Optimizing and updating LoRa communication parameters: a Machine Learning approach

Article *in* IEEE Transactions on Network and Service Management · July 2019

DOI: 10.1109/TNSM.2019.2927759

3 authors, including:

Ruben M. Sandoval
Universidad Politécnica de Cartagena
10 PUBLICATIONS **48** CITATIONS
SEE PROFILE

Antonio-Javier Garcia-Sanchez
Universidad Politécnica de Cartagena
72 PUBLICATIONS **599** CITATIONS
SEE PROFILE

Optimizing and updating LoRa communication parameters: a Machine Learning approach

Ruben M. Sandoval, *Student Member, IEEE,* Antonio-Javier Garcia-Sanchez and Joan Garcia-Haro, *Member, IEEE*

***Abstract*—LoRa is an extremely flexible low-power wide-area technology that enables each IoT node to individually adjust its transmission parameters. Consequently, the average per-node throughput of LoRa-based networks has been mathematically formulated and the optimal network-level configuration derived. For end nodes to update their transmission parameters, this centrally-computed global configuration must then be disseminated by LoRa gateways. Unfortunately, the regional limitations imposed on the usage of ISM bands –especially those related to the maximum utilization of the band– pose a potential handicap to this parameter dissemination. To solve this problem, a set of tools from the Machine Learning field have been used. Precisely, the updating process has been formulated as a Reinforcement Learning (RL) problem whose solution prescribes optimal disseminating policies. The use of these policies together with the optimal network configuration has been extensively analyzed and compared to other well-established alternatives. Results show an increase of up to 147% in the accumulated per-node throughput when our RL-based approach is employed.**

***Index Terms*—Reinforcement Learning, LoRa, throughput optimization, Machine Learning.**

I. Introduction

Low-power wide-area (LPWA) networks have recently been attracting great attention in the IoT community. Expected to be one of the key enablers of the fourth industrial revolution [1], LPWA networks provide a good balance between long-range and low-power communications at the expense of throughput. Among the most popular LPWA technologies, LoRa [2] excels for its extremely long battery lifetimes and reduced costs, especially when compared to its direct competitors (such as Narrowband IoT) [3]. Based on a proprietary *chirp spread spectrum* modulation, one of the most remarkable features of LoRa is its ability to adjust communication parameters to foster either low-power or robust links. The two main parameters are: the Spreading Factor (SF), which regulates the ratio between symbol rate and chip rate, and the Coding Rate (CR), which adjusts the ratio between the number of payload bits and the length of the error-correction code. By increasing the SF or reducing the CR, more robust transmissions are achieved at the cost of lengthening the Time on Air (ToA) of packets, and consequently, power consumption.

This flexibility is coupled with a low-cost philosophy that not only reduces capital expenditures by cutting down transceiver costs, but also decreases operational expenses by operating in sub-GHz license-free bands. In particular, LoRa works in the unlicensed ISM bands at 433, 868 or 915 MHz, depending on the region of operation [4]. However, due to this use of ISM bands, LoRaWAN networks (that is, networks where the LoRa modulation technique is used and where the upper MAC and Network layers are implemented following the LoRa Alliance standard [5]) are, in many countries, subject to strict limitations on the amount of time they can use the shared wireless medium [6]. The amount of time LoRa devices can use the shared wireless medium is known as Transmission Duty Cycle (TDC) and is usually expressed as the percentage of time nodes are allowed to access the shared wireless medium per hour, e.g. 1% of the time (which translates into 36 seconds per hour).

All in all, LoRaWAN networks are extremely customizable wireless networks capable of individually adjusting the transmission parameter of their constituent devices. Therefore, from a network perspective, an optimal global configuration that encompasses each node transmission parameter can be found. This would allow us to intelligently optimize the use of the available communication bandwidth (the scarcest network resource) so that some predefined figure of merit, such as the throughput, could be maximized. Note that intelligently managing the limited bandwidth (to increase throughput) is of special importance in LoRaWAN networks. As many authors [7]–[9] have pointed out, their reduced bitrate is one of the most limiting factors in enabling a ubiquitous, real-time IoT. This is precisely what we have focused our efforts on in the first part of this work. We have mathematically modeled the average per-node throughput as a function of the packet-generation behavior of nodes so their best transmission parameters can be found.

Since this optimal global configuration must be derived by some central entity (e.g. a LoRa gateway or a remote server commanded by it), the individual configuration of each IoT node must then be disseminated. In doing so, we have found that the TDC restriction which also applies to gateways may become an insurmountable limitation, which can jeopardize the effectiveness of this updating process. Therefore, intelligently managing this scarce resource, i.e. the TDC, becomes a must. The derivation of efficient ways of disseminating new transmission configurations has been posed as a Reinforcement Learning (RL) problem whose solution prescribes optimal updating policies. These policies are shown to outperform traditional alternatives in terms of accumulated per-node throughput.

Therefore, the main contribution of this paper is twofold: firstly, a global network configuration that maximizes through-

This research has been supported by the project AIM, ref. TEC2016-76465-C2-1-R (AEI/FEDER, UE), and the regional projects e-DIVITA, ref. 20509/PDC/18 (Proof of Concept, 2018) and ATENTO, ref. 20889/PI/18 (Fundacion Seneca, R. Murcia). Ruben M. Sandoval also thanks the Spanish MECD for an FPU ref. FPU14/03424. The authors are with the Department of Tecnologías de la Información y Comunicaciones, Universidad Politécnica de Cartagena (UPCT), Antiguo Hospital de Marina, Cartagena Murcia 30202, Spain, e-mail: {ruben.martinez, antoniojavier.garcia, joang.haro}@upct.es.

put has been analytically derived. Secondly, this configuration is intelligently disseminated to nodes using a set of tools from the RL field.

The rest of the paper is organized as follows. The related work is presented in Section II. The mathematical model for throughput maximization is formulated in Section III. In Section IV, the TDC limitation is discussed to illustrate why this constraint may severely affect the dissemination of the global network configuration. Then, to address this issue, we introduce an RL-based algorithm for updating the configuration of individual IoT nodes in such a way that the accumulated per-node throughput is maximized. In Section V, we elaborate on the order in which the described steps must be executed. Implementation details and results are presented in Section VI. Finally, Section VII summarizes the conclusions and lessons learned.

II. Related Work

Many works have looked into the effects that different LoRa transmission configurations have on IoT networks. For example, determining the overall network capacity of single-gateway networks [10]–[14], or deriving intricate SF-allocation schemes that maximize a fairness criteria [15], [16]. Similarly, other works such as [17]–[19] have proposed methods to increase network throughput by deriving every node transmission parameter. However, each of these works fails to fully regard one of the following aspects:

- They do not acknowledge the heterogeneity of IoT networks: [10]–[12], [14]–[16], [18], [19]. These works either assume that all nodes generate packets at the same rate, that nodes can only send packets of a fixed length, or that all IoT nodes are assumed to have the same importance and role in the network.
- They oversimplify the mathematical model by not considering some LoRa physical phenomena like the *capture effect* (described in Section III) [10], [17] or assume unrealistic node distributions [12], [14], [17] (e.g. nodes are evenly distributed).
- A figure of merit is analyzed but not maximized [10]–[14], a parameter is simply varied (like the number of nodes) and its effects quantified in terms of throughput or delay.

Nevertheless, it is indisputable that LoRa-related literature has increased greatly during the last 3 years. With the aim of taking a step forward in this direction, we take inspiration from the above literature and extend it to: acknowledge the heterogeneity of IoT networks by letting each individual node generate packets of different importance and length at different rates. We fully regard the *capture effect* and let each node sit in an arbitrary position; hence, not forcing uniform/radial distributions. Finally, we do this with the aim of maximizing network throughput and thus, we derive a global network configuration that attains this objective (attaining improvements of more than 140%).

In contrast to the topic of "analyzing/maximizing LoRa performance", much less studied is the problem of updating the transmission parameters of nodes once the aforementioned global network configuration has been derived. Especially, when the TDC constraint is considered. To the best of authors' knowledge, this is the first time this problem has been addressed. However, a few studies have underlined that this TDC limitation can potentially endanger the deployment of future large-scale LoRaWAN Networks [12], [20], [21]. In a previous work [22] we proposed a mathematical model, based on classic Markov Decision Processes (not RL), for determining the optimal transmission parameters of LoRa nodes. However, this work is oriented from a node-centric approach, neglecting packet collisions and not attempting to derive a network-wise optimal configuration. Furthermore, whereas in [22] the transmission configuration is computed within nodes (thus, there is no need for dissemination), in this work the network-wise optimal configuration is obtained within gateways and hence, we also propose a mathematically instructed way of updating the configuration of IoT nodes. This updating mechanism is based on RL and maximizes a criteria based on network throughput.

