n(\omega_n^j, t)^2 + \gamma_n(\omega_n^j, t)^2\right] < \infty, \quad (10b)$$

$$\lim_{t\to\infty} \frac{\gamma_n(\omega_n^j, t)}{\nu_n(\omega_n^j, t)} = 0, \quad (10c)$$

$$\lim_{t\to\infty} \frac{\mu_n(\omega_n^j, t)}{\gamma_n(\omega_n^j, t)} = 0. \quad (10d)$$

Note that the p-series $\sum_{t=1}^{\infty} \frac{1}{t^p}$ diverges when $p \leq 1$ and converges when $p > 1$. Hence, the learning rates $\mu_n(\omega_n^j, t), \gamma_n(\omega_n^j, t)$ and $\nu_n(\omega_n^j, t)$ can be selected as [26]:

$$\mu_n(\omega_n^j, t) = \mathbb{1}_{\{\omega_n = \omega_n^j\}} \cdot \frac{1}{t^{p_\mu}}, \quad (11)$$

$$\gamma_n(\omega_n^j, t) = \mathbb{1}_{\{\omega_n = \omega_n^j\}} \cdot \frac{1}{t^{p_\gamma}}, \quad (12)$$

$$\nu_n(\omega_n^j, t) = \mathbb{1}_{\{\omega_n = \omega_n^j\}} \cdot \frac{1}{t^{p_\nu}}, \quad (13)$$

with $0.5 < p_\mu, p_\gamma, p_\nu \leq 1$, so that the conditions in Eq. (10a) and Eq. (10b) are satisfied. Furthermore, $p_\mu > p_\gamma > p_\nu$ holds to meet the conditions in Eq. (10c) and (10d). The learning rates μ_n, γ_n and ν_n are updated at every round, which in turn affects the amount of new information adopted in updating the utilities, regrets, and probability distributions. These rates also depend on ω_n^j to update the parameter for each random state independently. Finally, the temperature is updated as:

$$\kappa_n(\omega_n^j, t) = \kappa_n(\omega_n^j, t-1) + \mathbb{1}_{\{\omega_n = \omega_n^j\}} \cdot \psi(t), \quad (14)$$

with a non-decreasing function $\psi(t)$. In this work, $\psi(t) = t^2$, as it tends to infinity. As a consequence, the nodes explore their actions when t is small and then exploit the best ones as t increases. This, in turn, makes ϵ smaller with time, thus asymptotically converging to a CCE [18]. Specifically, nodes are constantly participating in the game and the convergence of the system is defined as follows.

Definition 2 (Convergence time)**.** *The convergence time t_0 for a random state $\boldsymbol{\omega}^j = \left(\omega_1^j, \cdots, \omega_N^j\right)$ and some small $\delta > 0$ is such that, for all $t \geq t_0$ and $n \in \mathcal{N}$,*

$$\frac{\sum_{\alpha_n^k \in \mathcal{A}_n} \left|\pi_n(\alpha_n^k, \omega_n^j, t) - \pi_n(\alpha_n^k, \omega_n^j, t-1)\right|}{K_n} \leq \delta.$$

NoReL is proven to converge to an ϵ-CCE [18] next.

Definition 3 (Lipschitz function)**.** *An f function is said to be Lipschitz if $|f(x) - f(y)| \leq C|x - y|$, for all x and y in the domain of f and for a real constant C independent of x and y.*

Proposition 1. *The learning in Eqs.* (7)–(9) *converge to an ϵ-CCE strategy $\boldsymbol{\pi}^*(\boldsymbol{\omega}^j) = \{\pi_1^*(\omega_1^j), \cdots, \pi_N^*(\omega_N^j)\}$ of $\mathcal{G}$ for all $\boldsymbol{\omega}^j$ if and only if the conditions in Eq.* (10) *are met, where $\lim_{t\to\infty} \pi_n(\omega_n^j, t) = \pi_n^*(\omega_n^j)$.*

Proof. For simplicity, the following considers a fixed $\boldsymbol{\omega}^j$ as the derivation is the same for all $\boldsymbol{\omega}^j \in \boldsymbol{\Omega}$. The learning in NoReL is a discrete-time process that can be expressed as the asymptotic trajectory of a flow [18], therefore, its convergence can be proven on the basis of [25, Proposition 4.1]. Accordingly, it is enough to show that u_n and $\beta_n(\boldsymbol{r}_n, \alpha_n, \omega_n^j)$ are Lipschitz functions, since the conditions in Eq. (10) hold by hypothesis.

From Eq. (1), the observed utility of node n at a certain round t can be expressed as $u_n = 1 - x$, wherein $x = \mathbb{P}_c + \mathbb{P}_o^* - \mathbb{P}_c\mathbb{P}_o^*$, with $\mathbb{P}_o^* = \prod_{m\in\mathcal{M}} \mathbb{P}_o^m$, is a function of the network-wide action and random state, i.e., $x = x(\alpha_n, \boldsymbol{\alpha}_{-n}, \omega_n^j, \boldsymbol{\omega}^j)$. Letting $1 \geq x \geq y \geq 0$ yields:

$$\frac{|u_n(x) - u_n(y)|}{|x - y|} = \frac{|(1-x) - (1-y)|}{|x - y|} = \frac{|y - x|}{|x - y|} = 1.$$

TABLE I: Simulation parameters.