III. LoRaWAN Network - Performance analysis and optimization

In this section we first analyze the behavior of LoRa nodes (subsection III-A), paying particular attention to when and how packets may collide. Based on this analysis, we mathematically describe the performance of LoRaWAN networks as a function of the transmission characteristics of constituent nodes (subsection III-B). Finally, we present a method to derive node parameters in such a way that the global performance is maximized (subsection III-C).

A. LoRa collision window analysis

The MAC of LoRaWAN can be regarded as a variant of the pure ALOHA algorithm with no collision avoidance/detection. When orthogonal SFs are considered, each of these SFs represents a virtual channel through which information can be conveyed [4]. Note that some works [23]–[25] have indicated that, under some circumstances of low Signal-to-Noise Ratio (SNR), SFs are not perfectly orthogonal. However, since the effects of this imperfect orthogonality have been proved to depend on: (i) the specific LoRa transceiver, (ii) perceived SNRs, and (iii) number of deployed nodes [24], and to make the mathematical model tractable, in this work we assume that SFs are orthogonal.

Furthermore, as demonstrated in [13], [14], the so-called *capture effect* causes more powerful LoRa packets to prevail over weaker ones, even when both transmissions take place on the same SF simultaneously. Therefore, for a packet to be successfully received, the following two conditions must be satisfied:

- The SNR of the received packet must be high enough. This will influence the bit-error-rate and, as expected, larger SNR values lead to more robust transferences.
- No collisions must occur during the transmission of the packet, or the *capture effect* will take place. That is, one of the following conditions must be met:

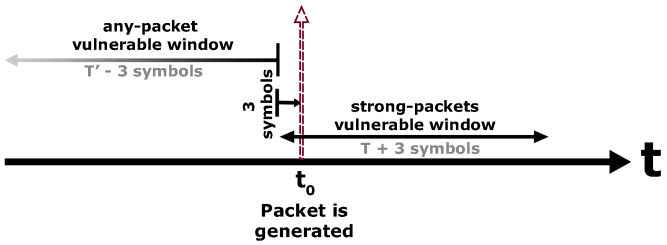


Fig. 1: The two vulnerable windows of a packet transmission: the *any-packet vulnerable window* and the *strong-packet vulnerable window*.

1) No other packets (with the same SF) should be sent during the transmission of the packet under consideration.
2) Although there is a same-SF interfering packet (referred to as P'): (i) the reception power of such a packet should be at least 6dB weaker than the interfered transmission [16], [17], [26] (referred to as P) and (ii) the gateway has not locked on it yet (i.e. has not fully started to receive P'). This happens when, either P started earlier than P', or P' was generated less than 3 symbols sooner than P (the time needed for the gateway to lock on the transmission).

Derived from the above second condition, Fig. 1 graphically represents the *capture effect* to further describe the two vulnerable windows of packet transmission. We consider a scenario in which a packet P is generated by an IoT mote at t_0 and whose Time on Air (ToA) equals T seconds. For the purpose of illustration, we also consider that there is another potentially interfering IoT mote generating packets P' under the same SF and whose ToA is T'. If P' is transmitted within the interval [$t_0 - T'$, $t_0 - 3$ symbols], it will unavoidably interfere with P (as the gateway would have locked on P' already). This interval defines the so-called *any-packet vulnerable window* for any potentially interfering packet P' with ToA T'. When the *any-packet vulnerable window* is over, the *strong-packet vulnerable window* starts. Then, only packets P', with sufficiently high transmission power may interfere with P. Specifically, for P to be successfully received, its perceived reception power must be at least 6dB stronger than the perceived reception power of P' [16], [17], [26]. Note that the length of this vulnerable window is equal to T seconds + 3 symbols.

In light of the capture effect, when a packet collision takes place in the *strong-packet vulnerable window*, the receiver may still decode the strongest transmission. This effect makes the LoRaWAN MAC performance slightly superior to ALOHA as demonstrated by R. Brandborg *et al.* [27].

B. LoRa performance analysis

In order to mathematically analyze the performance of the LoRaWAN MAC, we assume that the packet-generation process at each node i follows a Poisson distribution, with a mean generation rate of λ_i packets per second. This assumption, which is extensively used in current IoT literature [10], [11], [14], [17], [28]–[33], is reasonable for networks in which IoT motes generate packets based on (independently) detected events, e.g. via sensor readings [22]. Note that Poisson-like traffic naturally arises when sensors are constantly polled and checked for certain conditions (e.g. pressure over a given threshold or the temperature matches some value) and not when values are periodically sent to the gateway. Therefore, we focus on the first type of networks (event-driven networks) in the rest of the work. Note that many industrial/monitoring-oriented IoT networks can be considered to be event-driven networks. Furthermore, in this kind of event-driven network, it may be interesting to let motes generate packets of different importance. For example, some nodes might be in charge of supervising critical assets whereas others may simply be reporting low-importance measurements. Thus, we acknowledge the heterogeneity of an IoT network by letting each node i generate packets of different importance (G_i), length (L_i bytes), and at different rates (λ_i). Taking this into consideration, the average throughput per node Γ_{raw} (expressed in bytes per second) for a network of N IoT motes can be computed as:

$$\Gamma_{raw} = \frac{1}{N} \sum_{i=1}^{N} \lambda_i \cdot L_i \cdot G_i. \tag{1}$$

However, Eq. 1 does not consider the percentage of packets that get lost due to either low SNR or packet collisions. For the computation of the former, the Packet Reception Rate (PRR) of each transmission must be regarded. This figure evaluates the ratio of packets successfully received in a collision-free scenario. Unlike many works [14], [24], [34], we acknowledge the stochastic nature of the reception process by modeling the PRR as a function of SF, CR and SNR. We mathematically model the PRR as $PRR = f(SF, CR, SNR)$, which gives us the probability with which such a packet will be successfully received by the gateway. The specific characterization of PRR vs SNR can be found in [22] for a wide variety of SF and CR values. This way, the average effective throughput per node Γ of the IoT network is:

$$\Gamma = \frac{1}{N} \sum_{i=1}^{N} \lambda_i \cdot L_i \cdot G_i \cdot PRR_i \cdot \overline{\phi_i}, \tag{2}$$

where $\overline{\phi_i}$ represents the probability of no collisions when node i transmits. This probability can be derived from the traditional ALOHA analysis if the two vulnerable windows described above are considered. Starting with the *any-packet vulnerable window*, let $N_j \in N$ be the subset of all the nodes that employ the SF j (the same SF used by node i). The probability that none of them generate a packet within the *any-packet vulnerable window* of node i ($\overline{\phi^i_{apvw}}$) is:

$$\overline{\phi^i_{apvw}} = \prod_{\substack{n \in N_j \\ n \neq i}} e^{-(T_n - 3 \text{ symbols}) \cdot \lambda_n}, \tag{3}$$

with T_n the ToA of packets generated by a node $n \in N_j$, and λ_n its packet generation rate. The above formula results from the assumption that motes generate poisson-based traffic and from the classic ALOHA analysis [11]. On the other hand, to study the *strong-packet vulnerable window*, let $N^{\rho}_j \in N_j$ be the set of nodes whose SF is j and whose reception power (RXP), perceived by the gateway, is greater than or equal to ρ dB. The probability that none of these nodes will interfere

with a node i in its *strong-packet vulnerable window* ($\overline{\phi^i_{spvw}}$) is:

$$\overline{\phi^i_{spvw}} = \prod_{\substack{n \in N_j^{RXP_i-6} \\ n \neq i}} e^{-(T_i+3 \text{ symbols})\cdot\lambda_n}. \quad (4)$$

Note that in the equation above, we only consider those nodes whose RXP is greater than or equal to the RXP of the node under consideration (i) minus 6 dBs, that is, $n \in N_j^{RXP_i-6}$. Finally, the probability of no collisions in the transmission of the node i is $\overline{\phi_i} = \overline{\phi^i_{apvw}} \cdot \overline{\phi^i_{spvw}}$.

C. Performance maximization problem

Having a closed-form formula of Γ allows us to mathematically maximize it. That is, we can make each node of the network $(n_1, ..., n_i, ..., n_N)$ employ a specific transmission configuration (c_i) in such a way that the average effective throughput per node is maximized. If configurations are mutually exclusive (that is, a node must choose one and only one transmission configuration), this problem can be posed as a classic combinatorial optimization task. Unfortunately, an exhaustive search is not feasible due to the sheer number of combinations, and no *polynomial-time algorithm* is known to be applicable in a general case, especially when the problem under optimization does not present a linear or convex form. To avoid this, instead of forcing nodes to choose a single configuration, we let nodes use any combination of them, thus applying a linear relaxation over the binary (0-1) combinatorial problem. More formally, let $\mathbf{c_i}$ be the configuration vector used by node i. The *k-th* position of this vector (c_i^k) indicates the percentage of packets generated by node i that employs the specific transmission configuration k. Naturally, $\sum_{k \in K} c_i^k = 1$, where K is the total number of different SF-CR configurations and $0 \leq c_i^k \leq 1$ for $\forall k \in K$. By allowing motes to use any combination of configurations, we transform this problem into a bounded continuous maximization problem, for which faster algorithms are available. To solve such a maximization problem, we have to determine the $\mathbf{C} = \{\mathbf{c_1}, \mathbf{c_2}, ..., \mathbf{c_N}\}$ that maximizes Γ, i.e. $\mathbf{C_{OPT}}$. Therefore, the solution to the optimization problem is a matrix $\mathbf{C}$ of dimensions $K \times N$ where, as stated before, the entry c_i^k indicates the percentage of packets generated by node i that employs the specific transmission configuration k. Eq. 5 formally formulates the maximization problem.

$$\begin{aligned} \underset{\mathbf{C}}{\text{maximize}} \quad & \Gamma = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \lambda_i \cdot c_i^k \cdot L_i \cdot G_i \cdot PRR_i^k \cdot \overline{\phi_i^k} \\ \text{subject to} \quad & c_i^k \leq 1,\ i = 1, \ldots, N \text{ and } k = 1, \ldots, K. \\ & c_i^k \geq 0,\ i = 1, \ldots, N \text{ and } k = 1, \ldots, K. \\ & \sum_{k=1}^{K} c_i^k = 1, \end{aligned} \quad (5)$$

where PRR_i^k is the PRR of node i using the transmission configuration k, and $\overline{\phi_i^k}$ is the probability that a collision does not occur when node i transmits under configuration k. Note that for each node, we scale its packet generation rate λ_i by c_i^k and then aggregate it for all $k \in K$.

IV. Dissemination of the optimal configuration

Whereas in the previous section we focused on deriving the optimal global configuration $\mathbf{C_{OPT}}$ (middle of Fig. 2), in this section we elaborate on how and when this configuration is disseminated to the network (left and right of Fig. 2). First, we introduce a few background concepts (subsection IV-A) so the RL problem can be fully described (subsection IV-B). Finally, we present two methods that foster the reutilization of derived policies and speed up the learning process (subsections IV-C and IV-D).

A. General outlook of the problem

The global network configuration $\mathbf{C_{OPT}}$ that maximizes Γ must be computed by a centralized entity, such as a LoRa Gateway or a remote server under its command. This is a strict requirement because only gateways have a global vision/perfect knowledge of the network: perceived SNRs, length of packets, generation rates of nodes, etc. Due to the nature of LPWA networks, IoT motes remain asleep most of their lives, oblivious to what occurs in the network.

Therefore, after gathering information about nodes, the gateway may decide to (re)compute $\mathbf{C_{OPT}}$ by solving Eq. 5. Once this is accomplished, gateways should send the computed $\mathbf{c_i}$ to each node i, so it can update its transmission configuration. However, as indicated in Section I, LoRaWAN networks may be subject to the TDC limitation, restricting their capacity to use the shared medium at their will. This unavoidably poses a potential handicap to the Configuration Updating process (CU process); the procedure by which gateways update the transmission configuration of nodes. Since the TDC is extremely scarce (normally, 36 seconds or less per hour) and the ToA of updating packets can be relatively long (up to 2-3 seconds), gateways should derive intelligent policies to update node configurations. This is especially true for very populated networks, where gateways may easily run out of TDC. If a gateway spends all its TDC before the CU process is completed, only a subset of nodes will have their configuration updated (until some TDC is again available); leading to an undesirable network state in which the global throughput is severely degraded. Furthermore, LoRa nodes are asleep most of the time, offering very short receiving windows after each of their transmissions. Therefore, gateways can only update the configuration of a node immediately after receiving a packet from such a node.

The CU process may again be posed as a maximization problem, whose solution is a set of actions to be taken. We propose applying a set of mathematical tools commonly applied in RL to solve this problem for two main reasons:

- The problem of determining an optimal sequence of actions in such a way that some performance criteria is maximized perfectly aligns with RL.That is, we have an action-taking agent whose optimal behavior must be found. In these settings, we can leverage the power of RL tools to make such an entity *automatically* learn optimal behavior through experimentation (the feedback signal known as reward).

- When the *event horizon* (that is, the time span over which an action has some measurable impact) is particularly large, resorting to traditional combinatorial optimization algorithms is unfeasible. For this purpose, RL algorithms can exploit event independence to converge faster to an optimal policy.

B. RL description of the problem

We now present the formalism associated to the problem so that it can be formulated as a Markov Decision Process, the basic mathematical tool (and constituent element) of any RL algorithm. The basic and global idea is that, once $\mathbf{C_{OPT}}$ is computed, every time the receiving window of a mote is opened, gateways must decide which action to take; that is, to update or not the configuration of such a node (a_{up} and $\overline{a_{up}}$ respectively). Updating it alters the current global configuration (denoted simply as $\mathbf{C^t}$ for an instant t), and the average throughput per node of the network Γ^t (similarly, for an instant t). Furthermore, updating the configuration of a node implies sending a message with $\mathbf{c_i}$ to node i, and by doing so, gateways spend some of their TDC. The state (s) of a given IoT network can be described in terms of the set of already updated nodes ($N_u \in N$), the set of non-updated nodes ($N_{\overline{u}} \in N$), the active receiving windows $\mathbf{W} = (w_1, w_2, ..., w_i, ..., w_N)$ with $w_i \in \{0, 1\}$, and the remaining TDC of the gateway. Naturally $N_u \cup N_{\overline{u}} \iff N$. Note that $\mathbf{W}$ indicates which nodes are potentially available to update at that precise instant. Also, note that since packets are randomly generated by IoT motes, $\mathbf{W}$ is a random variable and thus, so is s. Since the throughput depends on this state, let $\Gamma^t(s)$ be the average effective throughput per node of the network for a given state s, and at an instant t. Furthermore, let $\mathbf{S}$ be the set of all feasible states of the network (whose size $|S|$, based on its definition, is equal to $2N + 1$), and $\mathbf{A} = \{a_{up}, \overline{a_{up}}\}$ the set of all actions a gateway may take. The goal of the CU process is to follow an updating policy π such as, for any given state $s \in \mathbf{S}$, an optimal action $a \in \mathbf{A}$ is taken.

We propose measuring the performance of a policy (P_π) as the expected accumulated Γ obtained over a given period of T seconds when policy π is followed. P_π, shown in Eq. 6, is expressed in bytes and represents the expected number of successfully transmitted bytes (weighted by the importance of such bytes) in a given period T. Therefore, the optimal policy (π^*) is simply the policy for which P_π is maximized.

$$P_\pi = \mathbb{E}\left[\int_{t=0}^{T} \Gamma^t(s)\, dt \mid \pi\right]. \tag{6}$$

To derive π^*, more traditional reinforcement learning approaches use tabular methods for which good convergence properties are known. However, for medium to large-sized IoT networks, $\mathbf{S}$ is too big for π^* to map every state to an action [35]. Newer approaches resort to approximation methods (such as Artificial Neural Networks, ANNs) to model action policies [36]. The general idea is to iteratively improve the ANN-defined policy π (by changing its weights vector, θ) to improve its performance; in our case, defined by P_π. Hence, π can be considered a function of θ, the weights of the ANN that implements such a policy: $\pi = f(\theta)$, however we use notation π for simplicity. Depending on how these weights are optimized, we can categorize ANN-based algorithms into two broad groups, namely gradient-based and gradient-free optimization methods [37]. The former would typically employ gradient-ascent to improve θ, whereas the latter does not.

For RL problems where episodes (T) are large and the effects of actions are long-lasting, the convergence of gradient-free approaches, such as Evolution Strategies (ES) [38], is generally better than gradient-based techniques (e.g. Policy Gradients) [39]. Since the CU process can last several hours (depending on the size of the network), and the effect of updating a node persists until the end of such a process, we propose employing ES for this matter. In particular, due to the remarkably good performance in modern RL tasks, we employ the variant proposed by OpenAI [39]. ES is a type of Genetic Algorithm, a black-box optimization metaheuristic loosely based on natural selection. The basic idea behind ES is to have a diverse population of ANNs implementing π. These ANNs are (i) mutated, (ii) their fitness evaluated, and later (iii) recombined to (iv) form better populations. By continuously repeating this procedure, the performance of π tends to improve. See Algorithm 1 for more details on how ES is applied to the problem at hand. One of the main benefits of ES, as compared to gradient-based optimization methods, lies in the invariability of the former to the frequency at which the agent (i.e. the gateway) acts in the environment [39]. Since Poisson-based traffic is considered, packets from nodes can be generated at any time; thus, the time resolution of the problem is critical. In Markov Decision Process-based RL algorithms (most of the gradient-based approaches), time is discretized and thus, packets are only allowed to be generated at discrete instants, which is an assumption that does not hold true in real situations.