Parameter	Value
Sending rate (λ_n)	0.001 s^{-1} [5, 7]
Duty cycle	0.01 [4]
Path loss	$\overline{L}(d_0)$ = 128.95 dBm, d_0 = 1 km, η = 2.32, σ = {3.54, 7.08} [2]
Packet length	20 bytes [29]
Preamble length	8 bytes [4]
Frequency	868 MHz
Bandwidth	125 kHz
Coding rate	4/8
SFs	{7, 8, 9, 10, 11, 12}
TPs	{2, 5, 8, 11, 14} dBm
Receiver sensitivity per SF	{7: -124, 8: -127, 9: -130, 10: -133, 11: -135, 12: -137} dBm [30]
Supply current per TP	{2: 24, 5: 25, 8: 25, 11: 32, 14: 44} mA [30]
Time on air per SF	{7: 0.0780, 8: 0.1397, 9: 0.2467, 10: 0.4935, 11: 0.8560, 12: 1.7121} s

Hence, u_n is a Lipschitz function. Consider now the regret vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of node n; trivially, it is $0 < |\beta_n(\boldsymbol{x}, \alpha_n, \omega_n^j) - \beta_n(\boldsymbol{y}, \alpha_n, \omega_n^j)| < 1$, therefore, $|\beta_n(\boldsymbol{x}, \alpha_n, \omega_n^j) - \beta_n(\boldsymbol{y}, \alpha_n, \omega_n^j)| < C|\boldsymbol{x} - \boldsymbol{y}|\ \forall \boldsymbol{x}, \boldsymbol{y}$ and for some scalar C. Consequently, both u_n and $\beta_n(\boldsymbol{r}_n, \alpha_n, \omega_n^j)$ are Lipschitz functions, which completes the proof. □

IV. Performance evaluation

This section evaluates NoReL against the state of the art by simulation. It first introduces the related setup and methodology, it then discusses how to estimate the path loss parameters based on observed channel conditions. It finally presents the results obtained for both static and dynamic scenarios, comprising networks with different layouts and characteristics.

A. Setup and methodology

A custom discrete-time Python simulator[2] was developed to determine the SF and TP assignments of nodes over time. The simulator incorporates – for all uplink and downlink packets – the capture effect, quasi-orthogonal SFs, duty-cycle restrictions for both LoRa nodes and gateways, and a collision model that requires the last five preamble symbols to be correctly received to decode the whole packet [4]. Moreover, gateways are considered half-duplex, i.e., no packets are received while they are transmitting and vice versa. As discussed in Section III-D, nodes update their SF and TP at the beginning of each round with NoReL. Accordingly, the utility and random state of each node are computed and Eqs. (7)–(9) are applied to determine its SF and TP at each round.

The performance is primarily evaluated in terms of *convergence time* with $\delta = 0.01$ (according to Definition 2) and the *average delivery ratio* in the network as the probability (expressed as a percentage) that its packets are correctly received, obtained with Eq. (1), i.e., by evaluating the model[3] in [5]. For comparison purposes, we also report the *energy consumption* E calculated as:

$$E = \sum_{n=1}^{N} \frac{V(I_{\text{tx}}(p_n)T_{\text{air}}(s_n) + 2I_{\text{rx}}T_{\text{rx}})}{D_n},$$

where V is the input voltage of the LoRa devices; $I_{\text{tx}}(p_n)$ is the current consumed for a transmission with TP p_n; $T_{\text{air}}(s_n)$ is the time on air of a packet transmitted with SF s_n; I_{rx} is the current consumed during the two receive windows, of duration T_{rx}, that follow an uplink transmission; and D_n is as in Eq. (1). In this work, $V = 3.3\,\text{V}$, $I_{\text{rx}} = 11\,\text{mA}$, $T_{\text{rx}} = 164\,\text{ms}$ [31], while $I_{\text{tx}}(p_n)$ and $T_{\text{air}}(s_n)$ are taken from [30]. Therefore, E is the energy needed by all nodes to transmit a packet that is successfully received.

For all the results presented next, each individual simulation run lasts for 15 days of simulated time, unless otherwise stated; all experiments are repeated 30 times and the average value is reported in the figures together with the corresponding standard deviation. Table I shows the simulation parameters; those for the path loss parameters were derived in [2] through extensive measurements on a real LoRa network deployed in a sub-urban environment. The packet length is set to 20 B, a typical value in LoRa networks as shown in [29]. The rest of the parameters were chosen according to those widely employed in the state of the art [5, 7].

As explained in Section III-D, NoReL requires the gateways to calculate, at every round, the random state of the nodes – namely, the observed standard deviation of the wireless channel (σ_n) and the average sending rate (λ_n). To simplify the implementation, the channel conditions and sending rates are computed across the whole network rather than for individual nodes. Therefore, the gateway only sends the network-wide parameters (σ, λ) to all the nodes in range, instead of sending individual values (σ_n, λ_n) to each node. The ordinary least squares method [32] is adopted to estimate the value of σ over a window of 80 packets as detailed in Section IV-B; λ is calculated with the packets received over one-hour intervals as $\lambda = \overline{\lambda}N$. Moreover, the random state space is discretized as follows: σ can belong to one of the intervals $[0, 5), [5, 10), [10, 15), [15, 20]$; $\overline{\lambda}$ to one of the intervals $[0, 0.003), [0.003, 0.006]$. Consequently, eight random states are considered in total.

Another critical step in the implementation of NoReL is the selection of the learning parameters and the duration of a round, as they affect the convergence time and the performance of the network. After extensive simulations (not included here due to lack of space), the following parameters were chosen as a trade-off between convergence time and delivery ratio: $p_\nu = 0.8, p_\gamma = 0.9, p_\mu = 1$ and the round size $c = 10$. That is, downlink packets are sent every 10 packets to each node similar to the protocol used in ADR [21], which is widely implemented in real-world LoRa networks.

B. Estimation of path loss parameters

The gateways estimate the channel conditions and report them to the nodes in downlink packets. To this end, the ordinary least squares (OLS) [32] method is adopted, as it

[2]Available at: https://github.com/VeronicaToro/NoReL/

[3]Available at: https://github.com/VeronicaToro/LoRa-model/

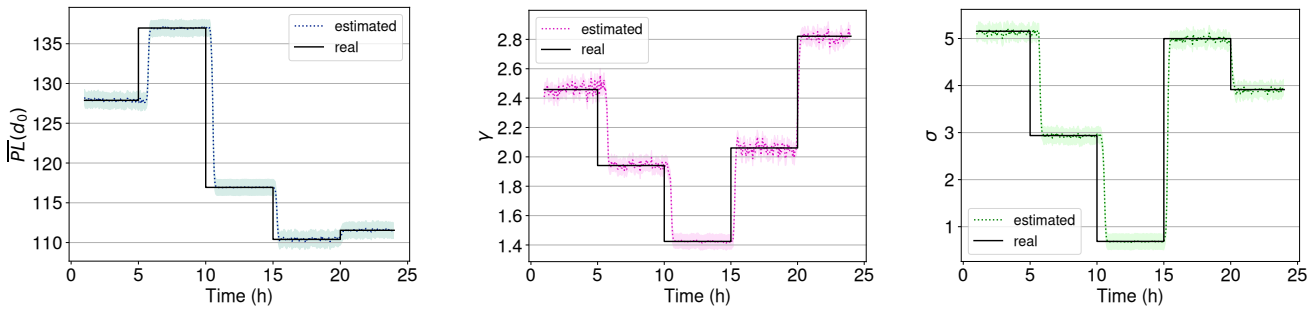


Fig. 1: Path loss parameters (a) $\overline{PL}(d_0)$, (b) γ and (c) σ for random changes occurring every 5 hours in a day.