C. Dimensionality reduction

As indicated by many works [37], [39], [40], ES-like methods are more efficient in low-dimensional problems; that is, for ANNs with a low-dimensional parameter space (θ). Since the dimension of θ largely depends on the size of the input $\mathbf{S}$ (denoted as $|\mathbf{S}|$), and this rapidly scales with the number of IoT nodes in the network, we propose implementing a dimensionality reduction technique to speed up ES convergence. Instead of directly feeding the ANN with the state of the network $s \in \mathbf{S}$, this state is first mapped to a new smaller space $s' \in \mathbf{S'}$, with $|\mathbf{S'}|<|\mathbf{S}|$ –thus effectively reducing the dimensionality of θ–. Furthermore, by forcing this new low-dimensional, compact space to have a fixed size, regardless of the number of IoT nodes in the network, the computed optimal policies can be reused in different-sized networks. Since computing π^* might potentially take several hours, or even days, being able to reuse π^* dramatically facilitates the applicability of the proposed solution to real-world scenarios. The reutilization of RL policies is known as transfer learning [40], and is a major current area of research.

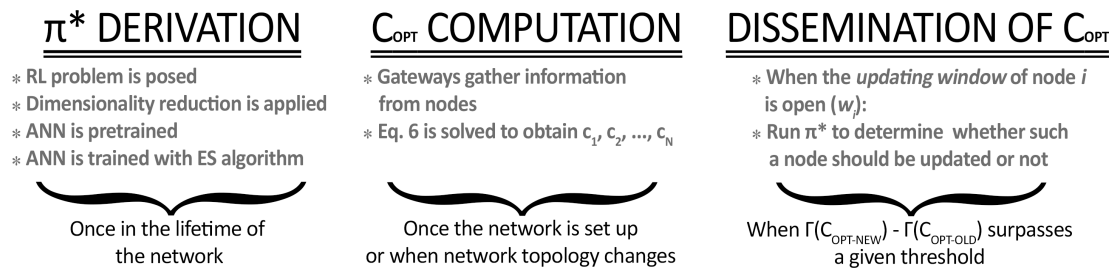


Fig. 2: Steps involved in the proposed solution along with the frequency in which they are executed.

The mapping from the original state space $\mathbf{S}$ to the new state space $\mathbf{S}'$ has been chosen to: (i) permit fixed-length representations of s', (ii) make it computable in real time and in resource-constrained devices such as gateways, and (iii) be expressive enough to derive π^*. Considering these properties, we have opted for a hand-crafted size-independent mapping between $\mathbf{S}$ and $\mathbf{S}'$. Learning-based/data-driven methods such as Variational Autoencoders [41] or Principal Component Analysis (PCA) [42] have been discarded due to the amount of data and computational resources required to compute adequate mappings. Also, note that with these methods, every time that an IoT mote joined the network, the state space $\mathbf{S}$ would change and thus, a new mapping from $\mathbf{S}$ to $\mathbf{S}'$ would have to be learned by the gateway, which again, is a time/data consuming task. Eq. 7 describes the state space $\mathbf{S}'$ and how to compute it. This state $s' \in \mathbf{S}'$ is fed into the ANN policy network every time the gateway has to make a decision on whether to update a certain node (i), or not (i.e. s' is the input of the ANN):

$$\mathbf{S}' = \begin{cases} \lambda_i, & \widetilde{\lambda_i} \\ \Gamma_i, & \widetilde{\Gamma_i} \\ TDCC_i, & \widetilde{TDCC_i} \\ G_i, & \widetilde{G_i} \\ TDC & \\ N & \end{cases} \tag{7}$$

The operator $\widetilde{\cdot}$ applies a min-max normalization[1] as follows (an example is given for λ, but the same procedure also applies to the rest of the variables): $\widetilde{\lambda_i} = \frac{\lambda_i - \min_{\forall j \in N_{\overline{u}}} \lambda_j}{\max_{\forall j \in N_{\overline{u}}} \lambda_j - \min_{\forall j \in N_{\overline{u}}} \lambda_j}$.

That is, $\widetilde{\lambda_i}$ indicates the relative packet generation rate of node i with respect to the rest of non-updated nodes $N_{\overline{u}}$. In turn, Γ_i, represents the average effective throughput per node obtained when node i is updated. $TDCC_i$ reflects the TDC consumption (in seconds) of updating node i, and G_i, again, specifies the importance of packets generated by node i. Therefore, a given state s' consists of absolute values related to the node and the impact of updating it (λ_i, Γ_i, $TDCC_i$, and G_i), their *relative* counterparts ($\widetilde{\lambda_i}$, $\widetilde{\Gamma_i}$, $\widetilde{TDCC_i}$, and $\widetilde{G_i}$), and the state of the gateway in terms of the remaining TDC and total number of nodes (N). Therefore, $|\mathbf{S}'|$ (the size of $\mathbf{S}'$), equals 10. Note that for networks with more than 5 IoT nodes, a very common situation in LoRa deployments, $|\mathbf{S}'|$ is effectively smaller than $|\mathbf{S}|$ –as $|\mathbf{S}| = 2N + 1$–.

[1]The reason why we have opted for applying a normalization technique is to make NN policies more re-usable by abstracting their input from the actual scale of λ, Γ, $TDCC$, and G. Specifically, we have employed min-max normalization to force inputs to lay within the range $[0, 1]$, thus, having a very concise and predefined input range.

D. Pre-training to speed up the learning process

To further accelerate the learning process, the ANN is pre-trained by using a variant of the teacher-student approach proposed in [43]. The ANN is first taught and encouraged to update nodes with high probability. Then, the ES-based learning process learns to discriminate which nodes should really be updated. Note that by first teaching the ANN the perks of always updating nodes and later letting them decide which updating opportunities should be discarded, the learning process is speeded up by a factor of 8 in our experiments.

V. Frequency of the involved steps

For readers to have a global outlook of the presented approach, we describe here the main steps into which it can be divided (see Fig. 2). First, the optimal updating policy π^* is derived following the steps described in Section IV. This process has been designed to be carried out only once in the lifetime of the IoT network as it is a time-consuming process. The second step, the derivation of C_{OPT}, presented in Section III, should be performed once the network is up and every time a node is added or removed from the network. When the new $C_{OPT-new}$ is computed, the performance under such a new global configuration $\Gamma(C_{OPT-new})$ is compared to the performance under the old configuration $\Gamma(C_{OPT-old})$. If this difference exceeds a given threshold γ, a new CU process is triggered (completing the last step in Fig. 2). Then, each time an updating window is open in a node, the gateway will use π^* to compute the most adequate action. Note that the usage of the policy (not its derivation) requires a single forward pass of the ANN, a process which is known to be extremely fast –a fraction of a second even in severely hardware-constrained devices– [44]. The above mentioned threshold γ, is a design value that trades off consumption of TDC for Network Performance. Lower threshold values would keep the network better updated at the expense of consuming more TDC of the gateway. Finally, to carry out the CU process, and from an implementation perspective, LoRa gateways could use the second reception window (RW2) and potentially, a different frequency channel to further avoid collisions between uplink and downlink traffic –as suggested in [45]–.

VI. Implementation and evaluation of the proposed solution

A. Implementation details

For the derivation of $\mathbf{C_{OPT}}$, Eq. 5 must be solved. We have done so by employing Sequential Least Squares programming (SQP), an iterative method for constrained nonlinear optimization problems [46]. Note that other optimization algorithms could be adopted, and a further analysis on the impact of this algorithm is left as a future work. SQP optimizes a (surrogate) quadratic model of the objective function, forcing both problem constraints and the objective function itself to be twice continuously differentiable. Although faster parallel implementations of this algorithm are available [47], for IoT networks of up to 80-100 nodes, the computation of $\mathbf{C_{OPT}}$ does not take more than 2-3 minutes with a normal desktop computer[2] and thus, the non-parallel SciPy implementation has been used to foster usability in single-core machines like LoRa Gateways [48]. Should the process be speeded up, LoRa gateways can offload the computation of $\mathbf{C_{OPT}}$ to cloud-based servers.

Regarding the CU process, different architectures of the ANN that models π have been tested. The best trade-off between results and training times has been obtained for ANNs with two hidden layers of 45 and 5 neurons respectively. ReLUs have been selected as the activation function of the hidden layers and the logistic function has been chosen as the activation function of the output, since there is only one output neuron indicating whether or not to update the IoT node. The size of the input of the ANN corresponds to the dimension of $\mathbf{S}'$ plus a bias term, that is, 11 neurons. Note that, thanks to the reduced size of the state space $\mathbf{S}'$, the total number of weights (size of θ) is only 776, a relatively small value which dramatically speeds up the convergence of the ES algorithm.