is easy to implement and it has been also used in real LoRa networks [33]. Here, the gateway estimates $\overline{PL}(d_0), \gamma$ and σ based on the receive and transmission power of packets as well as on its distance from the source node over a window of L packets. For this purpose, the following matrix representation is introduced. First, $\boldsymbol{\chi} = [\mathbf{1} \quad 10\log(\boldsymbol{d}/d_0)]$ is an L by 2 matrix with $\mathbf{1}$ and $\boldsymbol{d}$ the column vectors of ones and distances, respectively. That is, $\boldsymbol{d}$ contains the distances between the gateway and the source node for each of the L packets. Moreover, $\boldsymbol{C} = \left[\overline{PL}(d_0) \quad \gamma\right]^T$ is a 2 by 1 matrix and $\boldsymbol{\Sigma} = [\boldsymbol{X_\sigma}]$ a column vector of size L containing the standard deviation of the channel for each of the L received packets, modeled by a normal distribution, i.e., $X_\sigma \sim \mathcal{N}(0, \sigma)$. Then, the vector of receive powers is given by:

$$\boldsymbol{y} = \boldsymbol{\chi}\boldsymbol{C} + \boldsymbol{\Sigma}.$$

This allows to estimate $\overline{PL}(d_0)$ and γ, respectively, as the first and second entry of the following vector:

$$\hat{\boldsymbol{C}} = \left(\boldsymbol{\chi}^T\boldsymbol{\chi}\right)^{-1}\boldsymbol{\chi}^T\boldsymbol{y}.$$

Finally, the variance σ^2 of X_σ is estimated as:

$$\sigma^2 = \frac{1}{L-1}\left(\boldsymbol{y} - \boldsymbol{\chi}\hat{\boldsymbol{C}}\right)^T\left(\boldsymbol{y} - \boldsymbol{\chi}\hat{\boldsymbol{C}}\right).$$

To keep the estimated parameters smooth, a learning rate ζ is used to satisfy:

$$\begin{aligned}\widehat{PL}(d_0)_t &= \zeta\widehat{PL}(d_0)_{t-1} + (1-\zeta)\widehat{PL}(d_0)_t,\\ \hat{\gamma}_t &= \zeta\hat{\gamma}_{t-1} + (1-\zeta)\hat{\gamma}_t,\\ \hat{\sigma}_t &= \zeta\hat{\sigma}_{t-1} + (1-\zeta)\hat{\sigma}_t,\end{aligned}$$

where $\hat{x}_t$ is the estimation of parameter x at time t. The value $\zeta = 0.3$ was empirically found to achieve the best trade-off between the variance in the estimated data and the response delay after changes in the channel conditions.

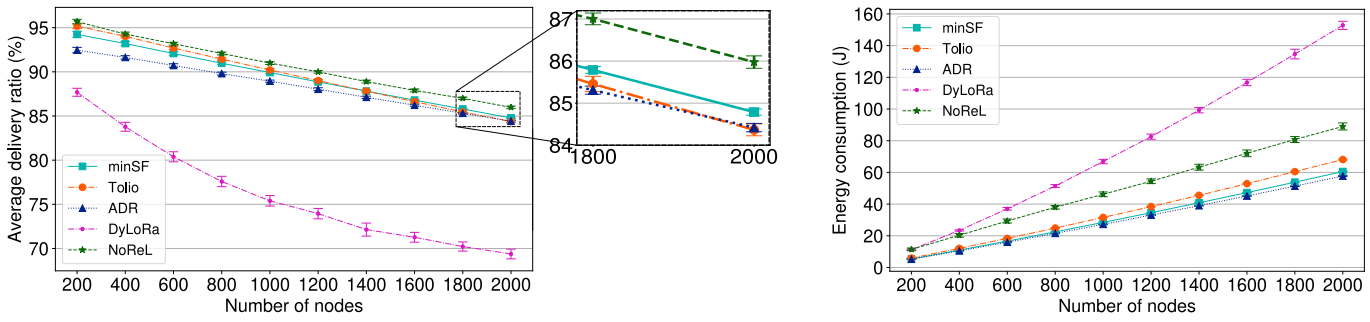
Figure 1 characterizes the accuracy of path loss parameters estimation for a case where $\overline{PL}(d_0), \gamma$ and σ vary randomly every 5 hours during one day; the shaded areas in the plots indicate the 95% confidence intervals. The figure clearly shows how the OLS-based estimation quickly obtains[4] the correct value upon changes, with only a small variation over time – namely, less than 5% in all cases.

[4]The convergence to the correct value could be shortened by decreasing ζ at the cost of a higher variance.

C. Comparison against state of the art

The performance of NoReL is compared against the schemes discussed next. **MinSF** is a simple baseline that assigns each node the lowest SF required to achieve connectivity to the nearest gateway at the highest TP. This scheme requires information on the distance of each node to the nearest gateway and the path loss parameters. Thus, it runs at the central network server [19]. **Tolio** is the two-player game proposed in [14] to assign SFs to LoRa nodes, played between each node and the rest of the network. Each node is aware of the aggregate distribution of SFs in the network and chooses between keeping the current SF or increasing it by one. All nodes use the highest TP. **Adaptive Data Rate** (**ADR**) assigns SFs and TPs to each node such that the overall network capacity is increased and energy consumption is minimized [22]. It comprises two algorithms that run independently at the node and the network server. The algorithm running at the node is defined by the LoRaWAN specifications [27]: it increases the SF and TP of a node whenever it cannot reach a gateway. On the other hand, the algorithm running at the network server depends on the operator of the LoRa network. The implementation here follows the one presented in [22]: the network server adjusts the SF and TP depending on the quality of a link estimated by observing the average signal-to-noise ratio (SNR) of the last 20 packets received from the node. Unless otherwise stated, the device margin is set to 15 dB [21] and the remaining parameters are set to the default values reported in [22, 27]. **DyLoRa** is the mechanism proposed in [12] to assign SFs and TPs to nodes. Similar to ADR, there are two components that run at the node and the network server. The algorithm at the node simply resets the SF and TP to the highest values whenever uplink connectivity is lost (i.e., 12 consecutive packets are dropped). On the other hand, the network server predicts the delivery ratio for each node for all combinations of SFs and TPs based on the SNR of the 6 previously received packets. It then chooses the configuration that maximizes the energy efficiency of the node, i.e., the one that achieves a high predicted delivery ratio with a low energy.

1) Static scenarios: First, the network performance is evaluated in a static scenario wherein nodes are deployed uniformly located within a radius of 2 km from a single gateway. Different densities of the network are evaluated by varying the number of nodes. Figure 2a presents the average delivery



(a) (b)

Fig. 2: (a) Average delivery ratio and (b) energy consumption as a function of the number of nodes in static scenarios.

ratio, showing similar values for MinSF, Tolio and NoReL in networks with fewer than 800 nodes. Still, NoReL outperforms other approaches in all network densities. Most nodes can use the smallest available SF (i.e., SF7) to reach the gateway with a deployment radius of 2 km. Thus, MinSF assigns SF7 to around 75% of the nodes and SF8 to the rest of the nodes. On the other hand, Tolio adds more SF diversity, with approximately 50% of the nodes using SF8. This results in a high delivery ratio for networks with few nodes, e.g., with 200 nodes. However, as the density of nodes increases, using only SF7 and SF8 increases the probability of collisions, even though the time-on-air of the packets is short. In such scenarios, NoReL assigns higher SFs to a few nodes (around 22% of the nodes use SFs higher than 8) resulting in a better SF diversity, thereby reducing the probability of collisions between packets. For this reason, NoReL achieves the highest average delivery ratio in all considered densities. ADR achieves a slightly lower average delivery ratio than MinSF because, while MinSF assigns TP14 to all the nodes and NoReL does the same for approximately 81% of the nodes, ADR assigns TPs lower than TP14 to 30% of the nodes. As discussed before, a high TP minimizes the probability of outage and collisions with other nodes. Finally, DyLoRa achieves the lowest average delivery ratio, as it is designed to achieve a high energy efficiency through lower SFs and TPs [12]. Specifically, DyLoRa assigns TP14 to only 5% of the nodes. Finally, additional simulations showed that in larger networks (with a 6 km deployment radius) the average delivery ratio achieved by all approaches is similar even with a high density of nodes (not included due to space constraints). Indeed, the nodes are more constrained in their communication parameters, under larger deployment areas. For instance, the nodes farthest from the gateway must use SF12 and TP14. Then, other nodes will avoid using SF12 to minimize collisions. Accordingly, the possibilities of changing SFs and TPs are limited, thus ADR, Tolio and NoReL converge to an assignment similar to that of MinSF.

Figure 2b depicts the energy consumption achieved by the different approaches in the considered static scenarios. The figure shows that MinSF, Tolio and ADR result in a similar energy consumption at all network densities. This happens because they assign low SFs, even though they also assign TP14 to most of the nodes. On the other hand, DyLoRa and NoReL consume higher energy as they assign higher SFs. In fact, it is the time on air of the packets determined by the SF that primarily affects the energy consumption. This can be clearly seen from the parameters $I_{\text{tx}}(p_n)$ and $T_{\text{air}}(s_n)$ in Table I: while $I_{\text{tx}}(2)$ and $I_{\text{tx}}(14)$ differ by 45.4%, $T_{\text{air}}(7)$ and $T_{\text{air}}(12)$ differ by 95.4%. Therefore, keeping a low SFs is the best option to save energy but not to achieve a high delivery ratio, as previously discussed. DyLoRa actually incurs in the highest energy consumption despite being designed to optimize energy efficiency because the related characterization [12, Eq. (3)] penalizes high TPs instead of high SFs.

Different network layouts are evaluated next, including scenarios with multiple gateways and a non-uniform distribution of nodes around a gateway. In both cases, the networks comprise 1,000 nodes. Figures 3a and 3b show these layouts where the gateways are represented by black triangles and the nodes by colored points whose colors show the SFs assigned by NoReL; Figures 3c and 3d, instead, show the SFs assignment of MinSF for comparison[5] purposes. Figure 3a clearly shows that NoReL mainly assigns SF7 to the nodes that are the closest to the gateway and SF8 to the nodes in the outer ring, similar to MinSF. However, there is some SF diversity among all nodes. In contrast, Figure 3c highlights how SFs are assigned in rings by MinSF. ADR produces the same assignment as the one obtained by MinSF, whereas Tolio and DyLoRa exhibit a greater SF diversity, with SF8 and SF10 as the most used, respectively. For the non-uniform scenario, the difference between NoReL (Figure 3b) and MinSF (Figure 3d) is less visible, as both approaches primarily assign SFs in rings. This happens for NoReL because nodes are located far away from the gateway, resulting in a few actions to choose from. Nevertheless, NoReL still assigns a high SF to a few nodes. The assignments of Tolio and DyLoRa, instead, mostly rely on SF8 and SF10, respectively, similar to the previous scenario.

Figures 3e and 3f show the delivery ratio of the different schemes for the two considered static scenarios. These results clearly depend on the diversity of the assigned SFs. In fact, NoReL and Tolio achieve the highest delivery ratio for the scenario with multiple gateways (Figure 3e) – in detail, NoReL achieves a delivery ratio 2.3% higher than ADR – as they exhibit the highest TP and SF diversity. For the non-uniform scenario, the difference between the average delivery ratios achieved

[5]The rest of the discussion also refers to the assignments obtained by ADR, Tolio and DyLoRa even though they are not reported in figures for brevity.