To derive π^*, a population of 101 ANNs evolves for 1000 iterations (following Algorithm 1). Each iteration simulates a twelve-hour CU process in 128 different randomly-generated IoT networks with a particular ANN-based updating policy π. These simulations have been carried out using the SimPy simulator [49], a discrete-event Python simulator. The size of this IoT network varies in a range from 20 to 200 nodes, and the specific parameters of such nodes are described in Table I. This process took 58 hours with an 8-core Intel Xeon server. It is worth remarking that the process of deriving π^* is carried out just once in the lifetime of an IoT network and that it can be shortened by reducing the number of iterations (at the expense of potentially achieving slightly lower throughput improvements).

Regarding Algorithm 1, our results (shown in Fig. 3) use $N_{its} = 1000$, $N_{pop} = 101$, $\alpha = 0.01$, and $\sigma = 0.1$. To evaluate the performance of each $\pi(\tilde{\theta})$ (line 13 of the pseudo-code), 128 different randomly-generated IoT networks are simulated using the parameters specified in Table I. Finally, since the proposed mathematical model allows arbitrary node distributions, the node SNR values must be provided. In this regard, we opt for giving values to this variable instead of assuming a particular node distribution and employing a path-loss model to obtain such SNR values, which would diffuse the idea of the proposed model being distribution-agnostic.

[2]We tested it in a desktop computer fitted with an i5-7400 CPU, where just one core was used.

Parameter	Value
Packet generation rate (λ)	$U(0.01, 2)$ packets/s
Importance of generated packets (G)	$U(0, 1)$
Signal-to-Noise ratio (SNR)	$U(-23, 23)$ dB
Transmission power	14dBm (maximum)
Length of packet payloads	$U(15, 30)$ bytes
Frequency channels	2 (1 uplink + 1 downlink)
Header length	13 bytes
Bandwidth of LoRa channels	125KHz
Spreading Factors	7, 8, 9, 10, 11, 12

TABLE I: Variables for each IoT node of the network. $U(a, b)$ represents a uniform distribution ranging from a to b.

Algorithm 1 Derivation of π^* via ES algorithm and teacher-student pre-training

1: **Input:** Initial weight vector θ (of size $|\theta|$) derived
2: by the teacher-student approach
3: σ: Standard deviation for the noise
4: α: Learning Rate
5: N_{its}: Number of iterations
6: N_{pop}: Population size
7: **Output:** The weight vector θ of the ANN-modeled π^*
8:
9: **for** it=1 **to** it=N_{its} **do**
10: $\Phi \leftarrow$ Generate N_{pop} perturbation vectors with $N\left[0, \sigma \cdot \mathbf{I}_{|\theta|}\right]$
11: $R \leftarrow$ empty list
12: **for** j=1 **to** j=N_{pop} **do**
13: $\tilde{\theta} \leftarrow \theta + \Phi[j]$ ▷ Perturb θ to generate a new candidate
14: $r \leftarrow$ evaluate performance of $\pi(\tilde{\theta})$
15: $R \leftarrow R + r$ ▷ Store the performance of this candidate
16: **end for**
17: $A = \frac{R - E[R]}{std(E)}$ ▷ Compute the normalized performance of each candidate
18: $\theta = \theta + (\frac{\alpha}{N_{pop} \cdot \sigma} \cdot \Phi^T A)$ ▷ Combine them to generate new θ
19: **end for**

B. Results, methodology and comparisons

With the objective of highlighting the benefits of our proposal and obtaining a deeper insight into the individual impact of $\mathbf{C}$ and π, reasonable alternatives for the global configuration, as well as for the updating policy, are also analyzed:

- The alternative $\mathbf{C}$ is obtained by running the Adaptive Data Rate (ADR) algorithm; yielding $\mathbf{C_{ADR}}$. ADR is widely employed in LoRaWAN networks [7] and it is the de-facto mechanism for adapting transmission configurations in such networks. Theoretically, the ADR is a mechanism for optimizing data rates and ToA in the network [50]. The LoRaWAN network server is in charge of running the ADR algorithm and making LoRa gateways inform IoT nodes of the configuration that should be employed.
- The alternative updating policy is an *always update* policy (π_{au}). Every time the updating window of a node i is open ($w_i = 1$), its new configuration $\mathbf{c_i}$ is sent. This also represents the most traditional way of updating the configuration of IoT nodes.

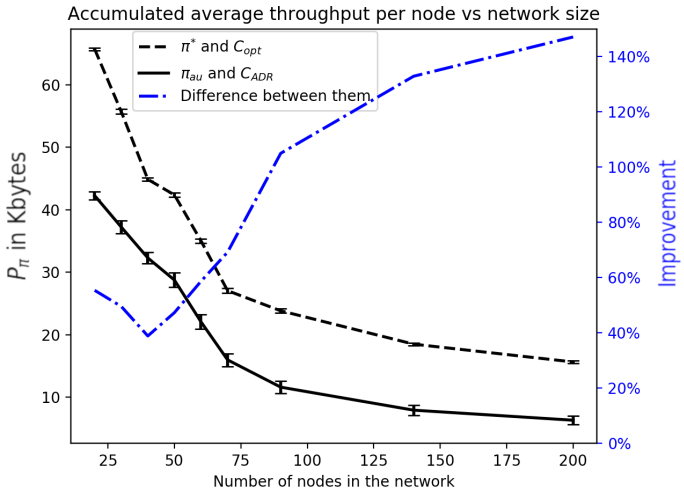


Fig. 3: Accumulated average throughput per node (in Kbytes) obtained under: the proposed approach ($\mathbf{C_{OPT}}$ and π^*) and the alternative one ($\mathbf{C_{ADR}}$ and π_{au}). Differences between them are also covered to illustrate the improvement derived from using the proposed approach. Standard deviation around the mean values are included as error bars.

Thus, the results attained under the proposed solution (which includes the computation of a global network configuration, $\mathbf{C_{OPT}}$, and its proper dissemination via the adequate updating policy π^*) are compared to those obtained under the alternative approach based on using $\mathbf{C_{ADR}}$ as the network configuration and updating nodes according to π_{au}. Note that if a non-optimal $\mathbf{C}$ is found, despite intelligently updating the configuration of nodes via π^*, the attained accumulated per-node throughput (P_π) will be sub-optimal. Similarly, if the true $\mathbf{C_{OPT}}$ is computed but nodes are updated non-optimally, P_π will also be degraded. Therefore, the performance of $\mathbf{C}$ plus π can be jointly and effectively assessed by P_π.

To obtain (and compare) the aforementioned results, we have employed SimPy to simulate different-sized networks (20 to 200 nodes). Every network simulation has been repeated with 100 different random seeds to obtain solid average and standard deviation values. Each node of the simulated networks presents a different value of packet generation rate (λ), packet importance (G), SNR and length of generated packets (L). These values are realizations of the random variables described in Table I; values that are usually found in LoRaWAN networks [22]. As a figure of merit for comparisons, the accumulated throughput per node (P_π) over the simulated twelve-hour period is used (see Eq. 6). Furthermore, all simulations start with nodes being assigned equally-probable transmission configurations $c_i^k = \frac{1}{K}$.

Fig. 3 represents P_π for the proposed and alternative solutions in networks ranging from 20 to 200 nodes. The differences between both approaches are also shown (in blue) to illustrate the remarkable improvement derived from using our solution. Results indicate that it is in larger networks where our solution truly excels, attaining an improvement of 147% for 200 nodes. In fact, this performance enhancement seems to

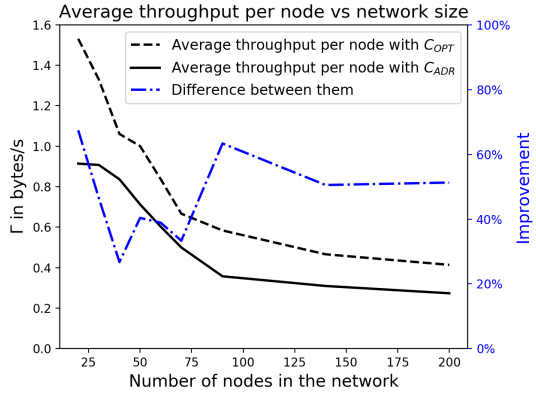


Fig. 4: Average throughput per node (in bytes/s) obtained when $\mathbf{C_{ADR}}$ (solid black line) and the proposed $\mathbf{C_{OPT}}$ (dashed black line) are used. Relative improvement from using $\mathbf{C_{OPT}}$ is also shown (blue line).