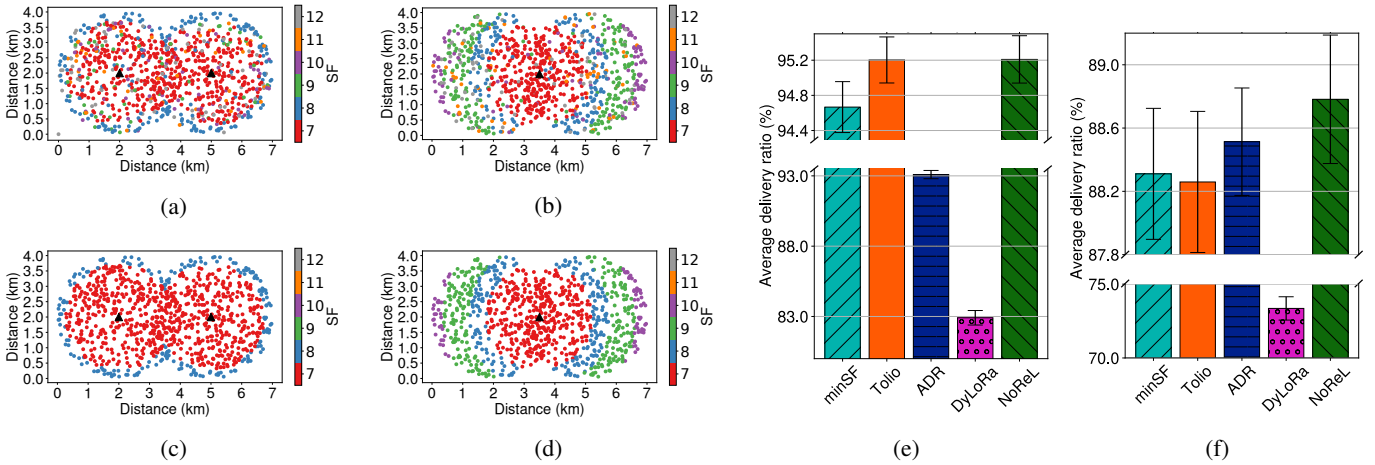


Fig. 3: Spreading factor assignment of: NoReL for the (a) 2-gateway and (b) non-uniform scenarios; MinSF for the (c) 2-gateway and (d) non-uniform scenarios. Gateways are denoted as black triangles and nodes as colored circles in the figures. Average delivery ratio of the considered approaches for the (e) 2-gateway and (f) non-uniform scenarios.

by MinSF, Tolio, ADR and NoReL is below 1% (Figure 3f). The slightly better performance of NoReL is due to the few nodes that are assigned a high SF, as previously discussed. DyLoRa achieves a significantly lower delivery ratio than the other schemes in both scenarios due to the low TPs assigned to the nodes while aiming to maximize energy efficiency.

2) Dynamic scenarios: The following three scenarios are considered to evaluate how the different schemes adapt to changes in network conditions, when: (i) new nodes are added to an existing network with a stable configuration; (ii) the average sending rate changes after a certain time, for instance, an application may temporarily require more data during certain time periods; (iii) the channel conditions vary due to changes in weather conditions [9], in-band traffic from other networks or even the presence of obstacles (e.g., people) near end devices [10]. All scenarios are evaluated in a network with one gateway and 500 nodes uniformly deployed within a radius of 4 km. MinSF and Tolio are no longer considered, as they are designed for static scenarios and do not define means to identify when nodes are reconfigured or notified about the new configuration. In addition to the delivery ratio, the results reported next provide the SF and TP assignment obtained by the considered schemes in the different scenarios. In all cases, values are shown as a function of the simulation time, expressed in terms of the number of packets sent in the network, to abstract from the actual sending rate. The time at which a change takes place is marked with a dotted vertical line in all figures. In addition, Figure 4 shows the standard deviations of the obtained values as a shaded area and the convergence time for NoReL, marked as t_0.

Additional nodes. In the first scenario (Figure 4a), the network is initially configured with a stable configuration achieved by the specific approach being evaluated (ADR, DyLoRa or NoReL). After two days of simulated time (around 86,700 sent packets), 100 new nodes are added at random locations within the deployment area. The newly-added nodes initially use the minimum SFs that allow them to reach the gateway and TP14. Figure 4a shows that NoReL achieves the highest delivery ratio. In fact, NoReL achieves a delivery ratio 1.4% higher than ADR and 12% higher than DyLoRa, on average. Moreover, the convergence time of NoReL is low, as it takes around 14,000 packets (approximately 7 hours) to converge after the new nodes are added. This is due to the learning rates that the other nodes had beforehand. That is, all the pre-existing nodes had already reached the steady state and only the new ones had to learn the best strategy. It is worth observing that DyLoRa sets the new communication parameters more quickly for the recently-added nodes (they all reach their final parameters in around 4 hours), even though its delivery ratio (around 76.4%) is lower than that of both ADR and NoReL.

Figure 5 shows the SF and TP assignment over time of the different methods for the considered scenario. It is clear how NoReL tries different strategies before converging to the final assignment, which is almost a uniform distribution of SFs and mainly high TPs (Figure 5a). The SF and TP assignment of ADR (Figure 5b) and DyLoRa (Figure 5c), instead, remains basically the same after the new nodes are added.

Additional traffic. In the second scenario, the initial network configuration is obtained with MinSF. Then, the average sending rate per node is increased from $\overline{\lambda} = 0.001$ to $\overline{\lambda} = 0.005$ after five days. The sending rate then goes back to its initial value ($\overline{\lambda} = 0.001$) after five more days. Figure 4b clearly shows that the delivery ratio is affected by the increase of network traffic for all considered schemes (i.e., NoReL, ADR and DyLoRa). Specifically, the average delivery ratio of NoReL goes down and then up again while the network converges to the ϵ-CCE. This happens when the nodes change their configuration from MinSF to NoReL (at the very beginning) and when $\overline{\lambda}$ is changed (after about 210,000 packets). Moreover, the nodes rapidly adopt the configuration they had already learned at the very beginning after 10 days, when the initial sending rate is used again. NoReL requires longer time than ADR and DyLoRa to converge to a stable configuration. This is because of its completely distributed nature. In particular, the

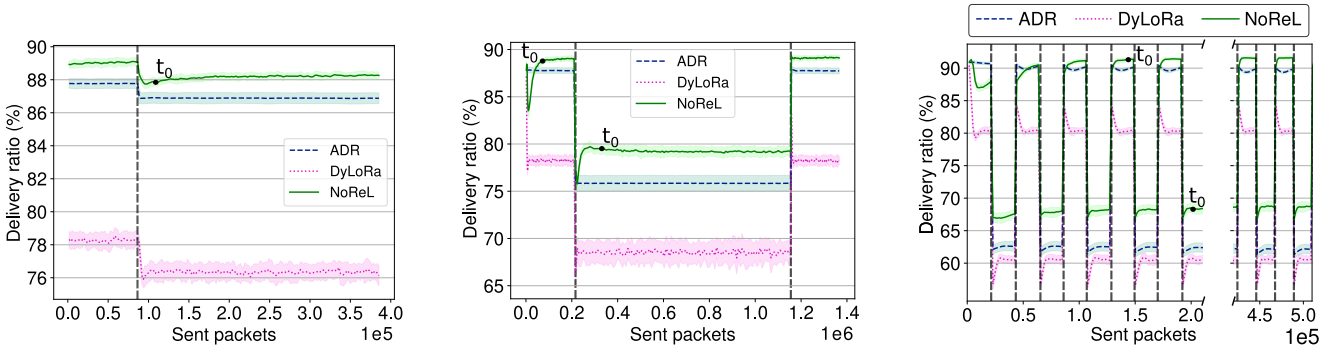


Fig. 4: Delivery ratio for scenarios where (a) 100 new nodes are added to the network, (b) the average sending rate is increased by a factor of 5, and (c) the channel conditions alternate between two settings every 12 hours.

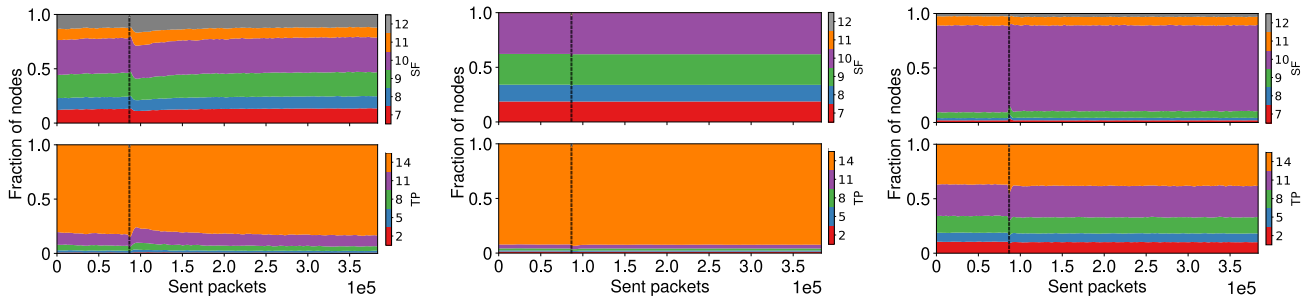


Fig. 5: SF and TP assignment over time for (a) NoReL, (b) ADR and (c) DyLoRa in the scenario with added nodes.

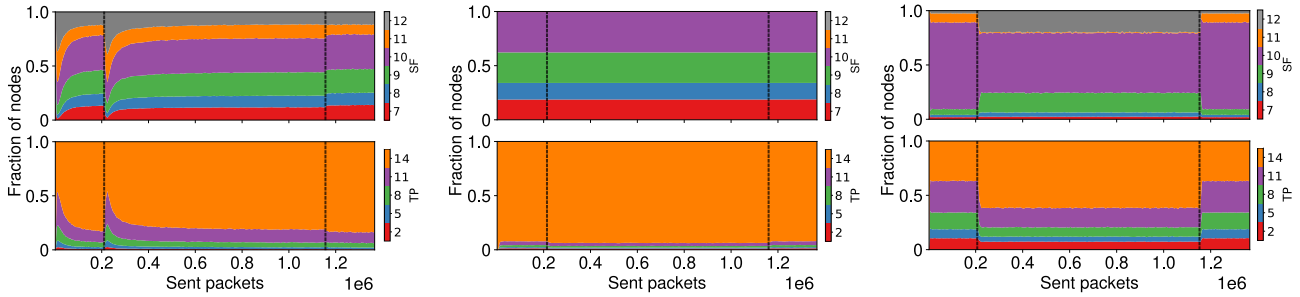
convergence time of NoReL at normal network traffic is around 82,800 packets and 115,000 packets at high traffic, as shown by the black circles in Figure 4b. The convergence time is higher for the increased traffic as there is a higher probability of packets being dropped, which results in longer time for the nodes to complete their rounds. However, the transient only takes place the first time a random state is met, after which the nodes learn the best configuration and use it every time they return to such a state. This explains why NoReL returns almost instantaneously to the highest delivery ratio after the traffic goes back to the initial conditions.

NoReL achieves the highest delivery ratio; compared to ADR, 3.6% higher with high network traffic and around 1% higher at normal traffic. Surprisingly, NoReL presents a peak shortly after the sending rate is increased, before converging to a slightly lower delivery ratio. This happens because the overall performance of the configuration used at that peak does not correspond to the equilibrium achieved by NoReL. In fact, converging to an ϵ-CCE does not imply that the network reaches its optimum configuration then. Instead, the convergence is in the ϵ-neighborhood of a CCE [15], i.e., the overall performance can deviate from the CCE by ϵ.

Figure 6 shows the SF and TP assignment over time in this case. With NoReL, nodes again converge to their strategies in each random state and then quickly adopt the configuration they previously had when a certain state is met again (Figure 6a). Instead, ADR is mainly unaffected by the increased network traffic, except for small changes in the TP distribution (Figure 6b). On the other hand, DyLoRa quickly converges to a SF and TP assignment at each network traffic level (Figure 6c).