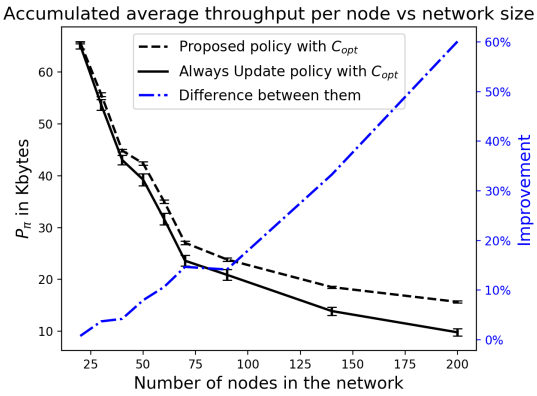


Fig. 5: Average accumulated throughput per node (in Kbytes) obtained when π_{au} (solid black line) and the proposed π^* (dashed black line) are used. Relative improvement from using π^* is also shown (blue line).

consistently increase with the number of nodes, although this improvement might eventually saturate (as can be derived from the behavior of P_π for both approaches). By fully considering the TDC limitation, which becomes increasingly important in larger networks, our RL-based proposal consistently outperforms the most traditional way of managing LoRaWAN networks. Another result worth remarking on, is the progressive reduction of the average accumulated throughput per node in more populated networks. This can be explained by considering that packet collisions progressively increase when the number of nodes grows, despite the *capture effect* reducing them. However, to validate this and to gain a deeper insight into the results, the effect of $\mathbf{C}$ has been individually analyzed. The attained average throughput per node (Γ) is computed when $\mathbf{C_{ADR}}$ and $\mathbf{C_{OPT}}$ are adopted.

Fig. 4 illustrates the Γ values obtained for different network sizes and the two $\mathbf{C}$ configurations. It can be seen how the average throughput per node (expressed in bytes per second) decreases as the number of nodes grows for both $\mathbf{C_{OPT}}$ and $\mathbf{C_{ADR}}$. This demonstrates the hypothesis that the reduction in P_π is derived from an increase in the number of nodes, and with it, an increase in the prevalence of collisions and a reduction of the available throughput per node. Nevertheless,

a consistent increase of between 26% to 67% in the value of Γ can be observed when the proposed $\mathbf{C_{OPT}}$ is preferred over $\mathbf{C_{ADR}}$ (the network configuration derived from the ADR mechanism). Also note that the relatively small absolute values of Γ appreciated across all network sizes and $\mathbf{C}$ configurations are due to the very low data-generation rates of simulated networks (which are obviously in line with the true nature of LoRaWAN networks).

To gain a better understanding of the individual effect of a good updating policy, π^* and π_{au} policies have been used to update the same network configuration ($\mathbf{C_{OPT}}$). Fig. 5 illustrates the attained P_π for different network sizes. Standard deviation (std) around the mean values are included as error bars. Again, steady improvement is observed as the number of nodes grows, thus demonstrating the benefits of the proposed RL-based updating policy, which even with the same network configuration ($\mathbf{C}$), leads to an increase of up to 60% in P_π.

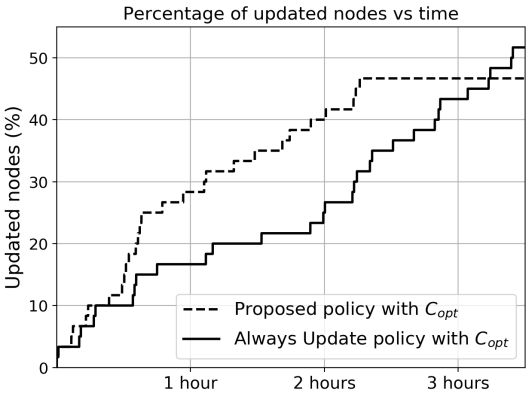
C. Evolution of the CU process

A time analysis can also be performed on the Configuration Updating (CU) process to further appreciate the benefits of the proposed RL-based updating policy. Fig. 6 represents the first three hours of the CU process from a threefold perspective: the percentage of updated nodes with time is depicted in Fig. 6a, the evolution of Γ is presented in Fig. 6b, and P_π is shown in Fig. 6c.

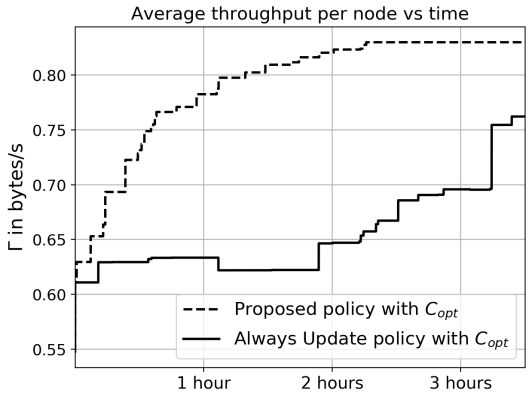
It is worth noting that although the raw number of updated nodes has a direct impact on P_π, it is not a determining factor, and therefore, at some points π_{au} disseminates $\mathbf{C}$ faster than π^* (as can be seen at around 3 hours). However, the global network-level effect of updating nodes is disregarded under the π_{au} policy, and as such, in Fig. 6b it can be seen how Γ can even decrease when certain nodes are updated (this is clearly observed at around 1.5 hours). Our proposed method not only considers the global network-level effect of updating individual nodes but also the rate at which the updating windows tend to appear, as well as the remaining TDC. Another important fact derived from the time analysis of π^* is that empirical experiments show that, looking at the scale of hundredths of a byte ($\frac{1}{100}$ byte), Γ monotonically increases as the CU process completes. This, which does not hold true for π_{au}, makes our RL-based approach an *anytime method*. In other words, should the CU process be interrupted, the network throughput at time t will always be higher or equal to the throughput at time t' (with $t' < t$). Again, this is looking at a scale of $\frac{1}{100}$ of a byte. This allows LoRa Gateways to stall the CU process if they need to generate any downlink traffic (for which some TDC must be expended), and then continue the CU process where they have left it. However, if π_{au} is followed, pausing the CU process may lead to degenerated, undesirable network states s, in which Γ is severely degraded (as a result of a temporary increase in the number of packet collisions).

D. The effect of adding/removing nodes on C_{OPT}

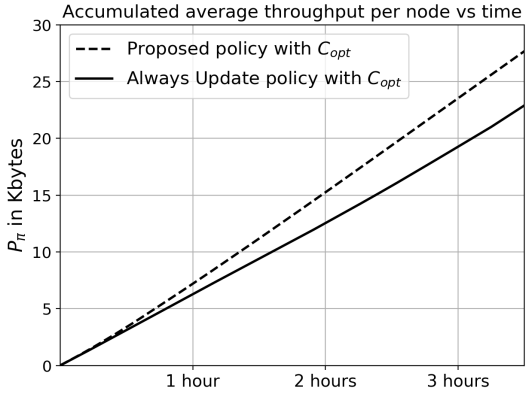
When new nodes are added or removed from the network, a new C_{OPT} originates. To speed up the computation of such a new global configuration, the SQP optimization algorithm can be started with the previous C_{OPT}, which would act as a *good* initialization point. From a network-wise point of view, it is also interesting to analyze how the distribution of Spreading Factors changes as more nodes are incorporated into the network. To this end, a base LoRaWAN network of N=40 nodes is simulated with the parameters described in Table II. Then, we repeat the following experiment three times: 8 more nodes are added to the network and the distribution of SFs is analyzed again. That is, we evaluate the base network with



(a)



(b)



(c)

Fig. 6: First 3 hours of the CU process for a randomly generated 60-node network. (a) Percentage of updated nodes. (b) Average throughput per node. (c) Accumulated average throughput per node.

Parameter	Value
Packet generation rate (λ)	$\frac{1}{60}$ packets/s
Importance of generated packets (G)	1
Signal-to-Noise ratio (SNR)	$U(-23, 23)$ dB
Length of packet payloads	30 bytes

TABLE II: Variables for each IoT node of the base network. To isolate the effect of the number of nodes on the network, most variables are fixed, as opposed to the experiment presented in Section VI-A

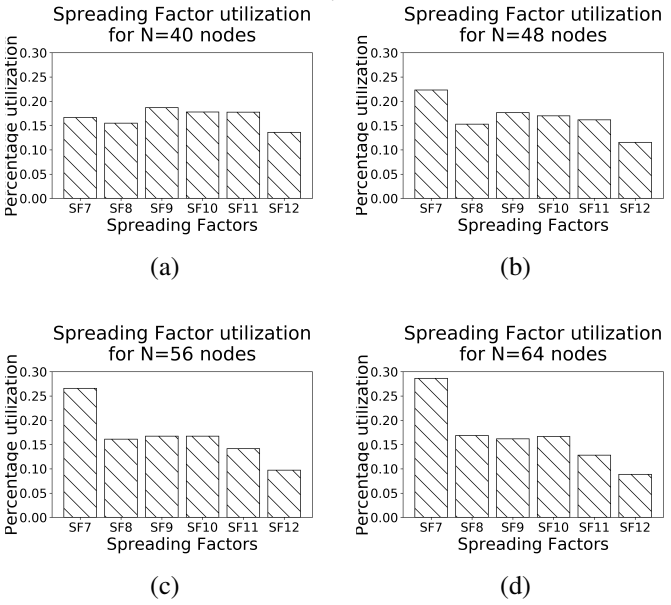


Fig. 7: Distribution of Spreading Factors (percentage of packets sent under a particular SF) for the base network described in Table II for (a) N=40 nodes, (b) N=48 nodes, (c) N=56 nodes, and (d) N=64 nodes.