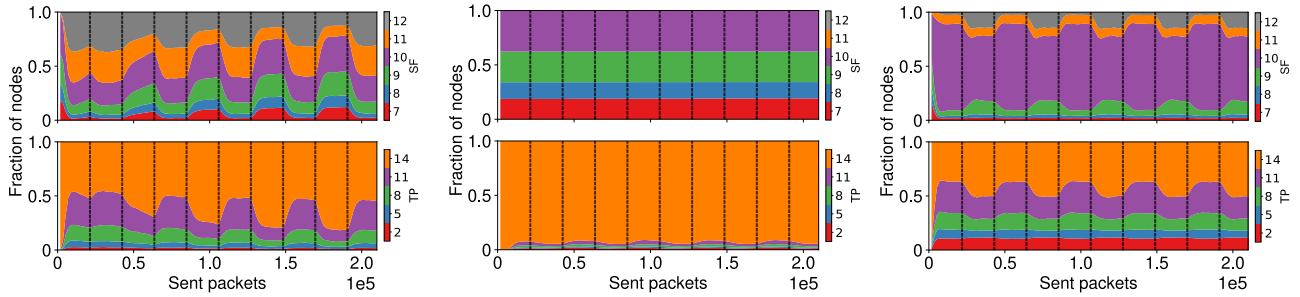
Changes in channel conditions. In the third scenario, the channel conditions vary following the empirical results presented in [9], i.e., the variance and mean of the received power change periodically by approximately 10 dBm in the worst case [9, Figure 5]. Accordingly, the path loss parameters (γ, σ) were varied between the two settings of $(2.32, 3.54)$ and $(3.32, 7.08)$. The device margin for ADR was set to $\sigma + 8$ dB in the experiments, following a preliminary analysis showing that such a margin had to be σ-dependent to avoid instability.

Figure 4c shows that NoReL achieves the highest delivery ratio throughout, about 7% higher than ADR and DyLoRa under the worst channel conditions. Note that the two channel settings starting at 0 and 12 hours (every 21,500 packets) represent distinct random conditions that involve two independent convergence periods. The figure shows that 12 hours are not enough for NoReL to converge in either random condition within a single time window. However, the nodes resume learning when similar channel conditions occur again, starting from the best configuration learned until then. Thus, the nodes do not encounter a drop in delivery ratio, for instance, towards the end of the simulated time. Nevertheless, there is some transient for NoReL after each change is implemented, where the delivery ratio starts from a slightly lower value and quickly sets to the converged value. This is simply the time it takes



(a) (b) (c)

Fig. 6: SF and TP assignment over time for (a) NoReL, (b) ADR and (c) DyLoRa in the scenario with increased traffic.



(a) (b) (c)

Fig. 7: SF and TP assignment over time for (a) NoReL, (b) ADR and (c) DyLoRa in the scenario with varying channel settings.

for all the nodes to receive information from the gateway about the random condition. In contrast, nodes do not leverage previously-known best configurations with ADR or DyLoRa.

Finally, Figure 7 shows the SF and TP assignment over time in the considered scenario. Interestingly, NoReL converges to a stable assignment in two different random conditions (Figure 7a). It is also worth noting that lower SFs and higher TPs are used in the state corresponding to $(\gamma, \sigma) = (2.32, 3.54)$, while the opposite happens when the channel conditions deteriorate. The assignment of ADR is again mostly the same across all simulation time, except for small changes in the TPs (Figure 7b). Finally, DyLoRa converges to a solution every time the channel conditions change; however, DyLoRa tends to assign higher SFs and TPs when the channel conditions deteriorate (Figure 7c), in contrast to NoReL.

TABLE II: Summary of results.

Scenario	Average delivery ratio (%)				
	MinSF	Tolio	ADR	DyLoRa	NoReL
200 nodes	94.2	95.1	92.4	87.6	95.6
2,000 nodes	84.7	84.3	84.4	69.3	85.9
2 gateways	94.6	95.2	93.1	82.9	95.2
Non-uniform	88.3	88.2	88.5	73.3	88.7
100 more nodes	-	-	86.8	76.4	88.2
$\overline{\lambda} = 0.001$	-	-	87.7	78.2	89.2
$\overline{\lambda} = 0.005$	-	-	75.8	68.5	79.2
$\gamma = 2.32, \sigma = 3.54$	-	-	89.6	80.5	91.5
$\gamma = 3.32, \sigma = 7.08$	-	-	62.1	60.6	68.8

D. Summary

Table II summarizes the obtained results. Clearly, NoReL outperforms all other schemes for all considered scenarios. This happens as it takes into consideration the delivery ratio of individual nodes to make decisions. Unsurprisingly, MinSF and Tolio perform well in static conditions, as they are specifically designed for such use cases; they are not suitable for the dynamic environments of real-world deployments though. In contrast, ADR and DyLoRa are adaptive, therefore, they allow nodes to set their parameters according to the actual environmental conditions. However, they mostly rely on the SNR of packets sent by nodes, which does not account for all the factors that may arise in dynamic scenarios. In conclusion, the obtained results establish that NoReL is flexible and suitable for diverse, highly-dynamic environments.

V. Conclusion

This article has proposed NoReL, a game-theoretic approach for setting SFs and TPs to improve communication reliability in LoRa networks. Specifically, a no-regret learning procedure was devised for nodes to independently learn the utilities of their actions and maximize their delivery ratio by choosing the appropriate SF and TP. This procedure is shown to converge to an equilibrium where nodes are satisfied with their utilities. Extensive simulations have shown that NoReL achieves a higher average delivery ratio than the state of the art. Moreover, it quickly obtains a new equilibrium in dynamic scenarios where the channel and network conditions suddenly change. The flexibility of the proposed solution makes it ideal for any type

of LoRa network. The implementation of NoReL in real-world LoRa deployments is left as a promising future work.

ACKNOWLEDGMENTS

We would like to thank Yinghui Li for clarifying some details of DyLoRa, Thomas Marchioro for providing the source code used in [14], and the CSC – IT Center for Science for provisioning the computational resources used in the study.