N=48, N=56, and N=64 nodes. Figure 7 depicts the result of this experiment. The Y axis represents the percentage of all packets sent under a particular SF. For example, a value of 0.17 for SF7 (as in Fig. 7A) indicates that 17% of all sent packets (in the network) are transmitted by using SF7.

When network congestion increases due to the presence of more nodes, shorter ToAs are preferred to reduce collisions. This ultimately tips the balance in favor of smaller SFs, as can be clearly appreciated with the prevalence of SF7 in Fig. 7c and Fig. 7d. It is also worth noting that due to the unavoidable increase in network congestion, per-node performance decreases 1.1%, 1.4%, and 1.6% when network size increases to 48, 56, and 64 nodes respectively. The interesting fact behind this decrease in performance is that (i) it is small and (ii) it is non-linear, i.e. adding 16 nodes does not reduce the performance two times more than adding 8 nodes. These two facts indicate the good scalability of LoRaWAN networks, which can easily accommodate new nodes by exploiting shorter SFs when the true C_{OPT} is derived.

E. The effect of node importance G_i on C_{OPT}

Similar to the number of nodes in the network, the importance of packets generated by a node i (G_i) has a measurable effect on its optimal transmission configuration $\mathbf{c_i}$. More

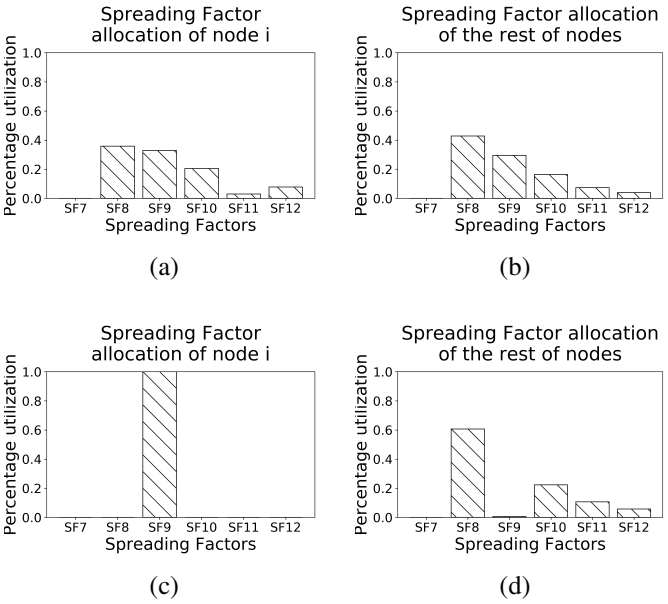


Fig. 8: Distribution of SFs for the base network described in Table II. Distribution of SFs for the node under study (shown in (a)) and the rest of network nodes (shown in (b)) when all generate packets of importance $G = 1$. Distribution of SFs for the node under study (shown in (c)) and the rest of network nodes (shown in (d)) when such a node generates packets of importance $G = 30$.

important LoRa nodes are expected to be assigned to emptier SFs, or even make other unimportant nodes leave crucial SFs.

This can be tested by employing the base network presented in the previous subsection, and while leaving all the variables fixed, vary the importance G of the packets generated by a given node i. We present the results of this experiment in Fig. 8. Again, the Y axis represents the percentage of all packets sent under a particular SF. In the first column, the SF distribution of a given node i is depicted. In the second column, the distribution of SFs of the other 39 nodes of the network is shown. Conversely, the first row shows the result of the simulations when said node generates packets with importance $G = 1$ (as the rest of nodes in the network), and the second row covers the results when such a node generates packets of importance $G = 30$ (leaving the rest of nodes with $G = 1$). It can be easily appreciated how, when all nodes generate packets of the same importance, the SF distributions of the node under study and the rest of the network nodes are almost the same. Conversely, when such a node gains in importance, the C_{OPT} derivation process favors the most important node by assigning it to a previously emptied SF (the SF9). This can have many advantages over traditional node-centric approaches (such as ADR) and is the result of incorporating the importance of each node in the process of deriving a global network configuration. Furthermore, it is worth noting that importance values can be changed over time, deriving a truly optimal network configuration that can react to different requirements of network designers.

VII. Conclusions and Future Works

Based on a thorough analysis of the *capture effect* and the LoRa modulation scheme, we have mathematically modeled the average per-node throughput of LoRaWAN networks. This model acknowledges the heterogeneity of IoT deployments by letting each constituent node generate packets of different importance/length and at a different rate, while not enforcing any particular node distribution. Thanks to this analytical approach, the optimization of network performance has been posed as a maximization problem whose solution yields optimal network-level configurations. The problems derived from updating/disseminating this centrally-computed configuration to end nodes have been solved by applying a set of techniques from the Reinforcement Learning (RL) field. Specifically, we have made use of the Evolution Strategies (ES) algorithm to derive optimal disseminating policies that aim to maximize the accumulated average per-node throughput. Furthermore, by designing a compact representation of the inner state of the IoT network, the derived policies can be employed in IoT deployments of different size; thus allowing us to reuse the time-consuming learned policies. The training times of the ES algorithm have also been largely reduced by employing a teacher-student approach that encourages policies to first consider the action of disseminating new configurations.

The RL-based updating policies, together with the optimal network-level configuration, have been compared to the de-facto alternative: using the LoRaWAN ADR mechanism and updating nodes greedily. Results show a remarkable increase in the accumulated average per-node throughput of 147% when the network is composed of 200 IoT nodes. These results stem from two sources: (i) using an optimal network-level configuration and (ii) optimally updating nodes, as demonstrated in the experiments conducted. Finally, we have analyzed the dissemination process from a time perspective and shown that our approach can be considered an *anytime updating method*; thus letting LoRa gateways pause the updating process without network degradation, should some downlink traffic be generated.

As part of future work, we plan to experimentally study the impact (in terms of power consumption and CPU usage) of the proposed RL algorithm on commercial LoRa gateways. Also, we plan to study how our proposal could be integrated in LoRaWAN networks with multiple gateways (e.g. aggregating transmission configurations from different gateways in a single updating packet, hence, reducing network congestion). Finally, and as indicated in Section VI, the impact of the optimization algorithm on computing $\mathbf{C_{OPT}}$ will be further studied.

References

[1] R. S. Sinha, Y. Wei, and S. H. Hwang, "A survey on LPWA technology: LoRa and NB-IoT," *ICT Express*, vol. 3, no. 1, pp. 14–21, 2017.

[2] LoRa-Alliance, "LoRa," 2017. [Online]. Available: https://www.lora-alliance.org/

[3] S. Persia, C. Carciofi, and M. Faccioli, "NB-IoT and LoRA connectivity analysis for M2M/IoT smart grids applications," in *2017 AEIT International Annual Conference*, sep 2017, pp. 1–6.

[4] J. Haxhibeqiri, F. den Abeele, I. Moerman, and J. Hoebeke, "LoRa Scalability: A Simulation Model Based on Interference Measurements," *Sensors*, vol. 17, no. 6, 2017.

[5] LoRa Alliance, "A technical overview of LoRa and LoRaWAN," Technical Marketing Workgroup 1.0, Tech. Rep. 1, nov 2015. [Online]. Available: https://www.tuv.com/media/corporate/products_1/electronic_components_and_lasers/TUeV_Rheinland_Overview_LoRa_and_LoRaWANtmp.pdf

[6] ETSI, "Final draft ETSI EN 300 220-1 V2.4.1 (2012-01)," ETSI, Tech. Rep. REN/ERM-TG28-434, 2012.

[7] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things," *Sensors*, vol. 16, no. 9, 2016.

[8] N. E. Rachkidy, A. Guitton, and M. Kaneko, "Decoding Superposed LoRa Signals," *CoRR*, vol. abs/1804.0, 2018.

[9] C. A. Trasviña-Moreno, R. Blasco, R. Casas, and Á. Asensio, "A Network Performance Analysis of LoRa Modulation for LPWAN Sensor Devices," in *Ubiquitous Computing and Ambient Intelligence*, C. R. García, P. Caballero-Gil, M. Burmester, and A. Quesada-Arencibia, Eds. Cham: Springer International Publishing, 2016, pp. 174–181.

[10] D. Bankov, E. Khorov, and A. Lyakhov, "Mathematical model of LoRaWAN channel access," in *18th IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks, WoWMoM 2017 - Conference*, no. 15. IEEE, jun 2017, pp. 1–3.

[11] D. Bankov, E. Khorov, and A. Lyakov, "Mathematical model of LoRaWAN channel access with capture effect," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, vol. 2017-Octob, no. 14. IEEE, oct 2017, pp. 1–5.

[12] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the Limits of LoRaWAN," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[13] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa Low-Power Wide-Area Networks Scale?" in *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWiM '16. New York, NY, USA: ACM, 2016, pp. 59–67.

[14] O. Georgiou and U. Raza, "Low Power Wide Area Network Analysis: Can LoRa Scale?" *IEEE Wireless Communications Letters*, vol. 6, no. 2, pp. 162–165, apr 2017.

[15] B. Reynders, W. Meert, and S. Pollin, "Power and spreading factor control in low power wide area networks," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, may 2017, pp. 1–6.

[16] K. Q. Abdelfadeel, V. Cionca, and D. Pesch, "Fair Adaptive Data Rate Allocation and Power Control in LoRaWAN," *CoRR*, vol. abs/1802.1, feb 2018.

[17] D. Zorbas, G. Z. Papadopoulos, P. Maille, N. Montavont, and C. Douligeris, "Improving LoRa Network Capacity Using Multiple Spreading Factor Configurations," in *25th International Conference on Telecommunications*, no. June, Saint-Malo, France, 2018, pp. 3–8.

[18] F. Cuomo, M. Campo, A. Caponi, G. Bianchi, G. Rossini, and P. Pisani, "EXPLoRa: Extending the performance of LoRa by suitable spreading factor allocations," in *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, oct 2017, pp. 1–8.

[19] M. Bor and U. Roedig, "LoRa Transmission Parameter Selection," in *Proceedings of the 13th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), Ottawa, ON, Canada*, 2017, pp. 5–7.

[20] C. Pham, "QoS for Long-Range Wireless Sensors Under Duty-Cycle Regulations with Shared Activity Time Usage," *ACM Trans. Sen. Netw.*, vol. 12, no. 4, pp. 33:1—-33:31, sep 2016.

[21] C. Phan, "Deploying a pool of long-range wireless image sensor with shared activity time," in *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, oct 2015, pp. 667–674.

[22] R. M. Sandoval, A.-J. Garcia-Sanchez, J. Garcia-Haro, and T. M. Chen, "Optimal policy derivation for Transmission Duty-Cycle constrained LPWAN," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 1–1, aug 2018.

[23] A. Mahmood, E. Sisinni, L. Guntupalli, R. Rondon, S. A. Hassan, and M. Gidlund, "Scalability Analysis of a LoRa Network under Imperfect Orthogonality," *IEEE Transactions on Industrial Informatics*, p. 1, 2018.

[24] A. Waret, M. Kaneko, A. Guitton, and N. E. Rachkidy, "LoRa Throughput Analysis with Imperfect Spreading Factor Orthogonality," mar 2018.

[25] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita, and I. Tinnirello, "Impact of LoRa Imperfect Orthogonality: Analysis of Link-Level Performance," *IEEE Communications Letters*, vol. 22, no. 4, pp. 796–799, apr 2018.

[26] J. Finnegan and S. Brown, "A Comparative Survey of LPWA Networking," *CoRR*, vol. abs/1802.0, feb 2018.
[27] R. B. Sørensen, D. M. Kim, J. J. Nielsen, and P. Popovski, "Analysis of Latency and MAC-layer Performance for Class A LoRaWAN," *IEEE Wireless Communications Letters*, vol. PP, no. 99, p. 1, 2017.
[28] H. Byun and J. Yu, "Adaptive Duty Cycle Control with Queue Management in Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 6, pp. 1214–1224, jun 2013.
[29] J. Zhu, Y. Song, D. Jiang, and H. Song, "A New Deep-Q-Learning-Based Transmission Scheduling Mechanism for the Cognitive Internet of Things," *IEEE Internet of Things Journal*, vol. PP, no. 99, p. 1, 2017.
[30] M. A. Alsheikh, D. T. Hoang, D. Niyato, H. P. Tan, and S. Lin, "Markov Decision Processes With Applications in Wireless Sensor Networks: A Survey," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1239–1267, 2015.
[31] F. Van den Abeele, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Scalability Analysis of Large-Scale LoRaWAN Networks in ns-3," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2186–2198, dec 2017.
[32] A. Azari and C. Cavdar, "Self-organized Low-power IoT Networks: A Distributed Learning Approach," *ArXiv e-prints*, jul 2018.
[33] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. C. Chen, and L. Hanzo, "Machine Learning Paradigms for Next-Generation Wireless Networks," *IEEE Wireless Communications*, vol. 24, no. 2, pp. 98–105, apr 2017.
[34] D. Magrin, M. Centenaro, and L. Vangelista, "Performance evaluation of LoRa networks in a smart city scenario," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, may 2017, pp. 1–7.
[35] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.
[36] A. Geramifard, T. J. Walsh, S. Tellex, G. Chowdhary, N. Roy, J. P. How, and Others, "A tutorial on linear function approximators for dynamic programming and reinforcement learning," *Foundations and Trends®in Machine Learning*, vol. 6, no. 4, pp. 375–451, 2013.
[37] M. P. Deisenroth, G. Neumann, and J. Peters, "A Survey on Policy Search for Robotics," *Found. Trends Robot*, vol. 2, no. 1–2, pp. 1–142, aug 2013.
[38] W. Vent, "Rechenberg, Ingo, Evolutionsstrategie — Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. 170 S. mit 36 Abb. Frommann-Holzboog-Verlag. Stuttgart 1973. Broschiert," *Feddes Repertorium*, vol. 86, no. 5, p. 337.
[39] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution Strategies as a Scalable Alternative to Reinforcement Learning," *ArXiv e-prints*, mar 2017.
[40] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A Brief Survey of Deep Reinforcement Learning," *CoRR*, vol. abs/1708.0, aug 2017.
[41] C. Doersch, "Tutorial on Variational Autoencoders," *ArXiv e-prints*, jun 2016.
[42] R. Tripathy and I. Bilionis, "Deep {UQ:} Learning deep neural network surrogate models for high dimensional uncertainty quantification," *CoRR*, vol. abs/1802.0, 2018.
[43] S. Schmitt, J. Hudson, A. Zidek, S. Osindero, C. Doersch, W. Czarnecki, J. Leibo, H. Kuttler, A. Zisserman, K. Simonyan, and S. Eslami, "Kickstarting Deep Reinforcement Learning," *ArXiv e-prints*, mar 2018.
[44] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, ser. Prentice Hall series in artificial intelligence. Prentice Hall, 2016.
[45] K. Mikhaylov, J. Petäjäjärvi, and A. Pouttu, "Effect of downlink traffic on performance of lorawan lpwa networks: Empirical study," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2018, pp. 1–6.
[46] D. Kraft, "A software package for sequential quadratic programming," Koln, Germany, 1988.
[47] PyGMO, "PyGMO - SLQLP parallel implementation," 2018. [Online]. Available: https://esa.github.io/pygmo/documentation/algorithms.html
[48] SciPy, "SciPy SLSQP implementation," 2008. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html
[49] Simpy, "Event discrete simulation for Python." [Online]. Available: https://simpy.readthedocs.io
[50] TheThingsNetwork, "LoRaWAN Adaptive Data Rate," p. 1, 2016. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/adr.html

Ruben M. Sandoval (M'12) received the B.S. degree in telematics engineering in 2011 and M.S. degree in telecommunication engineering in 2014 from the Universidad Politecnica de Cartagena (UPCT), Cartagena, Spain. He is currently pursuing the Ph.D. degree at Universidad Politecnica de Cartagena (UPCT), Cartagena, Spain. His research interest includes Machine Learning, IoT and the applicability of both paradigms to manage critical environments.

Antonio-Javier Garcia-Sanchez received the M.S. degree and Ph.D. degree from the Universidad Politecnica de Cartagena, Spain in 2000 and 2005 respectively. Since 2001, he has joined the Department of Information Technologies and Communications (DTIC), UPCT. He is a (co)author of more than 50 conference and journal papers, fifteen of them indexed in the Journal Citation Report (JCR). He has been the main head in several research projects in the field of communication networks and optimization. His main research interests are wireless sensor networks (WSNs), and Smart Grids.

Joan Garcia-Haro (M'91) received the M.S and Ph.D degrees in telecommunication engineering from the Universitat Politecnica de Catalunya, Spain, in 1989 and 1995 respectively. He is currently a Professor with the Universidad Politecnica de Cartagena. He is author or co-author of more than 80 journal papers mainly in the fields of switching, wireless networking and performance evaluation. Dr. Garcia-Haro served as Editor-in-Chief for the IEEE Global Communications Newsletter, included in the IEEE Communications Magazine, from April 2002 to December 2004. He has been Technical Editor of the same magazine from March 2001 to December 2011. He also received an Honorable Mention for the IEEE Communications Society Best Tutorial paper Award (1995). He has been a visiting scholar at Queen's University at Kingston, Canada (1991-1992) and at Cornell University, Ithaca, USA (2010-2011).