REFERENCES

[1] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, "Long-range communications in unlicensed bands: the rising stars in the IoT and smart city scenarios," *IEEE Wireless Communications*, vol. 23, no. 5, pp. 60–67, 2016.

[2] J. Petäjäjärvi, K. Mikhaylov, A. Roivainen, T. Hanninen, and M. Pettissalo, "On the coverage of LPWANs: range evaluation and channel attenuation model for LoRa technology," in *2015 14th International Conference on ITS Telecommunications*.

[3] A. Mahmood, E. Sisinni, L. Guntupalli, R. Rondón, S. A. Hassan, and M. Gidlund, "Scalability analysis of a LoRa network under imperfect orthogonality," *IEEE Trans. Ind. Inform*, 2018.

[4] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa low-power wide-area networks scale?" in *ACM MSWiM '16*.

[5] V. Toro-Betancur, G. Premsankar, M. Słabicki, and M. Di Francesco, "Modeling Communication Reliability in LoRa Networks with Device-level Accuracy," in *INFOCOM 2021*.

[6] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita, and I. Tinnirello, "Impact of LoRa imperfect orthogonality: Analysis of link-level performance," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 796–799, 2018.

[7] G. Premsankar, B. Ghaddar, M. Słabicki, and M. Di Francesco, "Optimal configuration of LoRa networks in smart cities," *IEEE Trans. Ind. Inform*, vol. 16, no. 12, pp. 7243–7254, 2020.

[8] D. Garlisi, I. Tinnirello, G. Bianchi, and F. Cuomo, "Capture aware sequential waterfilling for LoRaWAN adaptive data rate," *IEEE Trans. Wireless Commun.*, 2021.

[9] T. Ameloot, P. Van Torre, and H. Rogier, "Variable link performance due to weather effects in a long-range, low-power LoRa sensor network," *Sensors*, vol. 21, no. 9, 2021.

[10] ——, "A compact low-power LoRa IoT sensor node with extended dynamic range for channel measurements," *Sensors*, vol. 18, no. 7, 2018.

[11] M. Centenaro, L. Vangelista, and R. Kohno, "On the impact of downlink feedback on LoRa performance," in *IEEE PIMRC 2017*.

[12] Y. Li, J. Yang, and J. Wang, "DyLoRa: Towards energy efficient dynamic LoRa transmission control," in *INFOCOM 2020*.

[13] D.-T. Ta, K. Khawam, S. Lahoud, C. Adjih, and S. Martin, "LoRa-MAB: A flexible simulator for decentralized learning resource allocation in IoT networks," in *IFIP WMNC 2019*.

[14] A. Tolio, D. Boem, T. Marchioro, and L. Badia, "Spreading factor allocation in LoRa networks through a game theoretic approach," in *IEEE ICC 2020*, pp. 1–6.

[15] S. Hart and A. Mas-Colell, "A Simple Adaptive Procedure Leading to Correlated Equilibrium," *Econometrica*, 2000.

[16] J.-W. Lee, A. Tang, J. Huang, M. Chiang, and A. R. Calderbank, "Reverse-engineering MAC: A non-cooperative game model," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 6, pp. 1135–1147, 2007.

[17] M. A. Khan and Y. Sun, "Non-cooperative games with many players," ser. Handbook of Game Theory with Economic Applications. Elsevier, 2002, vol. 3, ch. 46, pp. 1761–1808.

[18] S. Samarakoon, M. Bennis, W. Saad, and M. Latva-aho, "Backhaul-Aware Interference Management in the Uplink of Wireless Small Cell Networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 11, pp. 5813–5825, 2013.

[19] D. Magrin, M. Centenaro, and L. Vangelista, "Performance evaluation of LoRa networks in a smart city scenario," in *IEEE ICC 2017*.

[20] P. Kumari, H. P. Gupta, and T. Dutta, "An incentive mechanism-based stackelberg game for scheduling of LoRa spreading factors," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2598–2609, 2020.

[21] The Things Network, "Adaptive Data Rate," https://www.thethingsnetwork.org/docs/lorawan/adaptive-data-rate/.

[22] M. Słabicki, G. Premsankar, and M. Di Francesco, "Adaptive configuration of LoRa networks for dense IoT deployments," in *IEEE/IFIP NOMS 2018*.

[23] I. Ilahi, M. Usama, M. O. Farooq, M. Umar Janjua, and J. Qadir, "LoRaDRL: Deep reinforcement learning based adaptive PHY layer transmission parameters selection for LoRaWAN," in *IEEE LCN 2020*, pp. 457–460.

[24] R. Kerkouche, R. Alami, R. Féraud, N. Varsier, and P. Maillé, "Node-based optimization of LoRa transmissions with multi-armed bandit algorithms," in *2018 25th International Conference on Telecommunications (ICT)*, 2018, pp. 521–526.

[25] M. Benaïm, "Dynamics of stochastic approximation algorithms," in *Séminaire de Probabilités XXXIII*. Springer Berlin Heidelberg, 1999, pp. 1–68.

[26] S. Batewela, C. Liu, M. Bennis, H. A. Suraweera, and C. S. Hong, "Risk-sensitive task fetching and offloading for vehicular edge computing," *IEEE Commun. Lett.*, 2020.

[27] LoRa Alliance, "LoRaWAN Specification v1.1."

[28] J. Nash, "Non-cooperative games," *Annals of mathematics*, pp. 286–295, 1951.

[29] N. Blenn and F. Kuipers, "LoRaWAN in the wild: Measurements from the things network," *arXiv preprint arXiv:1706.03086*, 2017.

[30] *SX1272/73: Low Power Long Range Transceiver*, Semtech, March 2017, rev. 3.1.

[31] Semtech, "An In-depth look at LoRaWAN Class A Devices," https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lorawan-class-a-devices.

[32] C. Gustafson, T. Abbas, D. Bolin, and F. Tufvesson, "Statistical modeling and estimation of censored pathloss data," *IEEE Wireless Communications Letters*, 2015.

[33] G. Callebaut and L. Van der Perre, "Characterization of LoRa point-to-point path loss: Measurement campaigns and modeling considering censored data," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1910–1918, 2020.